

达梦技术丛书

# DM8 备份与还原



# 前言

## 概述

本文档主要介绍备份还原的基本概念、技术原理，如何使用 DM 的命令行工具 `DISql` 和 `DMRMAN`（DM RECOVER MANAGER）、客户端工具 `MANAGER`（管理工具）和 `CONSOLE`（控制台工具）进行备份还原，它们所提供的功能、以及详细的参数介绍。

## 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
 <b>警告：</b>	表示可能导致系统损坏、数据丢失或不可预知的结果。
 <b>注意：</b>	表示可能导致性能降低、服务不可用。
 <b>小窍门：</b>	可以帮助您解决某个问题或节省您的时间。
 <b>说明：</b>	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。
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。标题、警告、注意、小窍门、说明等内容均采用粗体。
<>	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](http://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](http://www.dameng.com)

技术服务电话：400-991-6599

技术服务邮箱：[dmtech@dameng.com](mailto:dmtech@dameng.com)

# 目录

<b>1</b>	<b>备份还原简介 .....</b>	<b>1</b>
1.1	概述 .....	1
1.2	基本概念 .....	2
1.2.1	表空间与数据文件 .....	2
1.2.2	重做日志 .....	4
1.2.3	归档日志 .....	5
1.2.4	LSN 介绍 .....	6
1.2.5	包序号介绍 .....	7
1.2.6	检查点 .....	7
1.2.7	备份集 .....	8
1.2.8	备份 .....	9
1.2.9	还原与恢复 .....	13
1.3	介质管理层 .....	14
1.4	备份方式 .....	15
<b>2</b>	<b>备份还原原理 .....</b>	<b>15</b>
2.1	归档说明 .....	16
2.1.1	本地归档 .....	16
2.1.2	远程归档 .....	16
2.1.3	归档切换 .....	17
2.1.4	归档修复 .....	17
2.2	备份 .....	18
2.2.1	数据备份 .....	19
2.2.2	日志备份 .....	20
2.2.3	压缩与加密 .....	21
2.2.4	并行备份 .....	22
2.3	还原与恢复 .....	22
2.3.1	数据还原 .....	23
2.3.2	数据恢复 .....	27
2.3.3	解密与解压缩 .....	30
2.3.4	并行还原 .....	31
2.4	归档日志备份与还原 .....	31
2.4.1	归档日志备份 .....	31
2.4.2	归档日志还原 .....	32
<b>3</b>	<b>备份还原实战 .....</b>	<b>33</b>
3.1	准备工作 .....	33

3.1.1	支持与限制 .....	33
3.1.2	归档配置 .....	35
3.2	使用联机执行 SQL 语句进行备份还原 .....	39
3.2.1	概述 .....	39
3.2.2	数据备份 .....	40
3.2.3	数据备份高级主题 .....	65
3.2.4	管理备份 .....	67
3.2.5	数据还原 .....	85
3.2.6	数据还原高级主题 .....	89
3.3	使用脱机工具 DMRMAN 进行备份还原 .....	90
3.3.1	DMRMAN 概述 .....	91
3.3.2	启动和配置 DMRMAN .....	92
3.3.3	数据备份 .....	99
3.3.4	管理备份 .....	111
3.3.5	数据库还原和恢复 .....	130
3.3.6	表空间还原和恢复 .....	153
3.3.7	归档还原 .....	161
3.3.8	归档修复 .....	164
3.3.9	查看操作日志 .....	165
3.4	使用图形化客户端工具进行备份还原 .....	175
3.4.1	使用 MANAGER 工具进行联机备份还原 .....	175
3.4.2	使用 CONSOLE 工具进行脱机备份还原 .....	186
4	<b>联机拷贝数据库还原恢复 .....</b>	<b>205</b>
4.1	概述 .....	205
4.2	约束与限制 .....	205
4.3	拷贝文件 .....	205
4.4	修改文件路径参数 .....	206
4.5	还原恢复 .....	207
4.6	高级主题 .....	207

# 1 备份还原简介

数据库备份是 DBA 日常最重要的工作内容。备份的主要目的是数据容灾，保证数据的安全性，在数据库发生故障时，通过还原备份集，将数据恢复到可用状态。

本章主要介绍 DM 备份与还原的概述和基本概念，为理解归档配置、备份还原的原理与工具使用奠定基础，内容主要包括：

- 概述
- 基本概念

## 1.1 概述

DM 数据库中的数据存储于数据库的物理数据文件中，数据文件按照页、簇和段的方式进行管理，数据页是最小的数据存储单元。任何一个对 DM 数据库的操作，归根结底都是对某个数据文件页的读写操作。

因此，DM 备份的本质就是从数据库文件中拷贝有效的数据页保存到备份集中，这里的有效数据页包括数据文件的描述页和被分配使用的数据页。而在备份的过程中，如果数据库系统还在继续运行，这期间的数据库操作并不是都会立即体现到数据文件中，而是首先以日志的形式写到归档日志中，因此，为了保证用户可以通过备份集将数据恢复到备份结束时间点的状态，就需要将备份过程中产生的归档日志也保存到备份集中。

还原与恢复是备份的逆过程。还原是将备份集中的有效数据页重新写入目标数据文件的过程。恢复则是指通过重做归档日志，将数据库状态恢复到备份结束时的状态；也可以恢复到指定时间点和指定 LSN。恢复结束以后，数据库中可能存在处于未提交状态的活动事务，这些活动事务在恢复结束后的第一次数据库系统启动时，会由 DM 数据库自动进行回滚。

备份、还原与恢复的关系如图 1.1 所示。

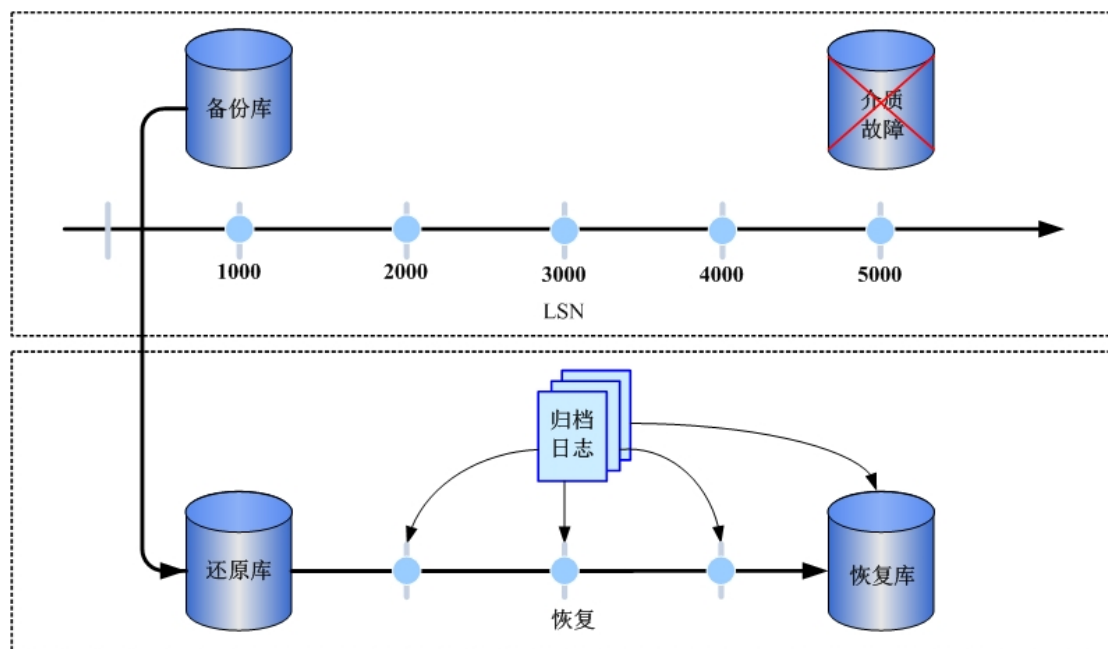


图 1.1 备份、还原与恢复

## 1.2 基本概念

本节介绍与 DM 备份与还原相关的一些基本概念，了解这些基本概念是阅读后续章节的基础。

### 1.2.1 表空间与数据文件

DM 数据库的表空间是一个逻辑概念，其目的主要是为了方便数据库的管理，数据库的所有对象在逻辑上都存放在某个表空间中，而物理上都存储在所属表空间的数据文件中。一个表空间由一个或多个数据文件组成。

数据文件是数据库中最重要文件类型，是真实数据存储的地方。DM 中数据文件的扩展名为 .dbf，分为系统默认生成的数据文件和用户自己创建的数据文件两类。

在创建 DM 数据库时，系统会自动创建 5 个表空间：SYSTEM 表空间、ROLL 表空间、MAIN 表空间、TEMP 表空间和 HMAIN 表空间。DM 自动为这几个自动创建的表空间分别生成默认的数据文件：SYSTEM.DBF、ROLL.DBF、MAIN.DBF 和 TEMP.DBF，HMAIN 表空间没有默认的数据文件。

用户也可以创建自己的表空间，由用户创建的表空间统称为用户自定义表空间，在创建自定义表空间时需要为表空间指定数据文件。用户可通过为已存在的表空间增加数据文件或

创建一个新的表空间来创建数据文件。

因此，DM 中的表空间有以下几类：

### SYSTEM 表空间

存放了 DM 数据库全局字典信息和全局系统数据，是 DM 数据库能够正常运行的必要前提，默认对应数据文件 SYSTEM.DBF。CREATE TABLE 等 DDL 操作会修改 SYSTEM 表空间数据。

### ROLL 表空间

存放 DM 数据库运行过程中产生的所有回滚记录。DM 中几乎所有的数据库修改操作都会生成回滚记录，并保存在 ROLL 表空间的数据文件中。ROLL 表空间是数据库全局对象，不论修改哪一个表空间，生成的回滚记录都是写入 ROLL 表空间，该表空间由系统自动维护，默认数据文件为 ROLL.DBF。

### TEMP 表空间

存放临时表数据以及数据库运行过程中产生的临时数据。在数据库运行过程中，SORT、HASH JOIN 等操作都可能会生成临时结果集，它们作为临时数据存放在 TEMP 表空间中。TEMP 表空间是数据库全局对象，由系统自动维护。若数据库重启，保存在 TEMP 表空间中的所有数据都会丢失。TEMP 表空间的默认数据文件为 TEMP.DBF。

### MAIN 表空间

在创建用户时，如果没有指定默认表空间，系统自动指定 MAIN 表空间为用户默认的表空间。

### HMAIN 表空间

DM 的 HUGE 表空间，用来存放 HUGE 表数据文件。HMAIN 表空间和 HUGE 表相关知识可参考《DM8 系统管理员手册》。

### DM 的物理备份还原不支持：



注意：

1. TEMP 表空间。
2. HUGE 表空间，即数据库备份和还原不包括 HMAIN 和用户创建的 HUGE 表空间；也不支持对 HMAIN 和用户创建的 HUGE 表空间进行表空间备份和还原。DM 提供了 HUGE 表备份还原系统函数支持 HUGE 表的联机备份还原，可参考《DM8\_SQL 语言使用手册》。

### 用户自定义表空间



即用户通过创建表空间相关操作生成的表空间。

图 1.2 显示了一个完整的 DM 数据库的表空间和数据文件分布情况，其中 TSDA 为用户自定义表空间。

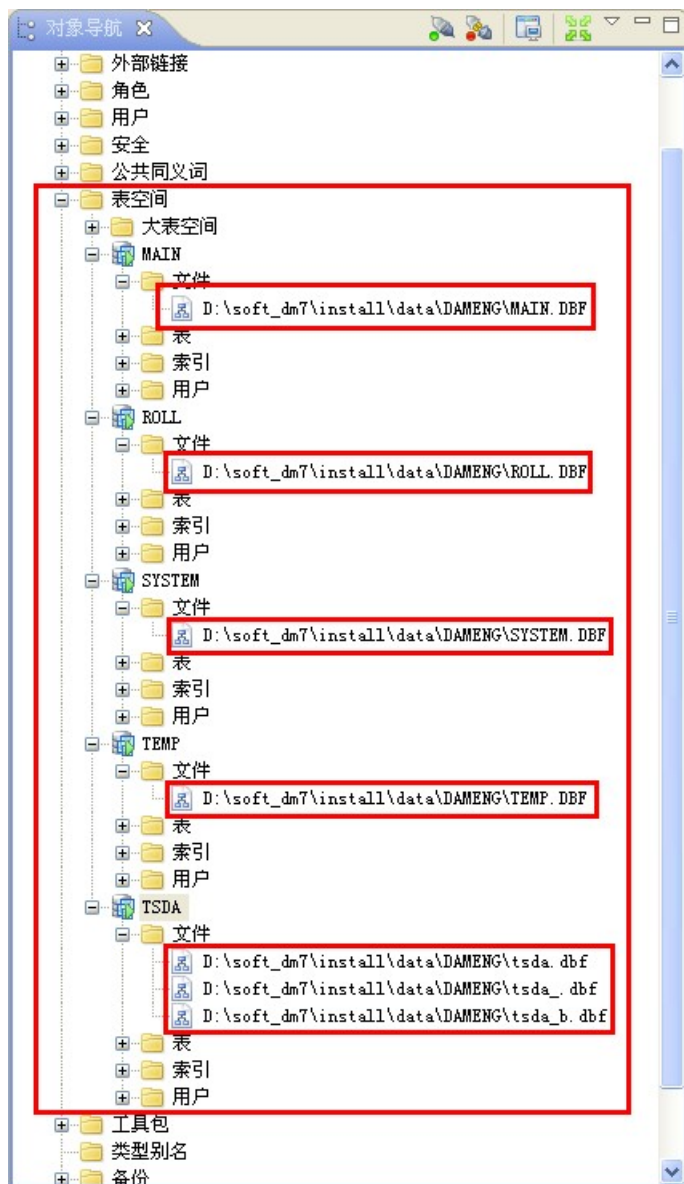


图 1.2 一个完整 DM 数据库的表空间和数据文件分布

## 1.2.2 重做日志

重做日志，又叫 REDO 日志，忠实记录了所有物理页的修改，基本信息包括操作类型、表空间号、文件号、页号、页内偏移、实际数据等。数据库中 INSERT、DELETE、UPDATE 等 DML 操作以及 CREATE TABLE 等 DDL 操作最终都会转化为对某些数据文件、某些数据页的修改。因此，在系统故障重启时，通过重做 REDO 日志，可以将数据库恢复到故障时的

状态。

DM 数据库默认包含两个扩展名为 log 的日志文件，用来保存 REDO 日志，称为联机重做日志文件。这两个文件循环使用。任何数据页从内存缓冲区写入磁盘之前，必须保证其对应的 REDO 日志已经写入到联机日志文件。

Redo 日志包 (RLOG\_PKG) 是 DM 数据库保存 Redo 日志的数据单元，一个日志包内可保存一个或多个 PTX 产生的 Redo 日志。日志包具有自描述的特性，日志包大小不固定，采用固定包头和可变包头结合的方式，包头记录日志的控制信息，包括类型、长度、包序号、LSN 信息、产生日志的节点号、加密压缩信息、日志并行数等内容。

日志包生成时按照序号连续递增，相邻日志包的 LSN 顺序是总体递增的，但是在 DMDSC 集群环境下不一定连续。如果未开启并行日志，RLOG\_PKG 包内日志的 LSN 是递增的。如果开启并行日志，一个 RLOG\_PKG 包内包含多路并行产生的日志，每一路并行日志的 LSN 是递增的，但是各路之间的 LSN 并不能保证 LSN 有序，因此并行日志包内 LSN 具有局部有序，整体无序的特点。

### 1.2.3 归档日志

DM 数据库可以在归档和非归档两种模式下运行，归档模式的具体配置参考 3.1.2 [归档配置](#)。DM 支持多种归档方式。系统在归档模式下运行会更安全，当出现介质故障，如磁盘损坏导致数据文件丢失、异常时，利用归档日志，系统可以恢复至故障发生的前一刻。因此，为了保证归档日志文件和数据文件不同时出现问题，建议将归档目录与数据文件配置、保存到不同的物理磁盘上。除了表备份还原，其他的联机备份与还原必须运行在归档模式下。

DM 数据库定义了多种归档方式，包括本地归档、实时归档、即时归档、异步归档和远程归档，其中本地归档和远程归档与备份还原密切相关。

系统将 REDO 日志先写入联机日志文件后，根据归档的配置情况，异步地将 REDO 日志写入本地归档日志文件，或者通过 MAL 系统发送到远程归档的目标实例，写入目标实例的远程归档日志文件中。

Primary/Normal 模式库归档日志文件的命名规则

是：ARCH\_NAME\_DB\_MAGIC[SEQNO]\_日期时间.log。其中 ARCH\_NAME 是在 dmarch.ini 中配置的 LOCAL/REMOTE 归档名称，DB\_MAGIC 是生成日志的数据库魔数，SEQNO 代表 DSC 节点号，日期时间是归档日志文件的创建时间。比

如：ARCHIVE\_LOCAL1\_0x567891[0]\_2018-05-30\_10-35-34.log。

Standby 模式库（备库）归档日志文件的命名规则

是：STANDBY\_ARCHIVE\_DB\_MAGIC[SEQNO]\_日期时间.log。其中，

STANDBY\_ARCHIVE 表示备库生成的归档日志文件；DB\_MAGIC 是生成日志的数据库（主库）魔数；SEQNO 代表主库对应的 DSC 节点号；日期时间是归档日志文件的创建时间。比

如：STANDBY\_ARCHIVE\_0x123456[0]\_2018-05-30\_10-35-34.log。由于

STANDBY\_ARCHIVE 用于表示备库生成的归档日志，因此，不允许将归档名称配置为

STANDBY\_ARCHIVE。

### 1.2.4 LSN 介绍

LSN (Log Sequence Number) 是由系统自动维护的 Bigint 类型数值，具有自动递增、全局唯一特性，每一个 LSN 值代表着 DM 系统内部产生的一个物理事务。物理事务 (Physical Transaction, 简称 ptx) 是数据库内部一系列修改物理数据页操作的集合，与数据库管理系统中事务 (Transaction) 概念相对应，具有原子性、有序性、无法撤销等特性。

DM 数据库中与 LSN 相关的信息，可以通过查询 V\$RLOG 表来获取。DM 主要包括以下几种类型的 LSN：

- **CUR\_LSN** 是系统已经分配的最大 LSN 值。物理事务提交时，系统会为其分配一个唯一的 LSN 值，大小等于 CUR\_LSN + 1，然后再修改 CUR\_LSN=CUR\_LSN+1。

- **FILE\_LSN** 是已经写入联机 Redo 日志文件的最大 LSN 值。每次将 Redo 日志包 RLOG\_PKG 写入联机 Redo 日志文件后，都要修改 FILE\_LSN 值。

- **FLUSH\_LSN** 是已经发起日志刷盘请求，但还没有真正写入联机 Redo 日志文件的最大 LSN 值。

- **CKPT\_LSN** 是检查点 LSN，所有 LSN <= CKPT\_LSN 的物理事务修改的数据页，都已经从 Buffer 缓冲区写入磁盘，CKPT\_LSN 由检查点线程负责调整。

数据库故障重启时，CKPT\_LSN 之前的 REDO 日志不需要重做，只需要从 CKPT\_LSN+1 开始重做 REDO 日志，就可以将系统恢复到故障前状态。并且，在联机重做日志文件中，LSN 值 <= CKPT\_LSN 的 REDO 日志都可以被覆盖。

- **APPLY\_LSN** 是数据库还原后已经重演日志的最大 LSN。DSC 集群的每一个节点

独立维护 APPLY\_LSN。

### 1.2.5 包序号介绍

每个 RLOG\_PKG 都有对应的序号属性，称之为包序号 (PKG\_SEQNO)，日志包生成时按照序号连续递增。包序号包括本地包序号 (LSEQ) 和全局包序号 (GSEQ)，本地包序号是节点内唯一、连续递增的值，用于校验联机日志连续性；全局包序号由数据守护集群的主备库共同维护，具有全局唯一、连续、递增的特性，用于校验归档日志的连续性。

DM 数据库中与全局包序号相关的信息，可以通过查询 V\$RLOG 表来获取。数据库端主要包括以下几种类型的全局包序号：

- **CUR\_SEQ** 是系统已经分配的最大全局包序号。RLOG\_PKG 写入联机日志文件前，系统会为其分配一个唯一的全局包序号。
- **FILE\_SEQ** 是已经写入联机 Redo 日志文件的最大全局包序号。每次将 Redo 日志包 RLOG\_PKG 写入联机 Redo 日志文件后，都要修改 FILE\_SEQ 值。
- **APPLY\_SEQ** 是数据库还原后已经重演日志的最大全局包序号。DSC 集群的每一个节点独立维护 APPLY\_SEQ。

### 1.2.6 检查点

DM 数据库运行过程中，用户的所有操作都在内存中进行。每修改一条记录都必须先把记录所在的数据页加载到 BUFFER 缓冲区中，然后进行修改。事务运行时，会把生成的 REDO 日志保留在 Redo 日志包 RLOG\_PKG 中，每条日志记录对应一个 LSN，当事务提交或 Redo 日志包满或执行检查点时会进行日志刷盘。

检查点 (checkpoint) 是一个数据库事件，它的功能是按照数据页的修改顺序，依次将 BUFFER 缓冲区中的脏页写入磁盘，并在这个过程中动态调整 CKPT\_LSN 值，释放日志空间。

DM 的检查点分为两种：完全检查点和部分检查点：

- ✓ **完全检查点**：会将内存缓冲区中的所有脏页写入磁盘，并调整 CKPT\_LSN，在数据库正常关闭时会产生一个完全检查点。
- ✓ **部分检查点**：根据 dm.ini 配置文件中的参数 CKPT\_FLUSH\_RATE 和 CKPT\_FLUSH\_PAGES，确定每次检查点刷脏页的数量。执行部分检查点的过程中，DDL/DML

操作都可以正常执行，DM 系统中绝大多数情况下触发的都是部分检查点。

数据库运行过程中产生的待写入日志首先写入 Redo 日志包 RLOG\_PKG，当日志刷盘时一起写入联机日志文件中。在联机日志文件中，可以覆盖写入 REDO 日志的文件长度为可用日志空间；不能被覆盖的 REDO 日志，系统故障重启需要重做的 REDO 日志为有效日志，有效日志的 LSN 取值范围是 (CKPT\_LSN, FILE\_LSN]。

下图说明了联机日志文件、Redo 日志包 RLOG\_PKG 以及相关各 LSN 之间的关系。

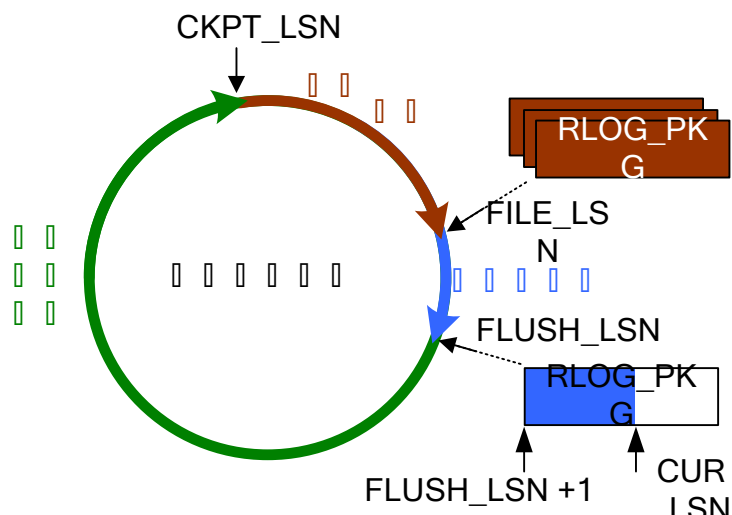


图 1.3 联机日志文件与 Redo 日志包

## 1.2.7 备份集

备份集用来存放备份过程中产生的备份数据及备份信息。一个备份集对应了一次完整的备份。一般情况下，一个备份集就是一个目录，备份集包含一个或多个备份片文件，以及一个备份元数据文件。并行备份和非并行备份情况下，备份集里的内容略有不同。

### 1.2.7.1 备份片

备份片用来存储备份数据的文件。备份时，目标数据文件内容或归档日志内容经过处理后，都会存放到各自的备份片文件中。备份片文件后缀为 .bak，用来存放备份数据，备份集中存放数据页的备份片称为数据备份片，存放 REDO 日志的备份片称为日志备份片。

备份片的大小可以在备份时通过 MAXPIECESIZE 指定，一个备份集中可能生成多个备份片。

### 1.2.7.2 元数据

元数据文件用来存放备份信息，元数据文件的后缀为`.meta`。通过元数据文件，可以了解整个备份集信息。元数据文件中包含的备份信息包括：

- ✓ 备份集本身相关的信息，如是否联机备份，备份的范围，备份的加密信息，以及备份的压缩信息等；
- ✓ 备份源库的建库参数信息，如 DSC 的节点数，是否大小写敏感，`PAGE_CHECK` 属性等；
- ✓ 数据文件信息，如备份了哪些数据文件，文件大小，以及文件相关的表空间信息等；
- ✓ 备份片的信息，如包含哪些备份片文件、备份片大小等信息；
- ✓ 备份库的 `dm.ini` 参数信息和密钥文件（`dm_service.private` 或者 `dm_external.config`，若指定 `usbkey` 加密，则不备份）。

### 1.2.7.3 备份集搜索目录

备份集搜索目录，用于搜集目标备份集的文件路径。备份集搜索目录包括 4 类：

- ✓ 数据库的默认备份目录；
- ✓ `WITH BACKUPDIR` 子句指定的目录；
- ✓ 还原时备份集所在的上级目录；
- ✓ 增量备份时基备份集所在的上级目录。

如果使用第三方备份（介质为 `TAPE` 类型），则只搜索 `WITH BACKUPDIR` 子句指定的备份集目录，具体搜集方式由第三方备份程序决定。

`WITH BACKUPDIR` 子句指定的目录列表，包含非法目录或目录不存在时，不会报错，只在日志中打印警告。

## 1.2.8 备份

备份的目的是当数据库遇到损坏的情况下，可以执行还原恢复操作，把数据库复原到损坏前的某个时间点。用于还原恢复数据库的载体是备份集，生成备份集的过程便是备份了。

备份就是从源库（备份库）中读取有效数据页、归档日志等相关信息，经过加密、压缩等处理后写入备份片，并将相关备份信息写入备份元数据文件的过程。

下面将根据数据组织形式、数据库状态、备份粒度、备份内容、以及备份的一致性和完整性等差异，对备份进行介绍。

### 1.2.8.1 逻辑备份和物理备份

逻辑备份是指利用 `dexp` 导出工具，将指定对象（库级、模式级、表级）的数据导出到文件的备份方式。逻辑备份针对的是数据内容，并不关心这些数据物理存储在什么位置。

物理备份则直接扫描数据库文件，找出那些已经分配、使用的数据页，拷贝并保存到备份集中。物理备份过程中，不关心数据页的具体内容是什么，也不关心数据页属于哪一张表，只是简单的根据数据库文件系统的描述，来挑选有效的数据页。

这两种备份方式，分别适应不同的应用场景，本书重点介绍物理备份，关于逻辑备份更详细的说明，可参考《DM8\_dexp 和 dimp 使用手册》。

### 1.2.8.2 联机备份和脱机备份

数据库处于运行状态、并正常提供数据库服务情况下进行的备份操作，我们称为联机备份。数据库处于关闭状态时进行的备份操作，被称为脱机备份。

从 V2.0 版本开始，使用 `DMRMAN` 工具进行脱机备份，并且支持对异常关闭的数据库进行脱机库备份。备份异常关闭的数据库，要求配置了本地归档，如果本地归档不完整，则需要先修复本地归档，再进行备份。

联机备份则使用客户端工具连接到数据库实例后，通过执行 `SQL` 语句进行；也可以通过配置作业，定时完成自动备份。联机备份不影响数据库正常提供服务，是最常用的备份手段之一。

联机备份时，可能存在一些处于活动状态的事务正在执行，为确保备份数据的一致性，需要将备份期间产生的 `REDO` 日志一起备份。因此，只能在配置本地归档、并开启本地归档的数据库上执行联机备份。



只有已经关闭的数据库才允许执行脱机备份。正在运行的数据库，无法执行

**注意：**脱机备份，系统会报错。

### 1.2.8.3 数据备份和归档日志备份

按照备份内容不同，可以分为数据备份和归档日志备份。数据备份主要针对数据文件内容，包括库备份、表空间备份和表备份。

库备份，顾名思义就是对整个数据库执行的备份，又称为库级备份。库备份会拷贝数据库中所有数据文件的有效数据页，如果是联机备份，则还会拷贝备份过程中产生的归档日志，写入到备份集中。

表空间备份是针对特定表空间执行的备份，又称为表空间级备份。表空间备份只能在联机状态下执行。

表备份则拷贝指定表的所有数据页到备份集中，并会记录各个数据页之间的逻辑关系用以恢复。表备份只能在联机状态下执行，一次表备份操作只能备份一张用户表，并且不支持增量表备份。

归档日志备份，是专门针对归档日志文件进行操作，不涉及任何数据文件内容。归档日志备份扫描归档目录收集归档日志文件，并将归档日志写入到备份集中。既可以在数据库运行状态下，执行联机归档日志备份；也可以在数据库关闭状态下执行脱机归档日志备份。

### 1.2.8.4 日志备份范围

联机备份过程中，用户可以正常访问、修改数据库，为了准确记录备份过程中产生了哪些 REDO 日志，确定日志备份范围，我们特别定义了下述几个包序号和 LSN：

- 节点 BEGIN\_LSN

为了保证备份的完整性和有效性，必须包含的归档日志起始 LSN 值。BEGIN\_LSN = 备份开始时检查点偏移前一个 RLOG\_PKG 的 max\_lsn。

- 节点 BEGIN\_SEQ

BEGIN\_SEQ 记录了 BEGIN\_LSN 所在 Redo 日志包的序号。

- 节点 END\_LSN

为了保证备份的完整性和有效性，必须包含归档日志结束 LSN 值。END\_LSN = 备份结束时 FILE\_LSN。

- 节点 END\_SEQ

BEGIN\_SEQ 记录了 END\_LSN 所在 Redo 日志包的序号。



### ● BAK\_END\_LSN

备份结束时，可以保证事务一致性的全局备份结束 LSN。单节点 BAK\_END\_LSN 等于 END\_LSN；DSC 集群环境中，每个节点的 END\_LSN 都不相同，BAK\_END\_LSN 大于等于最大的 END\_LSN。

如果 BEGIN\_SEQ 等于 END\_SEQ，则表明备份过程中，该节点没有任何数据被修改。为了简化还原过程，增量备份时要求 BEGIN\_LSN 必须大于等于基准备份的 END\_LSN，如果不满足条件，则强制生成检查点，直到 BEGIN\_LSN 满足条件为止。

### 1.2.8.5 一致性备份和非一致性备份

按照备份集中的数据是否满足一致性，可以将备份划分为一致性备份和非一致性备份。

一致性备份的备份集包含了完整的数据文件内容和归档日志信息；利用一个单独的备份集可以将数据库恢复到备份时状态。不指定 WITHOUT LOG 选项的联机备份生成的备份集就是一致性备份。脱机数据库备份会强制将检查点之后的有效 REDO 日志拷贝到备份集中，因此，脱机备份一定是一致性备份。数据库正常关闭时，会生成完全检查点，脱机备份生成的备份集中，不包含任何 REDO 日志。

非一致性备份的备份集只包含数据文件相关内容，没有归档日志信息，利用非一致性备份还原的数据库，无法直接启动，必须借助归档日志来恢复。表空间备份、指定 WITHOUT LOG 选项的联机备份生成的备份集都是非一致性备份集。

### 1.2.8.6 完全备份和增量备份

按照备份数据完整性，可将备份分为完全备份和增量备份。库备份和表空间备份支持增量备份，表备份不支持增量备份。

完全备份生成的备份集包含了指定库（或者表空间）的全部有效数据页。当数据规模比较大的情况下，生成的完全备份集通常会比较大，而且备份时间也会比较长。

增量备份是在某个特定备份集基础上，收集数据库新修改的数据页进行备份，可以有效减少备份集的空间占用、提高备份速度。这个特定的、已经存在的备份集称为增量备份的基备份，根据对基备份的要求不同，DM 的增量备份分为以下两种：

#### 1. 差异增量备份

差异增量备份的基备份既可以是一个完全备份集，也可以是一个增量备份集。

利用增量备份进行还原操作时，要求其基备份必须是完整的；如果差异增量备份的基备份本身也是一个增量备份，那么同样要求其基备份是完整的；任何一个增量备份，最终都是以一个完全备份作为其基备份。因此，完全备份是增量备份的基础。

### 2. 累积增量备份

累积增量备份的基备份只能是完全备份集，而不能是增量备份集。

增量备份的基备份集既可以是脱机备份生成的，也可以是联机备份生成的，脱机增量备份的基备份集可以是联机备份生成的，联机增量备份的基备份集也可以是脱机备份生成的。

## 1.2.9 还原与恢复

还原是备份的逆过程，就是从备份集中读取数据页，并将数据页写入到目标数据库对应数据文件相应位置的过程。

由于联机备份时，系统中可能存在一些处于活动状态的事务正在执行，并不能保证备份集中的所有数据页是处于一致性状态；而脱机备份时，数据页不一定是正常关闭的，也不能保证备份集中所有数据页是处于一致性状态。因此，还原结束后目标库有可能处于非一致性状态，不能马上提供数据库服务；必须要进行数据库恢复操作后，才能正常启动。

与备份类似，下面将从数据组织形式、数据库状态、还原粒度、还原内容等方面介绍还原的相关内容。

### 1.2.9.1 逻辑还原和物理还原

逻辑还原是逻辑备份的逆过程，逻辑还原就是使用 `dimp` 工具，把 `dexp` 导出的备份数据重新导入到目标数据库。

物理还原是物理备份的逆过程，物理还原一般通过 `DMRMAN` 工具（或者 `SQL` 语句），把备份集中的数据内容（数据文件、数据页、归档文件）重新拷贝、写入目标文件。

关于逻辑还原更详细的说明，可参考《DM8\_dexp 和 dimp 使用手册》。本文档主要介绍物理备份、还原相关内容。

### 1.2.9.2 联机还原和脱机还原

联机还原指数据库处于运行状态时，通过 SQL 语句执行还原操作。表还原可以在联机状态下执行。

脱机还原指数据库处于关闭状态时执行的还原操作，脱机还原通过 DMRMAN 工具进行。库备份、表空间备份和归档备份，可以执行脱机还原。脱机还原操作的目标库必须处于关闭状态。

### 1.2.9.3 数据还原和归档日志还原

根据备份集类型，数据还原可以分为库还原、表空间还原和表还原。库还原和表空间必须脱机执行；表还原操作只能联机执行。

表空间还原的数据来源既可以是表空间备份集，也可以是库备份集。还原的目标表空间不能是 TEMP 表空间，只能是 MAIN、SYSTEM、ROLL 表空间，或者用户定义的表空间。

表还原从表备份集读取数据，重新恢复目标表数据，还会在目标表上重建索引、约束。

归档日志还原则将归档日志备份集中的归档日志内容，重新生成到指定目录中。

### 1.2.9.4 完全还原和增量还原

完全还原是指直接利用完全备份集进行数据还原操作。增量还原指通过增量备份集进行数据还原操作。但是考虑到增量备份集的基础一定是一个完全备份集，因此增量还原过程中隐含了一个完全还原操作。如果增量备份集的基备份集被删除了，那么单独使用这个增量备份集是无法进行还原操作的。

## 1.3 介质管理层

DM 通过介质管理层 MML，将备份、还原和恢复过程中备份片文件和元数据文件的读取写入动作进行抽象、隔离，备份还原过程中 MML 层会调用一组 SBT 接口来访问备份存储介质。SBT 接口是 DM 定义的一组 API 函数，允许第三方厂商定制、管理备份存储介质；DM 系统自带的备份还原功能也遵循了 SBT 规范。

如果备份、还原时指定 DEVICE TYPE 为 TAPE，且使用第三方提供的

dmsbtex.dll(dmsbtex.so)来完成备份文件的存取功能。DEVICE TYPE 指定 TAPE 时还提供了 PARS 关键字，用于传递第三方可识别的自定义字符串。使用第三方提供的动态库时，用户层应用不受影响。默认 DEVICE TYPE 为 DISK，使用达梦自带 SBT 磁盘存储。

## 1.4 备份方式

当前仅支持备份集方式的备份还原，不再支持其他备份方式。备份还原实现策略有两种：dmap 辅助进程方式和无辅助进程方式。用户可通过 DM.INI 参数 bak\_use\_ap 来选择（dmrman 使用参数 use\_ap），bak\_use\_ap 取值 1、2。默认为 1。

1: DMAP 辅助进程方式，可支持第三方备份（指定 DEVICE TYPE 为 TAPE）。DMAP 插件执行，改造了备份还原任务子系统，允许指定并行度，大幅提升了备份还原的效率，特别是加密、压缩的处理效率。如果选择使用 DMAP 辅助进程，执行备份还原之前就必须启动 DMAP 服务。安装 DM 数据库以后，DMAP 服务会自动启动。如果需要手动启动，有两种途径，一是启动 DM 服务查看器中的 DmAPService。二是通过手动启动 DMAP 执行码实现，DMAP 执行码位于 DM 安装目录的 bin 子目录下。除此之外，Linux 下，还可以调用 bin 目录下的 DmAPService 脚本。

2: 无辅助进程方式，不依赖 DMAP，由主进程 dmserver 自身执行备份还原，但不支持第三方备份（指定 DEVICE TYPE 为 TAPE）。

DMAP 的使用说明，请参考《DM8 SQL 语言使用手册》。



说明：

DMSERVER 和 DMRMAN 使用 DMAP，都是通过管道通讯实现。其中 DMRMAN 产生的管道命名规则为：DM\_PIPE\_DMRMAN-PID\_SEQ，其中：PID 为 DMRMAN 实例进程 ID，SEQ 为同一实例上管道的序号（从 1 开始）。在 WIN 环境中，垃圾管道文件会被自动清理掉。在非 WIN 环境中，如果系统没有正常退出，就会遗留垃圾文件，需要用户手动删除。删除垃圾管道文件时，需要根据 PID 值来判断，文件名中的 ID 不是当前正在运行的 DMRMAN 的进程 ID 的即为垃圾文件，都可以删除。

## 2 备份还原原理

本章介绍 DM 备份与还原的实现原理，读者通过阅读本章内容，可以了解 DM 备份、还

原与恢复的实现原理。

## 2.1 归档说明

备份与恢复过程都依赖归档日志，归档日志是保证数据一致性和完整性的重要保障。配有归档日志的数据库系统在出现故障时丢失数据的可能性更小，这是因为一旦出现介质故障如磁盘损坏时，利用归档日志，系统可被恢复至故障发生的前一刻，也可以还原到指定的时间点。

### 2.1.1 本地归档

Redo 日志本地归档（LOCAL），就是将 Redo 日志写入到本地归档日志文件的过程。配置本地归档情况下，Redo 日志刷盘线程将 Redo 日志写入联机 Redo 日志文件后，对应的 RLOG\_PKG 由专门的归档线程负责写入本地归档日志文件中。

与联机 Redo 日志文件可以被覆盖重用不同，本地归档日志文件不能被覆盖，写入其中的 Redo 日志信息会一直保留，直到用户主动删除；如果配置了归档日志空间上限，系统会自动删除最早生成的归档 Redo 日志文件，腾出空间。如果磁盘空间不足，且没有配置归档日志空间上限（或者配置的上限超过实际空间），系统将自动挂起，直到用户主动释放出足够的空间后继续运行。

DM 提供了按指定的时间或指定的 LSN 删除归档日志的系统函数（SF\_ARCHIVELOG\_DELETE\_BEFORE\_TIME 和 SF\_ARCHIVELOG\_DELETE\_BEFORE\_LSN），但需谨慎使用。避免归档日志缺失，导致数据无法恢复。



注意：

为了最大限度地保护数据，当磁盘空间不足导致归档写入失败时，系统会挂起等待，直到用户释放出足够的磁盘空间。  
当磁盘损坏导致归档日志写入失败时，系统会强制 HALT。

### 2.1.2 远程归档

所谓远程归档（REMOTE ARCHIVE），顾名思义就是将写入本地归档的 REDO 日志信息，发送到远程节点，并写入远程节点的指定归档目录中。DMDSC 集群中各个节点在配置本地

归档之外，再相互配置一个远程归档，就可以在任意一个节点的本地磁盘中，找到 DMDSC 集群所有节点产生的、完整的归档日志文件。远程归档的触发时机是，在 REDO 日志写入本地归档日志文件的同时，将 REDO 日志通过 MAL 系统发送给指定的数据库实例。

远程归档与本地归档的主要区别是 REDO 日志写入的位置不同，本地归档将 REDO 日志写入数据库实例所在节点的磁盘，而远程归档则将 REDO 日志写入到其他数据库实例所在节点的指定归档目录。

远程归档与本地归档的另外一个区别就是归档失败的处理策略不同：本地归档写入失败（比如磁盘空间不足），系统将会挂起；而远程归档失败则会直接将远程归档失效，不再发送 REDO 日志到指定数据库实例。当节点间网络恢复、或者远程节点重启成功后，系统会自动检测并恢复远程归档，继续发送新写入的 REDO 日志，但不会主动补齐故障期间的 REDO 日志。因此，在出现节点故障等情况下，远程归档的内容有可能是不完整的，而本地归档的内容肯定是完整的；如果备份还原恰好需要用到这段丢失的远程归档日志，那么可以从源端的本地归档拷贝、补齐这部分内容。



**远程归档必须双向配置，单向配置时目标实例上不会接收归档日志，归档状态将会变成无效状态。**

### 2.1.3 归档切换

由于本地归档和远程归档是异步写入归档日志文件的，REDO 日志在写入联机日志文件后，再由专门的归档线程负责将这些 REDO 日志写入本地归档日志文件。通过归档切换功能，可以将这些已经写入联机日志文件，但还没有写入归档日志文件的 REDO 日志，写入到归档日志文件中。通过执行以下 SQL 命令，可以完成归档切换功能。三条语句功能一样，选择一条执行即可。

```
alter database archivelog current;  
alter system archive log current;  
alter system switch logfile;
```

### 2.1.4 归档修复

DM 数据库实例正常退出时，会将所有 REDO 日志写入本地归档日志文件中；但是数据

库实例异常关闭时，可能存在部分 REDO 日志未写入本地归档日志文件中，归档日志文件中的内容比实际可恢复的数据少一部分。这种情况下，将无法利用归档日志文件将数据恢复到最新状态，需要从联机日志文件拷贝该部分日志以补齐归档日志。

本地归档修复会扫描联机日志文件，将那些已经写入联机日志文件、但还没有写入到归档日志文件的 REDO 日志，重新写入到归档日志文件，流程如下：

1. 收集本地归档日志文件；
2. 扫描归档文件，获取最后一个有效 RLOG\_PKG 偏移
3. 首先，根据偏移来截取最后一个本地归档日志文件中有效内容，删除掉 RLOG\_PKG 偏移之后的多余内容，保留 RLOG\_PKG 偏移之前的内容，并调整日志文件头信息。然后，再创建一个新的空的归档日志文件；
4. 扫描联机日志文件，拷贝缺失的 REDO 日志并写入新创建的归档日志文件。

## 2.2 备份

任何一个对 DM 数据库的操作，归根结底都是对某个数据文件页的读写操作。物理备份就是把这些数据文件中的有效数据页备份起来，在出现故障时，用于恢复数据。DM 的物理备份一般包括数据备份和日志备份两部分，数据备份是拷贝数据页内容，日志备份则是拷贝备份过程中产生的 REDO 日志。

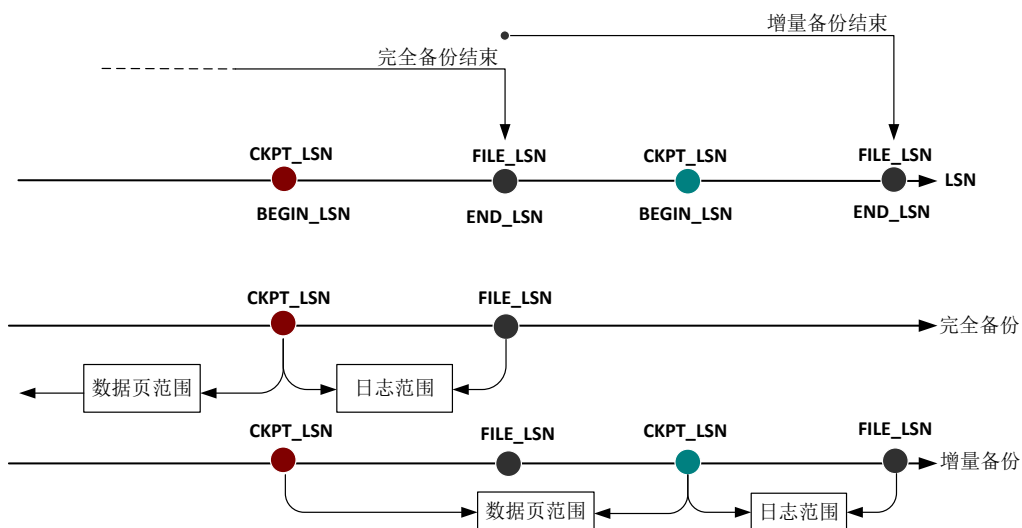


图 2.1 备份内容

## 2.2.1 数据备份

数据备份过程中，根据 DM 数据文件系统的描述信息，准确判断每一个数据页是否被分配、使用，将未使用的数据页剔除，仅保留有效数据页进行备份，这个过程我们称为智能抽取。与直接文件拷贝方式相比，DM 物理备份丢弃了那些没有使用的数据页，因此可以节省对存储空间的要求，并有效减少 IO 数量，提升备份、还原的效率。

对处于 RES\_OFFLINE 和 CORRUPT 状态的表空间，则只记录表空间相关信息和状态，不会真正拷贝数据页。数据备份过程中，会对数据进行校验，如果校验失败则会将相关信息写入到日志文件 dm\_BAKRES\_xxx.log 中，但不会终止当前备份操作。



使用 dminit 建库时，通过设置 INI 参数 `page_check`，指定页校验模式，

说明：当指定值不为 0 时，在执行备份过程中会对数据页执行校验操作，校验结果会记录在备份集中，并在备份结束后给出警告信息，警告码为 609，告知用户此备份集中存在被破坏的数据页。该警告只是提示作用，并不影响备份集的有效性。在还原库上，备份集中备份的被破坏数据页在经过还原和恢复操作后，可以正常使用。

dminit 建库时，通过设置参数 `page_check`，指定页校验模式，当指定值不为 0 时，在执行备份过程中会对数据页执行校验操作，校验结果会记录在备份集中，并在备份结束后给出警告信息，警告码为 609，告知用户此备份集中存在被破坏的数据页。该警告只是提示作用，并不影响备份集的有效性，备份集中备份的被破坏数据页在经过备份后的还原和恢复操作后，可以正常使用。

### ■ 完全备份

执行完全备份，备份程序会扫描数据文件，拷贝所有被分配、使用的数据页，写入到备份片文件中（如图 2.1 中第二根横轴所示）。库备份会扫描整个数据库的所有数据文件，表空间备份则只扫描表空间内的数据文件。

参照图 1.2，若执行库备份，则备份数据文件包括除 TEMP 表空间以外的其他表空间内的所有数据文件；若执行表空间备份，以用户表空间 TSDA 为例，那么只会扫描 TSDA 表空间的数据文件，包括 `tsda.dbf`、`tsda_.dbf` 和 `tsda_b.dbf`。数据文件备份结束后，BEGIN\_LSN 之前修改的所有数据页都被备份下来。



### ■ 增量备份

执行增量备份，备份程序会扫描数据文件，拷贝所有基备份结束以后被修改的数据页，写入到备份片文件中。

为了简化增量备份的还原过程，避免还原过程中重做基备份集对应的归档日志，DM 要求执行增量备份时， $\leq$ 基备份 `END_LSN` 的所有数据页已经写入磁盘。也就是说增量备份会拷贝 `LSN >` 基备份 `BEGIN_LSN` 的数据页写入备份片文件中(如图 2.1 中第三根横轴所示)，`LSN  $\leq$`  基备份 `BEGIN_LSN` 的数据页不需要写入到备份片文件中。

同样库增量备份会扫描整个数据库的所有数据文件，表空间增量备份只扫描表空间内的数据文件。

### ■ 表空间备份

表空间备份只拷贝指定表空间的数据页，因此，相对于数据库备份而言，表空间备份的速度会更快，生成的备份集会更小。对一些包含关键数据的用户表空间，我们可以使用表空间备份功能，进一步保障数据安全。

表空间备份支持完全备份和增量备份，但只能在联机状态下执行。不支持 `temp` 表空间备份还原。

### ■ 表备份

表备份主要包括数据备份和元信息备份两部分。与库备份和表空间备份不同，表备份不是直接扫描数据文件，而是从 `BUFFER` 中加载数据页，拷贝到备份片文件中。表备份的元信息则包括建表语句、重建约束语句、重建索引语句，以及其他相关属性信息。表备份不需要配置归档就可以执行，并且不支持增量表备份。

## 2.2.2 日志备份

所谓日志备份，就是将备份过程中产生的 `REDO` 日志拷贝到备份片文件中，用来在数据库还原结束后，将数据库恢复到一致性状态。如图 2.1 所示，在执行备份过程中，用户可能修改数据库中的数据，并产生对应的 `REDO` 日志。增量备份和完全备份的日志备份流程完全相同。

在备份开始时，记录一个 `BEGIN`，在备份结束后记录一个 `END_LSN`，那么 `[BEGIN,`

END\_LSN]之间的 REDO 日志，就对应备份过程中用户对数据的修改。其中 BEGIN\_LSN = CKPT\_LSN，作为日志备份的起点，END\_LSN = 数据备份结束时的 FILE\_LSN，作为日志备份的终点。

联机库备份默认开启日志备份，将备份过程中产生的 REDO 日志单独写入备份片作为备份集的一部分。也可以通过 WITHOUT LOG 子句取消这部分 REDO 日志的拷贝，这种情况下生成的备份本身是不完整的，数据库还原后，还要依赖备份库的本地归档日志来进行恢复操作后，才能将数据恢复到一致性状态；否则，还原后的数据库将无法启动。

还原后目标库中 redo 日志的数量，由目标库本身 redo 日志的数量决定。跟源库中 redo 日志的数量无关。例如，源库中有 3 个 redo 日志，目标库中只有 2 个，将源库还原到目标库后，目标库仍然只有 2 个 redo 日志。

### 2.2.3 压缩与加密

DM 支持对备份数据进行压缩和加密处理，用户在执行备份时，可以指定不同的压缩级别，以获得不同的数据压缩比。默认情况下，备份是不进行压缩和加密处理的，详细使用方法参考 [3 DM8 备份与还原实战](#) 中的相关介绍。

DM 共支持 9 个级别（1~9 级）的压缩处理，级别越高压缩比越高，但相应的压缩速度越慢、CPU 开销越大。

备份加密包括加密密码、加密类型和加密算法三个要素。加密密码通过使用 IDENTIFIED BY<加密密码>来指定，使用备份集的时候必须输入对应密码。加密类型分为不加密、简单加密和完全加密。简单加密仅仅对部分数据进行加密，加密速度快；完全加密对所有数据执行加密，安全系数高。加密算法可参考《DM8 安全管理》加密章节。

加密类型和算法，用户均可手动指定。如果用户指定了加密密码，但没有指定加密类型和算法，则使用默认加密算法进行简单加密。用户也可以指定加密密码，但将加密类型指定为不加密。

如果同时指定加密和压缩，则备份过程中，会先进行压缩处理，再进行加密处理，备份的所有数据页和 REDO 日志都会进行压缩、加密处理。



注意：

如果基备份集没有指定加密类型，那么增量备份也不能指定加密类型；

如果基备份集指定了加密算法，那么增量备份的加密类型、加密算法和密码必须与基备份保持一致。

## 2.2.4 并行备份

库备份、表空间备份以及归档日志备份可以进行并行处理，用户通过关键字 `PARALLEL` 指定是否执行并行备份，以及并行备份的并行数。并行备份以数据文件为单位，也就是说一个数据文件仅可能出现在一个并行分支中。如果指定了 `PARALLEL` 关键字，但不指定并行数，那么默认的并行数为 4，但实际的备份并行度由 `DMAP` 最终创建成功的并行子任务数决定。增量备份是否并行以及并行数与其基备份集无关。

目前的数据库并行备份还原都是以文件为单位，适用于待备份文件大小比较均匀的情况。若文件大小差别比较大，特别存在个别文件巨大时，并行备份还原基本没有优势。因此在进行数据库备份时，需要指定 `READ SIZE <拆分块大小>`，将巨大的数据文件先进行拆分之后再行备份。

执行并行备份会生成一个主备份集和若干个子备份集，子备份集不能单独还原，也不能作为其他备份集的基备份。备份过程中产生的 `REDO` 日志保存在主备份集中，子备份集仅包含数据文件相关内容。

以下为一个并行数为 3 的脱机并行备份集（左）和非并行脱机备份集（右）比较截图：

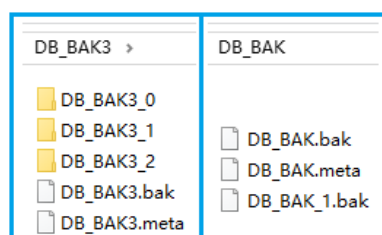


图 2.2 并行备份集目录结构示意图

## 2.3 还原与恢复

还原与恢复是备份的逆过程，还原与恢复的主要目的是将目标数据库恢复到备份结束时刻的状态。还原的主要动作是将数据页从备份集中拷贝回数据库文件相应位置，恢复则是重做 `REDO` 日志将数据库恢复到一致性状态。

## 2.3.1 数据还原

### 2.3.1.1 库还原

库还原就是根据库备份集中记录的文件信息重新创建数据库文件，并将数据页重新拷贝到目标数据库的过程。DM 既可以将一个已存在的数据库作为还原目标库，也可以指定一个路径作为还原目标库的目录。库还原的主要步骤包括：清理目标库环境；重建数据库文件；拷贝数据页；重建联机日志文件；修改配置参数等。

#### ■ 清理目标库环境

如果指定已存在的数据库作为还原目标库，还原操作首先解析 `dm.ini` 配置文件，获取 `dm.ctl` 控制文件路径，删除控制文件中的数据文件，然后根据 `overwrite` 选项，如果指定 `overwrite` 选项，若待还原文件存在，则删除；如果未指定 `overwrite` 选项，若待还原文件存在，则报错，但保留目标库的日志文件、控制文件等。

如果指定还原到一个目录，则根据 `OVERWRITE` 参数选择策略，检查目标目录内的 `dm.ini` 文件、`dm.ctl` 文件，默认的日志文件 `DBNAME01.log` 和 `DBNAME02.log`（其中 `DBNAME` 为数据库名称），待还原的数据文件等。如果用户指定 `OVERWRITE`，并且存在相关文件情况下，还原过程中会自动删除这些已经存在的文件；如果没有指定 `OVERWRITE`，并且存在相关文件，则会报错。

#### ■ 重建数据库文件

如果将一个已存在数据库作为还原目标，则需要将目标数据库的 `dm.ini` 路径作为还原参数。还原过程中，会重新创建数据文件，并将相关信息写入 `dm.ctl` 控制文件中。

如果将数据库还原到指定目录，则会在这个目录创建一个 `dm.ini` 配置文件，设置 `CTL_PATH`、`SYSTEM_PATH` 配置项指向这个目录，并在这个目录下创建 `dm.ctl` 控制文件。DMDSC 不支持指定目录还原数据库。

数据文件重建策略：

1) 目标库和备份集中的 `SYSTEM_PATH` 路径相同，则按照备份集中记录的原始路径创建文件；

2) 目标库和备份集中的 `SYSTEM_PATH` 路径不相同，默认在 `SYSTEM_PATH` 目录创建

文件；

3) 如果已存在同名文件导致文件创建失败，会重命名文件后在 `SYSTEM_PATH` 目录创建，文件重命名规则是：`DB_NAME+序号.DBF`；如果重命名后仍然冲突，则会一直重试；

4) 使用 `mapped file` 指定源文件与目标文件的映射关系，定制数据库文件的物理分布情况，可以很好的满足用户关于数据文件分布的需求。

### ■ 重建联机日志文件

指定目录还原，系统目录使用指定还原目录，所有库配置文件均认为在指定还原目录下。联机日志文件命名规则，单机环境为 `db_name+文件编号.log`，其中 `db_name` 取自备份集备份库的名称，文件编号从 1 开始，如 `DAMENG01.log`、`DAMENG02.log`。联机日志文件至少 2 个。

### ■ 拷贝数据页

拷贝数据页是从备份集中读取数据页，并将数据页写入数据文件指定位置的过程。由于备份过程中，只将有效的数据页写入备份集中，因此，还原过程也只涉及这些被分配使用的数据页。

### ■ 重置目标库

具体包括：

- ✓ 更新日志信息，设置当前 `CKPT_LSN` 为备份集中 `BEGIN_LSN`，并设置日志文件状态为 `INACTIVE`；
- ✓ 更新 `DB_MAGIC`，还原后，库中 `PERMANENT_MAGIC` 仍与备份集中相同；
- ✓ 设置还原标志，标识当前库为指定库要还原的库，不允许使用；
- ✓ 更新目标库的控制文件 `dmctl`，把当前库中的数据文件信息都记录到控制文件中，使用备份集中的服务器密钥文件，重新生成新的密钥文件。

### ■ 修改配置参数

还原到指定库时，默认会保留目标库的配置参数不变，也可以在还原时指定 `REUSE DMINI` 子句，使用备份集中的配置参数替换目标库 `dm.ini` 中的配置参数。还原到指定目录时，会重新建立一个 `dm.ini` 配置文件，并用备份集中的参数值来设置这些配置项。需要注意的是，一般与路径相关的配置参数，比如 `SYSTEM_PATH` 等并不会被替换，而是保

留目标库 `dm.ini` 的原始值不变。

服务器密钥文件(`dm_service.private` 或者 `dm_external.config`)仅在备份集中备份库非 `usbkey` 加密的情况下重建, 并使用备份集中备份的密钥内容进行还原。

注意事项:

- 1) 指定的 `dm.ini` 必须存在且各项配置信息有效, 其中 `CTL_PATH` 必须配置且路径必须有效;
- 2) 若指定目录还原, 则指定目录作为数据库系统目录处理;
- 3) 由于还原是要确保数据库数据的完整性, 因此, 对于增量备份的还原, 需要搜集完整的备份集链表, 然后从前到后, 逐个还原备份集中数据。鉴于增量备份 `BEGIN_LSN` 确定规则, 增量备份的还原过程中, 不需要重做任何归档日志。

### 2.3.1.2 表空间还原

表空间还原是根据库备份集或表空间备份集中记录的数据信息, 重建目标表空间数据文件并拷贝数据页的过程, 该过程不涉及日志操作。

表空间还原只可以在脱机状态下执行。脱机时通过 `DMRMAN` 工具执行, 对表空间状态没有限制。

表空间还原后如果表空间状态为 `RES_OFFLINE`, 表明目标表空间已进行还原操作, 但数据不完整。

在部分数据文件损坏, 或者部分物理磁盘损坏情况下, 我们可以指定还原数据文件, 跳过那些正常的数据文件, 以提升还原速度。

表空间还原也支持使用 `mapped file` 进行数据文件映射, 如果不指定 `mapped file`, 则默认当前系统目录与备份集中一致。实际创建过程中, 若发现已经存在或者创建失败后, 处理方式与数据库还原中数据重建策略处理一致。



表空间状态包括：**ONLINE**（联机状态）、**OFFLINE**（脱机状态）、**RES\_OFFLINE**

**说明：**（还原状态）、**CORRUPT**（损坏状态）。V\$TABLESPACE 表的 STATUS\$列值表示表空间状态，0/1/2/3 分别代表 **ONLINE** 状态/**OFFLINE** 状态/**RES\_OFFLINE** 状态/**CORRUPT** 状态。

表空间发生故障，比如还原失败（处于 **RES\_OFFLINE** 状态）、表空间文件损坏或缺失（处于 **OFFLINE** 状态），这两种故障情况下如果想直接删除表空间，不考虑还原恢复的方式，则可以手动将表空间切换到 **CORRUPT** 状态，再执行删除操作，否则无法删除。已经切换到 **CORRUPT** 状态后，仍然允许再次执行还原恢复，**CORRUPT** 语法可参考《DM8\_SQL 语言使用手册》。

### 2.3.1.3 表还原

表还原是表备份的逆过程，表还原从表备份集中读取数据替换目标表，将目标表还原成备份时刻的状态。表还原主要包括三部分内容：表结构还原、数据还原、以及重建索引和约束。

如果还原目标表不存在，则利用备份集中记录的建表语句重建目标表；如果还原目标表已经存在，则清除表中的数据、删除二级索引和约束；如果备份表存在附加列（通过 **ALTER TABLE** 语句快速增加的列），那么还原目标表必须存在、并且目标表所有列的物理存储格式与备份源表完全一致。

数据还原过程从表备份集拷贝数据页，重构数据页之间的逻辑关系，并重新形成一个完整的表对象。

在数据还原结束后，使用备份集中记录的信息，重新在表上创建二级索引，并建立各种约束。

表还原只支持在联机状态下执行，表还原过程中也不需要重做 REDO 日志。并且，表备份集允许跨库还原，但要求还原目标库与源库的数据页大小等建库参数相同。需要匹配的建库参数参考表 2.1。

表 2.1 还原库与备份库需匹配建库参数

名称	描述
PAGE_SIZE	数据页大小

BLANK_PAD_MODE	空格填充模式，可选值：0/1
CASE_SENSITIVE	字符大小写是否敏感
CHARSET/UNICODE_FLAG	字符集(0)，可选值：0[GB18030]，1[UTF-8]，2[EUC-KR]
USE_NEW_HASH	是否使用新的 HASH 算法
LENGTH_IN_CHAR	VARCHAR 类型长度是否以字符为单位(N)
PAGE_ENC_SLICE_SIZE	数据页加解密分片大小

可以使用 DMRMAN 工具的 SHOW 功能查看备份的数据页大小等建库参数。

## 2.3.2 数据恢复

数据恢复是指在还原执行结束后，重做 REDO 日志，将数据库恢复到一致性状态，并执行更新 DB\_MAGIC 的过程。其中重做 REDO 日志可以多次执行，直到恢复到目标状态。还原结束后，必须经过恢复操作，数据库才允许启动。即使备份过程中没有修改任何数据，备份集不包含任何 REDO 日志，在数据库还原结束后，也必须使用 DMRMAN 工具执行数据恢复操作后，才允许启动数据库。未经过还原的数据库，也允许执行数据恢复。

数据恢复重做的 REDO 日志，既可以是那些在备份过程中产生的、包含在备份集中的 REDO 日志，也可以是备份数据库本地归档日志文件。在本地归档日志完整的情况下，数据还原结束后，可以利用本地归档日志，将数据库恢复到备份结束后任意时间点状态。

不管采用哪种数据恢复方法，REDO 日志的范围至少要覆盖备份过程中产生的 REDO 日志，也就是说必须完整包括备份集中记录的[BEGIN\_LSN, END\_LSN]之间的 REDO 日志，如果归档日志缺失将会导致数据库恢复失败。只有库备份和表空间备份还原后，需要执行数据恢复，表还原结束后，不需要执行数据恢复。

■ 为了让读者更好地了解恢复的过程，这里先介绍一下 DM 的 MAGIC。PERMANENT\_MAGIC 和 DB\_MAGIC 是用来标识数据库的 INTEGER 类型值。DM 在初始化数据库时生成 PERMANENT\_MAGIC 和 DB\_MAGIC 值，其中 PERMANENT\_MAGIC 一经生成，永远不会改变（DDL\_CLONE 还原库的 PMNT\_MAGIC 除外），称为数据库永久魔数。只有 DDL\_CLONE 还原库的 PMNT\_MAGIC 会发生改变，当一个库使用 DDL\_CLONE 备份集还原并恢复之后，在执行 RECOVER DATABASE ..... UPDATE DB\_MAGIC 时，PMNT\_MAGIC 会发生改变。DB\_MAGIC 称为数据库魔数，同样可以用来表示某一个数据库，但 DB\_MAGIC



是可以变化的，每经过一次还原、恢复操作后，DB\_MAGIC 就会产生变化，用来区分备份源库和还原目标库。

可以通过下列语句查看系统的 PERMANENT\_MAGIC 和 DB\_MAGIC 值。

```
SELECT PERMANENT_MAGIC;  
  
SELECT DB_MAGIC FROM V$RLOG;
```

### 2.3.2.1 指定备份集恢复

默认未指定 WITHOUT LOG 子句的联机库备份生成的备份集，包含了备份过程中产生的 REDO 日志，数据还原结束后，可以直接指定备份集，将数据库恢复到备份结束时的状态。

由于执行增量备份时，要求  $\leq$  基备份 END\_LSN 的所有数据页已经写入磁盘；因此基备份集中包含的 REDO 日志不需要重做，只要重做指定执行还原操作的备份集中包含的 REDO 日志，就可以将数据库恢复到一致性状态。

指定备份集恢复的简要过程包括：

1. 从备份集读取 REDO 日志，并生成一个临时的本地归档日志文件；
2. 利用生成的临时归档日志文件，重做 REDO 日志，并将数据修改写入磁盘；
3. 删除临时生成的归档日志文件；
4. 更新数据库日志信息，设置 CKPT\_LSN 为最后一个重做的 REDO 日志 LSN 值；
5. 修改数据状态为 ACTIVE，标记数据库启动时需要进行相应的回滚活动事务、PURGE 已提交事务。

### 2.3.2.2 指定归档恢复

如果备份时指定了 WITHOUT LOG 子句，那么产生的备份集不包含备份过程中产生的 REDO 日志。这种备份集还原后，必须利用本地归档日志，将数据库恢复到一致性状态。执行恢复前，会检查本地归档日志文件的完整性，要求必须包括 [BEGIN\_LSN, END\_LSN] 之间完整的 REDO 日志。

利用本地归档日志进行恢复时，DMRMAN 工具会扫描指定的归档日志目录，收集与恢复数据库 PERMANENT\_MAGIC 值相等的归档日志文件。与指定备份集恢复相比，利用本地归档日志恢复不需要生成、删除临时归档日志文件，其余的执行流程完全相同。

指定归档恢复的执行场景主要包括：

- 将还原后处于非一致性状态的数据库恢复到一致性状态
- 将已经处于一致性状态的数据库尽可能地恢复到最新状态
- 将数据库恢复到指定时间点状态
- 将数据库恢复到指定 LSN 产生时的状态



**注意：**使用 **DDL CLONE** 方式备份的数据库，不支持指定归档恢复。

DM 中的归档日志包含时间信息，重做归档日志过程中，一旦发现达到了指定时间点，就马上终止归档日志重做。在出现误操作的情况下，通过指定时间点恢复，可能帮助用户修复数据。比如：用户在下午 5 点做了一个误操作，删除了某些重要数据；我们可以将恢复时间设置为下午 4:59 分，在恢复完成后，重新找回被误删的数据。

除了指定时间点，我们还可以通过指定 LSN 进行恢复。DM 中每条 REDO 日志记录都对应一个唯一的 LSN 值，指定 LSN 值以后，数据库将会精准的恢复到产生这个 LSN 时间点的状态。

### 2.3.2.3 更新 DB\_MAGIC

若备份集满足 BEGIN\_LSN 等于 END\_LSN，即在备份过程中未产生 REDO 日志，则使用此备份集还原后只需要更新 DB\_MAGIC 即可完成恢复。更新 DB\_MAGIC 不重做 REDO 日志，仅仅更新库的 DB\_MAGIC 值和数据库状态。



**注意：**只能在还原后的数据库上执行更新 **DB\_MAGIC** 操作。

### 2.3.2.4 表空间恢复

考虑到用户表空间上的数据库对象定义是保存在 SYSTEM 表空间的系统表内，而用户表空间仅保存这些数据库对象的数据，为了避免出现数据库对象的数据与定义不一致的情况，一般要求在表空间还原后，重做指定表空间所有 REDO 日志将这个表空间数据恢复到最新状

态。与表空间还原类似，表空间恢复也只能在脱机状态下通过 DMRMAN 工具完成。

表空间恢复的 REDO 日志是从本地归档日志文件中提取。表空间恢复同样要求满足归档日志覆盖 [BEGIN\_LSN, END\_LSN] 的要求。

注意：表空间恢复结束，并执行 ONLINE 表空间操作后，用户就可以访问这个表空间；但 ONLINE 操作并不会触发事务回滚，所以重做 REDO 日志产生的未 COMMIT 事务，也不会被回滚掉。

### 2.3.2.5 DMDSC 库恢复

DMDSC 库与普通单节点数据库的区别在于 DMDSC 库的多个节点共同维护一份库数据，每个节点上都有独立的联机日志和本地归档日志。重做 REDO 日志恢复时，需要重做所有节点上的 REDO 日志，因此需要提供各个节点的归档日志。

DMDSC 库恢复也支持未还原库恢复。

### 2.3.3 解密与解压缩

解密和解压缩是备份过程中加密和压缩的逆操作，如果若备份时未指定加密或压缩，还原和恢复过程中也不需要执行解密或者解压缩操作。

如果备份时进行了加密，那么还原时用户必须指定与备份时一致的加密密码和加密算法，否则还原会报错。如果备份时没有加密，那么还原时用户不需要指定加密密码和加密算法，如果指定了，也不起作用。DM 还原时的解密过程主要包括：

1. 检查用户输入的密码和算法是否与备份集中记录的加密信息一致；
2. 从备份集读取数据之后，写到目标文件（包括目标数据文件和临时归档文件）之前执行解密操作。

与解密不同，解压缩不需要用户干预，如果备份集指定了压缩，从备份集读取数据写到目标文件之前，会自动进行解压缩操作。

如果备份时既指定了加密又指定了压缩，那么与备份过程处理相反，还原时会先进行解密，再进行解压缩，然后将处理后的数据写入到目标文件中。

### 2.3.4 并行还原

指定并行备份生成的备份集，在还原时默认采用并行方式还原，并行度上限为备份时指定的并行数，实际并行度由 DMAP 最终创建成功的并行子任务数决定。并行备份产生的备份集，在还原时可以通过指定 `NOT PARALLEL` 子句关闭并行还原功能，以非并行方式还原。目前，非并行备份生成的备份集，不支持以并行方式还原。

## 2.4 归档日志备份与还原

除了通常意义上的数据备份、还原之外，DM 还支持对本地归档日志文件进行备份和还原。归档日志备份是数据库备份的一个有效补充，我们知道归档日志文件中保存了所有数据库操作产生的 REDO 日志，因此，在理论上，只要有一个基准备份集，加上完整的归档日志，我们可以将数据库恢复到任意时间点的状态。

### 2.4.1 归档日志备份

与联机备份收集备份过程中产生的 REDO 日志写入备份集不同，归档日志备份专门用来备份本地归档日志文件，将符合条件的本地归档日志文件拷贝到备份集中保存起来。

归档日志备份仅备份指定数据库生成的本地归档日志文件，要求归档日志文件的 `DB_MAGIC` 与数据库的 `DB_MAGIC` 保持一致。如果本地归档目录中包含多个不同数据库的归档日志文件，也只会备份一个特定数据库的归档日志。由于经过还原后数据库的 `DB_MAGIC` 会产生变化，因此即便 `PERMANENT_MAGIC` 相同，`DB_MAGIC` 不同的数据库产生的归档日志也不会备份。

与普通的数据库备份一样，归档日志备份也支持加密与压缩功能，可以联机执行归档日志备份，也可以在数据库关闭情况下使用 `DMRMAN` 工具进行脱机备份。归档日志备份时，可以指定是否删除已经备份的归档日志文件，在生成归档日志备份集的同时，删除本地归档日志文件，释放磁盘空间。

由于本地归档的异步实现机制，为了确保归档日志备份的完整性，一般会在归档日志备份之前执行一个归档切换动作。

## 2.4.2 归档日志还原

归档日志还原就是将备份集中的归档日志文件重新拷贝到指定归档目录中。使用归档日志备份集，既可以将归档日志文件还原到指定数据（还原时指定目标库的 `dm.ini`）的归档目录，也可以还原到用户指定的任意归档目录中。

归档日志还原的过程包括：

1. 根据过滤条件，从归档日志备份集收集需要还原的归档日志文件。
2. 在指定的归档目录创建归档文件。如果目标归档文件已经存在，默认采用认为该归档完好，生成一条日志记录，不再还原策略。也可以使用 `OVERWRITE` 指定策略。  
`OVERWRITE` 参数为：1 表示认为归档文件完好，不再还原该归档文件，添加一条日志记录；2 表示存在同名归档立即报错返回，终止还原；3 表示强制删除归档，重新还原同名归档。
3. 从备份集拷贝 REDO 日志，写入目标归档日志文件。

如果备份时指定了加密或压缩，还原过程中会先经过解密和解压缩处理，再写回到目标归档日志文件中。

## 3 备份还原实战

本章详细介绍如何利用 DM 提供的各种工具进行备份还原与恢复的操作，包括 DISQL 工具、DMRMAN 工具、图形化客户端管理工具 MANAGER 和 CONSOLE。DISQL 工具用于执行联机的数据备份与数据还原，包括数据库备份、归档备份、表空间备份与还原、表备份与还原；DMRMAN 工具用于执行脱机的数据备份、还原与恢复，包括脱机的数据库备份、还原与恢复，脱机还原表空间，归档的备份、还原与修复；客户端工具 MANAGER 和 CONSOLE 对应命令行工具 DISQL 和 DMRMAN 的功能，分别用于联机和脱机备份还原数据。这四种工具都可以独立使用，也可以相互配合，如使用 DISQL 或 MANAGER 工具联机备份的数据库备份文件可以用 DMRMAN 或 CONSOLE 工具还原。读者在阅读完本章的内容后，就可以动手配置归档，并进行备份与还原操作了。

本章内容主要包括：

- 准备工作
- 使用联机执行 SQL 语句进行备份还原
- 使用脱机工具 DMRMAN 进行备份还原
- 使用图形化客户端工具进行备份还原



**OPEN** 状态支持备份、还原和恢复的操作；

**警告：** **MOUNT** 状态支持归档备份、表空间级还原；

**SUPEND** 状态均不支持。

### 3.1 准备工作

本节介绍着手备份与还原之前的准备工作，主要包括：

- 支持与限制
- 归档配置

#### 3.1.1 支持与限制

在着手备份与还原之前，先详细了解一下 DM 对备份与还原进行了哪些支持与限制，如下进行详细说明。

### ■ 联机备份

对联机备份的支持与限制：

- 1) MPP 环境仅允许库和归档备份，且各节点都会执行，生成相应的备份集，支持 DDL CLONE；
- 2) DSC 环境支持库备份、表空间备份和表备份，要求 DSC 环境的所有节点都处于 OPEN 状态；
- 3) MOUNT 状态仅支持归档备份；
- 4) SUSPEND 状态所有备份均不支持；
- 5) OPEN 状态支持所有备份，支持 DDL CLONE；
- 6) PRIMARY 模式支持所有备份，支持 DDL CLONE；
- 7) STANDBY 模式仅支持库级、表空间级和归档备份，支持 DDL CLONE；
- 8) DDL CLONE 必须备份归档，不允许指定 WITHOUT LOG。

### ■ 联机还原：

仅支持表级还原，对联机还原的支持与限制：

- 1) MPP 不支持；
- 2) PRIMARY 支持；
- 3) MOUNT 支持表空间级还原，SUSPEND 均不支持；
- 4) OPEN/NORMAL 支持。

### ■ 脱机备份

脱机备份支持库级和归档备份。

- 1) MPP 视同单机环境，仅当前节点执行备份操作；
- 2) 允许异常退出后备份，支持 DDL\_CLONE；
- 3) DSC 支持库级备份，支持 DDL\_CLONE。

### ■ 脱机还原：

脱机还原跟目标库所处的模式、状态以及集群环境（MPP 和 DSC）无关，允许库级、表空间级和归档还原。

在一般的应用场景中，常规性的数据库维护工作，即在不影响数据库正常运行的情况下，建议定期执行联机数据库备份，且完全备份和增量备份结合使用。执行两次完全备份的时间间隔可以尽量长一点，在两次完全备份之间执行一定数量的增量备份，比如，可以选择每周执行一次完全备份，一周内每天执行一次增量备份。为了尽量减少对数据库正常工作的影响，

建议备份时间，选择在工作量较少的时间，比如深夜。



无论管理员选用哪一种备份方式，都应注意不应将备份产生的备份集与源备份说明：库存放在同一磁盘或同一存储介质上，以避免存储介质发生硬件故障时，源备份库与备份集同时被毁坏。



注意：备份与还原时，指定的备份集名称和目录名中最好不要包含中文、空格以及特殊字符，否则可能会因为字符处理及字符集问题导致一些不可预期的问题。

## 3.1.2 归档配置

### 3.1.2.1 概述

DM 数据库可以运行在归档模式或非归档模式下。如果是归档模式，联机日志文件中的内容保存到硬盘中，形成归档日志文件；如果是非归档模式，则不会形成归档日志。

采用归档模式会对系统的性能产生些许影响，然而系统在归档模式下运行会更安全，当出现故障时其丢失数据的可能性更小，这是因为一旦出现介质故障，如磁盘损坏时，利用归档日志，系统可被恢复至故障发生的前一刻，也可以还原到指定的时间点，而如果没有归档日志文件，则只能利用备份进行恢复。

通过 `dm.ini` 和 `dmarch.ini` 可以配置本地归档。`dmarch.ini` 生效的前提是 `dm.ini` 中的参数 `ARCH_INI` 置为 1。

DM 的 `dmarch.ini` 可以进行本地归档和远程归档的设置，DM 备份与还原过程中使用的日志均为本地归档日志。

`dmarch.ini` 中与备份还原相关的配置参数及其介绍见下表。

表 3.1 dmarch.ini 相关配置项

配置项	配置含义
[ARCH_NAME]	REDO 日志归档名
ARCH_TYPE	REDO 日志归档类型，LOCAL 表示本地归档，REMOTE 表示远程
ARCH_DEST	REDO 日志归档目标，LOCAL 对应归档文件存放路径；REMOTE 对应远程目标节点实例名



ARCH_FILE_SIZE	单个 REDO 日志归档文件大小，取值范围(64M~2048M)，缺省 1024M，即 1G
ARCH_SPACE_LIMIT	REDO 日志归档空间限制，当所有本地归档文件达到限制值时，系统自动删除最老的归档文件。0 表示无空间限制，取值范围(1024M~4294967294M)，缺省为 0
ARCH_INCOMING_PATH	仅 REMOTE 归档有效，对应远程归档存放在本节点的实际路径

### 3.1.2.2 何时配置归档

联机备份数据库必须要配置归档。联机备份时，大量的事务处于活动状态，为确保备份数据的一致性，需要同时备份一段日志（备份期间产生的 REDO 日志），因此要求数据库必须配置本地归档且归档必须处于开启状态。

脱机备份数据库可配置归档也可以不配置。正常退出的库的备份不需要考虑本地归档日志的完整性，可以不配置归档；但对于故障退出的库的备份要求因故障未刷盘的日志也必须存在于本地归档中，因此必须配置归档，如果本地归档缺失，需要用户先修复归档，然后再备份。

备份表空间属于联机备份，必须配置归档。

备份表虽然是联机完全备份，但不需要配置归档。因为表在还原之后不需要再进行恢复操作，用不到归档日志。

备份归档日志必须配置归档。

### 3.1.2.3 配置本地归档

归档配置有两种方式：一是联机归档配置，数据库实例启动情况下，使用 SQL 语句完成 dmarch.ini 和 ARCH\_INI 配置；二是手动配置归档，数据库实例未启动的情况下，手动编写 dmarch.ini 文件和设置参数 ARCH\_INI。下面将分别说明这两种归档如何配置。

#### 联机配置归档

使用 SQL 语句配置本地归档。

语法如下：

---

```
ALTER DATABASE <ADD|MODIFY|DELETE> ARCHIVELOG <归档配置语句>;
```

---

<归档配置语句>::= 'DEST = <归档目标>,TYPE = <归档类型>'

<归档类型>::=<local 方式>|<remote 方式>

<local 方式>::=LOCAL [,FILE\_SIZE = <文件大小>][,SPACE\_LIMIT = <空间大小限制>]

<remote 方式>::=REMOTE [,FILE\_SIZE = <文件大小>][,SPACE\_LIMIT = <空间大小限制>],INCOMING\_PATH = <归档存放路径>

使用 SQL 语句开启和关闭归档模式。

语法如下：

```
ALTER DATABASE ARCHIVELOG | NOARCHIVELOG;
```



**警告：**

在归档模式下，不允许删除本地归档。

例如，联机归档配置如下：

1) 修改数据库为 MOUNT 状态。

```
SQL>ALTER DATABASE MOUNT;
```

2) 配置本地归档。

```
SQL>ALTER DATABASE ADD ARCHIVELOG 'DEST = /home/dm_arch/arch, TYPE = local,
FILE_SIZE = 1024, SPACE_LIMIT = 2048';
```

3) 开启归档模式。

```
SQL>ALTER DATABASE ARCHIVELOG;
```

4) 修改数据库为 OPEN 状态。

```
SQL>ALTER DATABASE OPEN;
```

### 手动配置归档

1) 手动编辑 dmarch.ini 文件，之后保存在 dm.ini 所在的目录。dmarch.ini 文件内容如下：

```
[ARCHIVE_LOCAL1]

ARCH_TYPE = LOCAL

ARCH_DEST = d:\dm_arch\arch

ARCH_FILE_SIZE = 1024

ARCH_SPACE_LIMIT = 2048
```

2) 编辑 dm.ini 文件，设置参数 ARCH\_INI=1，保存。

3) 启动数据库实例，数据库已运行于归档模式。



多路归档，指配置多个本地归档。配置的第一个归档，称为第一路归档，

说明：后面依次是第二路、第三路.....

### 3.1.2.4 配置远程归档

与本地归档一样，远程归档也是配置在 dmarch.ini 文件中，远程归档相关的主要几个配置项包括：

1. ARCH\_TYPE 设置为 REMOTE，表示是远程归档
2. ARCH\_DEST 设置为远程数据库实例名，表示 REDO 日志发送到这个节点
3. ARCH\_INCOMING\_PATH 设置为本地存储路径，用于保存 ARCH\_DEST 实例发送的 REDO 日志

一般建议 DMDSC 集群中的节点，在配置本地归档之外，再交叉配置集群中所有其他节点的远程归档。查询 V\$DM\_ARCH\_INI、V\$ARCH\_STATUS 等动态视图可以获取归档配置以及归档状态等相关信息。

下面以两节点 DMDSC 集群为例，说明如何配置远程归档，DSC0 和 DSC1 是 DMDSC 集群中的两个实例，交叉进行 REMOTE 归档配置：

DSC0 实例的 dmarch.ini 配置：

```
[ARCHIVE_LOCAL1]

ARCH_TYPE           = LOCAL

ARCH_DEST           = /dmdata/dameng/arch_dsc0

ARCH_FILE_SIZE      = 128

ARCH_SPACE_LIMIT     = 0

[ARCH_REMOTE1]

ARCH_TYPE           = REMOTE

ARCH_DEST           = DSC1

ARCH_INCOMING_PATH  = /dmdata/dameng/arch_dsc1

ARCH_FILE_SIZE      = 128

ARCH_SPACE_LIMIT     = 0
```

DSC1 实例的 dmarch.ini 配置:

```
[ARCHIVE_LOCAL1]

ARCH_TYPE          = LOCAL

ARCH_DEST           = /dmdata/dameng/arch_dsc1

ARCH_FILE_SIZE     = 128

ARCH_SPACE_LIMIT    = 0

[ARCH_REMOTE1]

ARCH_TYPE          = REMOTE

ARCH_DEST           = DSC0

ARCH_INCOMING_PATH  = /dmdata/dameng/arch_dsc0

ARCH_FILE_SIZE     = 128

ARCH_SPACE_LIMIT    = 0
```

## 3.2 使用联机执行 SQL 语句进行备份还原

本节介绍使用 DISQL 工具如何实现数据文件的备份、管理及还原。

本节内容主要包括:

- 概述
- 数据备份
- 数据备份高级主题
- 管理备份
- 数据还原
- 数据还原高级主题

### 3.2.1 概述

DM 支持通过联机执行 SQL 语句方式对数据库执行备份还原操作。联机方式支持数据库、用户表空间、用户表和归档的备份，用户表的还原。在进行联机库级备份、归档备份和表空间备份时，必须保证系统处于归档模式，否则联机备份不能进行。

联机备份之前，请设置 INI 参数 BAK\_USE\_AP，选用合适的备份方式。详细请参考 1.4 节 [备份方式](#)。



在使用的 SQL 语句中，若指定的备份名、加密算法名长度超过 128 个字节，  
注意：会导致语法分析错误。

需要说明的是，DM 新备份还原在执行联机操作时，语句的解析执行使用的是 DM 服务器的编码方式，而文件操作使用操作系统的编码方式。因此，当 DM 建库参数的编码方式与操作系统编码方式不一致时，使用中文文件名或路径名等可能造成控制台打印信息和日志文件中的信息的中文部分显示乱码。

## 3.2.2 数据备份

本节主要介绍使用 DISQL 工具如何备份数据库、用户表空间、用户表和归档。

### 3.2.2.1 数据库备份

本节描述了如何使用 DISQL 完成最基本的数据库备份及实施一些备份策略，如限制备份集大小、加密备份等。主要包括：

- 概述
- 设置备份选项
- 备份数据库

#### 3.2.2.1.1 概述

在 DISQL 工具中使用 BACKUP 语句你可以备份整个数据库。通常情况下，在数据库实例配置归档后输入以下语句即可备份数据库：

```
SQL>BACKUP DATABASE BACKUPSET 'db_bak_01';
```

语句执行完后会在默认的备份路径下生成名为“db\_bak\_01”的备份集目录，默认的备份路径为 dm.ini 中 BAK\_PATH 配置的路径，若未配置，则使用 SYSTEM\_PATH 下的 bak 目录。这是最简单的数据库备份语句，如果要设置其他的备份选项需了解联机备份数据库的语法。

语法如下：

---

```
BACKUP DATABASE [[FULL] [DDL_CLONE]] | INCREMENT [CUMULATIVE][WITH BACKUPDIR '<
```

---

---

```
基备份搜索目录>'{'<基备份搜索目录>'} | [BASE ON <BACKUPSET '<基备份目录>'] ] [TO <备份名>] [BACKUPSET '<备份集路径>']  
  
[DEVICE TYPE <介质类型> [PARMS '<介质参数>'] ]  
  
[BACKUPINFO '<备份描述>'] [MAXPIECESIZE <备份片限制大小>]  
  
[IDENTIFIED BY <密码>[WITH ENCRYPTION<TYPE>][ENCRYPT WITH <加密算法>]]  
  
[COMPRESSED [LEVEL <压缩级别>]] [WITHOUT LOG]  
  
[TRACE FILE '<TRACE 文件名>'] [TRACE LEVEL <TRACE 日志级别>]  
  
[TASK THREAD <线程数>][PARALLEL [<并行数>] [READ SIZE <拆分块大小>]];
```

---

**FULL:** 备份类型。FULL 表示完全备份，可不指定，默认为完全备份。

**DDL\_CLONE:** 数据库克隆。该参数只能用于完全备份中，表示仅拷贝所有的元数据不拷贝数据。如对于数据库中的表来说，只备份表的定义不备份表中数据。表空间和表备份不支持该参数。



说明:

**DDL\_CLONE 会克隆 HUGE 表的元数据信息，之后库还原时会还原 HUGE 表的表定义。**

**INCREMENT:** 备份类型。INCREMENT 表示增量备份，若要执行增量备份必须指定该参数。

**CUMULATIVE:** 用于增量备份中，指明为累积增量备份类型，若不指定则缺省为差异增量备份类型。

**WITH BACKUPDIR:** 用于增量备份中，指定基备份的搜索目录，最大长度为 256 个字节。若不指定，自动在默认备份目录下搜索基备份。如果基备份不在默认的备份目录下，增量备份必须指定该参数。

**BASE ON:** 用于增量备份中，指定基备份集目录。

**TO:** 指定生成备份名称。若未指定，系统随机生成，默认备份名格式为：DB\_库名\_备份类型\_备份时间。其中，备份时间为开始备份时的系统时间。

**BACKUPSET:** 指定当前备份集生成路径。若指定为相对路径，则在默认备份路径中生成备份集。若不指定，则在默认备份路径中按约定规则，生成默认备份集目录。库级备份默认备份集目录名生成规则：DB\_库名\_备份类型\_备份时间，如 DB\_DAMENG\_FULL\_20180518\_143057\_123456。表明该备份集为 2018 年 5 月 18 日

14时30分57秒123456毫秒时生成的库名为DAMENG的数据库完全备份集。若库名超长，使上述完整名称长度大于128个字节，则去掉库名字段，调整为DB\_备份类型\_备份时间。

DEVICE TYPE: 指存储备份集的介质类型，支持DISK和TAPE，默认DISK，详见1.3 [介质管理层](#)。

PARMS: 只对介质类型为TAPE时有效。

BACKUPINFO: 备份的描述信息。最大不超过256个字节。

MAXPIECESIZE: 最大备份片文件大小上限，以M为单位，最小128M，32位系统最大2G，64位系统最大128G。

IDENTIFIED BY: 指定备份时的加密密码。密码应用双引号括起来，这样避免一些特殊字符通不过语法检测。密码的设置规则遵行ini参数pwd\_policy指定的口令策略。

WITH ENCRYPTION: 指定加密类型，0表示不加密，不对备份文件进行加密处理；1表示简单加密，对备份文件设置口令，但文件内容仍以明文存；2表示完全数据加密，对备份文件进行完全的加密，备份文件以密文方式存储。当不指定WITH ENCRYPTION子句时，采用简单加密。

ENCRYPT WITH: 加密算法。当不指定ENCRYPT WITH子句时，使用AES256\_CFB加密算法。



### 加密算法包括:

说明: DES\_ECB、DES\_CBC、DES\_CFB、DES\_OFB、DESEDE\_ECB、  
DESEDE\_CBC、DESEDE\_CFB、DESEDE\_OFB、AES128\_ECB、  
AES128\_CBC、AES128\_CFB、AES128\_OFB、AES192\_ECB、  
AES192\_CBC、AES192\_CFB、AES192\_OFB、AES256\_ECB、  
AES256\_CBC、AES256\_CFB、AES256\_OFB、RC4

COMPRESSED: 取值范围0~9。0表示不压缩，1表示1级压缩，9表示9级压缩。压缩级别越高，压缩越慢，但压缩比越高。若未指定，但指定COMPRESSED，则默认1；否则，默认0。

WITHOUT LOG: 联机数据库备份是否备份日志。如果使用，则表示不备份，否则表示备份。如果使用了WITHOUT LOG参数，则使用DMRMAN工具还原时，必须指定WITH ARCHIVEDIR参数。

TRACE FILE: 指定生成的TRACE文件。启用TRACE，但不指定TRACE FILE时，

默认在 DM 数据库系统的 log 目录下生成 DM\_SBTTRACE\_年月.log 文件；若使用相对路径，则生成在执行码同级目录下。若用户指定，则指定的文件不能为已经存在的文件，否则报错；也不可以为 ASM 文件。

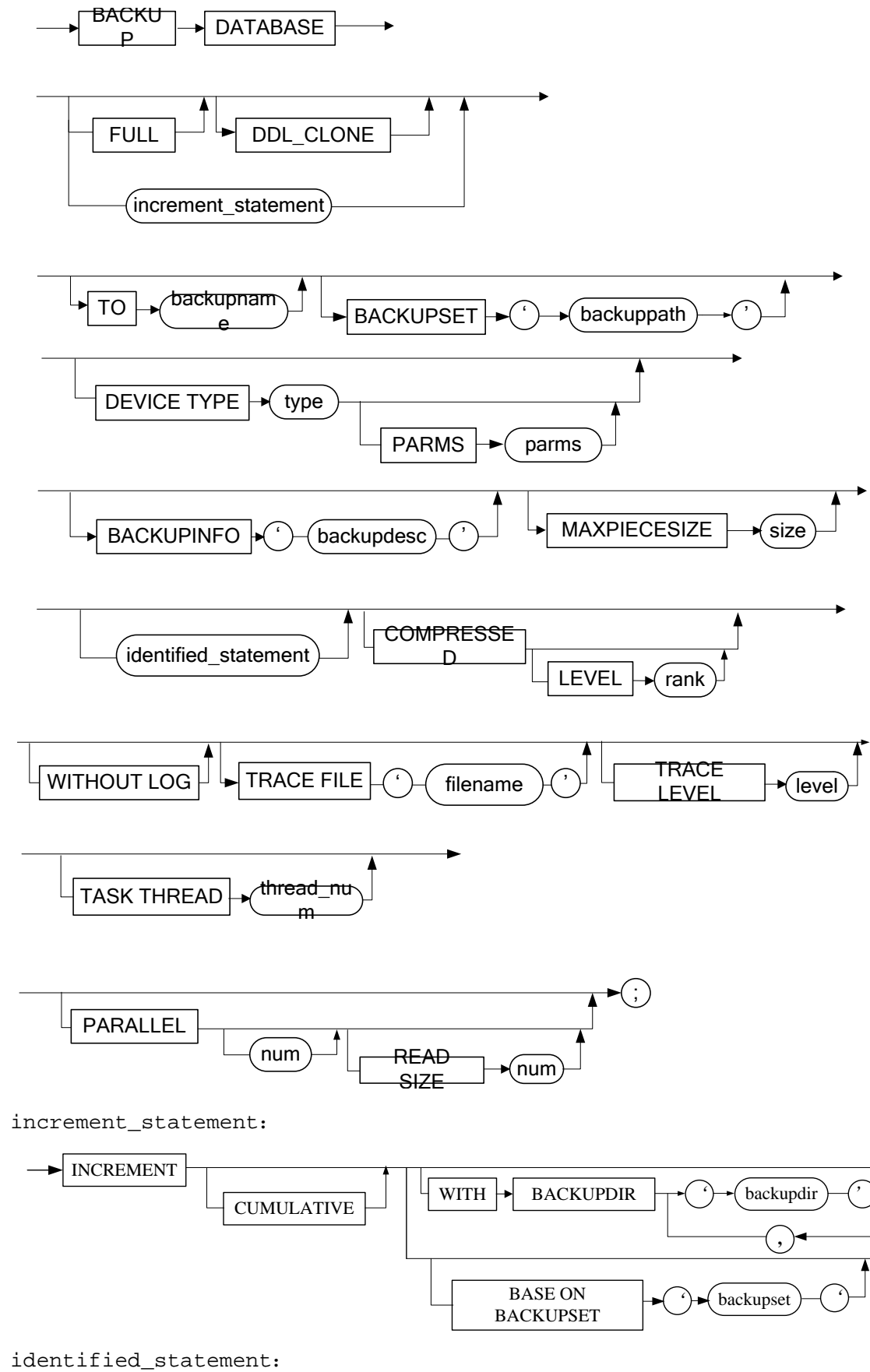
**TRACE LEVEL:** 有效值 1、2，默认为 1 表示不启用 TRACE，此时若指定了 TRACE FILE，会生成 TRACE 文件，但不写入 TRACE 信息；为 2 启用 TRACE 并写入 TRACE 相关内容。

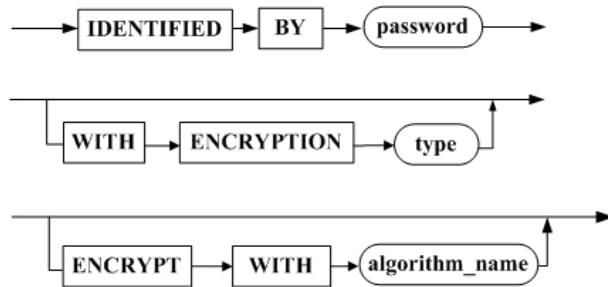
**TASK THREAD:** 备份过程中数据处理过程线程的个数，取值范围 0~64，默认为 4。若指定为 0，则调整为 1；若指定超过当前系统主机核数，则调整为主机核数。线程数（TASK THREAD）\* 并行数（PARALLEL）不得超过 512。

**PARALLEL:** 指定并行备份的<并行数>和 READ SIZE<拆分块大小>。<并行数>取值范围 0~128。若不指定<并行数>，则默认为 4，指定 0 或者 1 均认为为非并行备份。若未指定关键 PARALLEL，则认为非并行备份。并行备份不支持存在介质为 TAPE 的备份。线程数（TASK THREAD）\* 并行数（PARALLEL）不得超过 512。<拆分块大小>用户指定对于大数据量数据文件并行备份时拆分块的大小，默认为 1GB，最小不能低于 512MB，当指定的<拆分块大小>小于 512MB 时，系统会自动调整为 512MB。若指定并行备份，但未指定<拆分块大小>，则直接使用默认拆分块大小进行拆分。当数据文件的大小小于<拆分块大小>时，不执行拆分；当数据文件的大小大于<拆分块大小>时，执行拆分。并行数不能大于拆分之后的总块数。

**图例：**







#### 使用说明：

1. 备份成功后会在<备份集路径>或者备份默认目录下生成备份集。备份集中包括一个备份元数据文件，后缀.meta，一个或多个备份片文件，后缀.bak。
2. 当备份数据超过限制大小时，会生成新的备份文件，新的备份文件名是初始文件名后加文件编号。
3. 系统处于归档模式下时，才允许进行数据库联机备份。若备份库为 DSC 库，且未指定 WITHOUT LOG，则也需要配置 REMOTE 归档。
4. MOUNT 状态下不允许进行数据库备份。
5. DDL\_CLONE 库备份集不能作为增量备份的基备份，仅能用于库级还原。
6. 在执行联机数据库备份过程中，如果报错归档不完整，则需要先执行生成检查点操作，才能正常备份。例如：`select checkpoint(50);`

#### 3.2.2.1.2 设置备份选项

DISql 中备份语句如果仅指定了必选参数如“BACKUP DATABASE BACKUPSET 'db\_bak\_01';”，那么会默认地指定备份介质类型、备份路径、备份片大小及备份联机日志等。用户备份时也可以指定这些参数来修改默认值，本节将介绍以下几种常用的备份选项：

- 设置联机数据库备份集路径
- 设置备份名
- 增量备份指定基备份目录
- 指定介质类型
- 添加备份描述
- 限制备份片大小
- 加密备份
- 备份压缩

了解更多备份选项请参考 [3.2.3 数据备份高级主题](#)，数据库备份高级主题。

### 设置联机数据库备份集路径

备份语句中的 BACKUPSET 参数用于指定备份集的输出路径，例 3.1 备份数据库到指定的路径“/home/dm\_bak/db\_bak\_3\_01”。

例 3.1 指定备份集路径为“/home/dm\_bak/db\_bak\_3\_01”。

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_3_01';
```

BACKUPSET 为可选参数。如果该参数仅指定备份集名称即指定为相对路径，备份集会生成到默认的备份路径下。如果仅指定该关键字，不指定目标备份集目录，那么将会在默认备份路径下自动生成。

### 设置备份名

系统为每个备份指定一个备份名，它可作为识别备份的一种方式。备份时用户可以采用系统生成的备份名也可以指定属于自己的备份名。

备份名可以有效地表示备份的目的或者不同类型备份的用途。比如对于一个用于还原的数据库完全备份可设置备份名为“db\_full\_for\_restore”。

如果不显式地指定备份名，系统会默认为备份创建一个备份名。备份名的格式为：DB\_库名\_备份类型\_备份时间，DB 表示备份为数据库备份，备份类型表示该备份为完全备份还是增量备份，数据库名为当前连接数据库的名称，备份时间为开始执行备份的系统时间。

若一次备份产生多个备份片，那么每个备份片的备份名是相同的。

指定的备份名不能与默认备份路径中已有的备份名相同，但可以与非默认备份路径中已有的备份名相同。备份路径指备份集名称的上一层路径，如例 3.1 中的备份路径为“/home/dm\_bak”。

备份名最大长度为 128 个字节，如果超长会报语法分析出错。备份名的设置不可以使用特殊的格式，如%NAME。

例 3.2 创建备份集，备份名设置为“WEEKLY\_FULL\_BAK”。

```
SQL>BACKUP DATABASE TO WEEKLY_FULL_BAK BACKUPSET '/home/dm_bak/db_bak_3_02';
```

### 增量备份指定基备份目录

请参考 3.2.2.2.2 [设置备份选项](#)。

### 指定介质类型

具体请参考 1.3 [介质管理层](#)。

### 添加备份描述

联机备份可选择对执行的备份添加描述信息，相比备份名参数，描述信息可以更详细地

对备份类型、用途、场景等进行说明。描述信息最大长度支持 256 个字节。

例 3.4 创建备份为备份集添加描述信息为“完全备份”。

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_3_04' BACKUPINFO '完全备份';
```

用户如果不指定该参数，备份集对应的描述信息属性则为空。

### 限制备份片大小

MAXPIECESIZE 参数用于控制单个备份片的大小，当介质管理器对单个文件的大小有限制或者需要备份的数据文件很大时就可以使用这个参数。MAXPIECESIZE 不能大于磁盘剩余空间大小，否则报错磁盘空间不足。

MAXPIECESIZE 指定了单个备份片文件大小的上限，单位为 M，最小为 128M，32 位系统最大可设置为 2G，64 位系统最大可设置为 128G。如果不设置 32 系统默认为 2G，64 位系统默认为 4G。例如，要限制备份片大小不超过 300M，可指定 MAXPIECESIZE 300，备份时会限定所有的备份片大小不超过 300M。

限制备份片大小主要用于解决文件系统或介质管理器对文件最大值的限制小于备份片默认值的情况，当备份片较大时会导致无法存储。

例 3.5 创建备份限制备份片大小为 300M。

```
SQL> BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_3_05' MAXPIECESIZE 300;
```

### 备份压缩

所有的备份语句都可以设置压缩选项，指定 COMPRESSED 参数后备份集会先被压缩然后再写到磁盘或磁带。压缩选项有不同的压缩级别可以选择，取值范围为 0~9。0 表示不压缩，1 表示 1 级压缩，9 表示 9 级压缩。压缩级别越高，压缩越慢，但压缩比越高。若仅指定 COMPRESSED，压缩级别为 1。实际应用中用户可根据存储空间、数据文件大小、备份效率等确定合适的压缩级别。

例 3.6 执行备份压缩，压缩级别设置为 5。

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_3_06'COMPRESSED LEVEL 5;
```

### 并行备份

DM 支持对库级和表空间级的并行备份，待备份数据文件很大时使用并行备份可以显著提高备份效率。用户可通过关键字 PARALLEL 指定是否执行并行备份，以及执行并行备份的并行数。执行备份时实际使用的并行数由用户指定的并行数和实际可使用的最大并行数决定，而实际可使用的最大并行数由数据文件的个数决定。

例 3.7 创建并行备份，指定并行数为 8。

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_3_07'PARALLEL 8;
```

### 3.2.2.1.3 备份数据库

完全备份和增量备份是数据库备份的最常用策略，本节将详细介绍这两种类型备份。

#### 完全备份

执行数据库备份，数据库必须处于 OPEN 状态，MOUNT 和 SUSPEND 状态下不允许执行数据库备份。

数据库完全备份中包含了指定库的全部有效数据页，为了保证数据安全应该对数据库定期执行完全备份。多久执行一次完全备份才合适，一般规则是数据库经过频繁修改后最好执行一次完全备份。

完全备份数据库步骤如下：

- 1) 配置归档，参考 3.1 节。
- 2) 保证数据库处于 OPEN 状态。
- 3) DISql 中输入备份数据库语句，最简单的不设置其他参数的完全备份语句如下：

```
SQL>BACKUP DATABASE FULL BACKUPSET '/home/dm_bak/db_full_bak_01';
```

备份语句中的参数 FULL 参数可以省略，不指定备份类型会默认指定备份类型为完全备份。

#### 增量备份

增量备份指基于指定的库（或者表空间）的某个备份（完全备份或者增量备份），备份自该备份以来所有发生修改了的数据页。执行增量备份的主要目的是快速备份数据库中的修改，减少备份时间和避免重复的备份。

如何制定备份策略要根据可接受的最小恢复时间。例如，每周执行一次完全备份，每天执行一次增量备份，那么恢复时要重做的归档就不会超过一天。

增量备份数据库步骤如下：

- 1) 配置归档，参考 3.1 节。
- 2) 保证数据库处于 OPEN 状态。
- 3) DISql 中输入备份数据库语句，最简单的不设置其他参数的增量备份语句如下：

```
SQL>BACKUP DATABASE INCREMENT WITH BACKUPDIR '/home/dm_bak'BACKUPSET  
'/home/dm_bak/db_increment_bak_02';
```

备份语句中的 `INCREMENT` 参数不可省略，该参数用来指定执行的备份类型为增量备份。若要创建累积增量备份，还需要指定 `CUMULATIVE` 参数，否则缺省为差异增量备份。`WITH BACKUPDIR` 参数用来指定基备份集的搜索目录，如果基备份集不在默认目录该参数不可省略。

### 3.2.2.1.4 DSC 环境使用说明

DSC 环境下库备份情况与单机环境下类似，连接到 DSC 集群中的任意活动节点即可完成库备份操作。当 DSC 集群中存在故障节点时，库备份存在额外的限制：

1. 保证故障节点的 `[CKPT_LSN, FILE_LSN]` 之间日志修改的数据页已写入磁盘。只有所有活动节点的 `CKPT_LSN` 大于等于故障节点的 `FILE_LSN` 才允许备份操作。

2. 主库备份时，只有所有活动节点 `CKPT_LSN` 推进到 `MAX[APPLY_LSN]` 之后，才允许备份操作。

3. 备库备份时，需要触发检查点将所有活动节点 `CKPT_LSN` 推进到 `MAX[APPLY_LSN]` 之后，才能启动备份操作。

根据以上限制，DSC 集群联机备份时，如果活动节点 `CKPT_LSN` 小于故障节点 `FILE_LSN`，则强制推进检查点然后执行备份。对于故障节点，直接取其 `FILE_LSN` 及对应 `PKG_SEQNO` 作为 `BEGIN_LSN` 和 `BEGIN_SEQ`。该种策略下，由于故障节点所有修改数据页已经写入磁盘，无论活动节点推进到什么值，故障节点总是不需要备份该节点日志。

### 3.2.2.2 表空间备份

本节描述了如何使用 `DISQL` 完成最基本的表空间备份及实施一些备份策略，如限制备份集大小、加密备份等。主要包括：

- 概述
- 设置备份选项
- 备份表空间

#### 3.2.2.2.1 概述

在 `DISQL` 工具中使用 `BACKUP` 语句也可以备份单个表空间。同备份数据库一样，执行

表空间备份数据库实例也必须运行在归档模式下，启动 DISQL 输入以下语句即可备份表空间：

```
SQL>BACKUP TABLESPACE MAIN BACKUPSET 'ts_bak_01';
```

备份集“ts\_bak\_01”会生成到默认的备份路径下。如要设置其他备份选项需参考下文的联机备份表空间语法。

语法如下：

```
BACKUP TABLESPACE <表空间名> [FULL | INCREMENT [CUMULATIVE]][WITH BACKUPDIR '<基  
备份搜索目录>'{'<基备份搜索目录>'}]| [BASE ON BACKUPSET '<基备份集目录>']][TO <备份  
名>] BACKUPSET ['<备份集路径>']  
  
[DEVICE TYPE <介质类型> [PARMS '<介质参数>']]  
  
[BACKUPINFO '<备份集描述>'] [MAXPIECESIZE <备份片限制大小>]  
  
[IDENTIFIED BY <加密密码>[WITH ENCRYPTION<TYPE>][ENCRYPT WITH <加密算法>]]  
  
[COMPRESSED [LEVEL <压缩级别>]]  
  
[TRACE FILE '<TRACE 文件名>'] [TRACE LEVEL <TRACE 日志级别>]  
  
[TASK THREAD <线程数>][PARALLEL [<并行数>][READ SIZE <拆分块大小>]  
];
```

**表空间名：**指定备份的表空间名称（除了 temp 表空间）。

**FULL|INCREMENT：**备份类型，FULL 表示完全备份，INCREMENT 表示增量备份。若不指定，默认为完全备份。

**CUMULATIVE：**用于增量备份中，指明为累积增量备份类型，若不指定则缺省为差异增量备份类型。

**WITH BACKUPDIR：**用于增量备份中，指定备份目录，最大长度为 256 个字节。若不指定，自动在默认备份目录下搜索基备份。如果基备份不在默认的备份目录下，增量备份必须指定该参数。

**BASE ON：**用于增量备份中，指定基备份集目录。

**TO：**指定生成备份名称。若未指定，系统随机生成，默认备份名格式为：DB\_备份类型\_表空间名\_备份时间。其中，备份时间为开始备份的系统时间。

**BACKUPSET：**指定当前备份集生成路径。若指定为相对路径，则在默认备份路径中生成备份集。若不指定，则在默认备份路径下以约定规则生成默认的表空间备份集目录。表空间级备份默认备份集目录名生成规则：TS\_表空间名\_备份类型\_时间，如

TS\_MAIN\_INCREMENT\_20180518\_143057\_123456。表明该备份集为 2018 年 5 月 18 日 14 时 30 分 57 秒 123456 毫秒时生成的表空间名为 MAIN 的表空间增量备份集。若表空间名称超长，使上述完整名称长度大于 128 个字节，则去掉表空间名字段，调整为 TS\_备份类型\_时间。

**DEVICE TYPE:** 指存储备份集的介质类型，支持 DISK 和 TAPE，默认 DISK，详见 1.3 介质管理层。

**PARMS:** 只对介质类型为 TAPE 时有效。

**BACKUPINFO:** 备份的描述信息。最大不超过 256 个字节。

**MAXPIECESIZE:** 最大备份片文件大小上限，以 M 为单位，最小 128M，32 位系统最大 2G，64 位系统最大 128G。

**IDENTIFIED BY:** 指定备份时的加密密码。密码应用双引号括起来，这样避免一些特殊字符通不过语法检测。密码的设置规则遵行 ini 参数 pwd\_policy 指定的口令策略。

**WITH ENCRYPTION:** 指定加密类型，0 表示不加密，不对备份文件进行加密处理；1 表示简单加密，对备份文件设置口令，但文件内容仍以明文存；2 表示完全数据加密，对备份文件进行完全的加密，备份文件以密文方式存储。当不指定 WITH ENCRYPTION 子句时，采用简单加密。

**ENCRYPT WITH:** 加密算法。当不指定 ENCRYPT WITH 子句时，使用 AES256\_CFB 加密算法。



### 加密算法包括:

**说明:** DES\_ECB、DES\_CBC、DES\_CFB、DES\_OFB、DESEDE\_ECB、  
DESEDE\_CBC、DESEDE\_CFB、DESEDE\_OFB、AES128\_ECB、  
AES128\_CBC、AES128\_CFB 、AES128\_OFB、AES192\_ECB、  
AES192\_CBC、AES192\_CFB 、AES192\_OFB、AES256\_ECB、  
AES256\_CBC、AES256\_CFB 、AES256\_OFB 、RC4

**COMPRESSED:** 取值范围 0~9。0 表示不压缩，1 表示 1 级压缩，9 表示 9 级压缩。压缩级别越高，压缩越慢，但压缩比越高。若未指定，但指定 COMPRESSED，则默认 1；否则，默认 0。

**TRACE FILE:** 指定生成的 TRACE 文件。启用 TRACE，但不指定 TRACE FILE 时，默认在 DM 数据库系统的 log 目录下生成 DM\_SBTTRACE\_年月.LOG 文件；若使用相对路



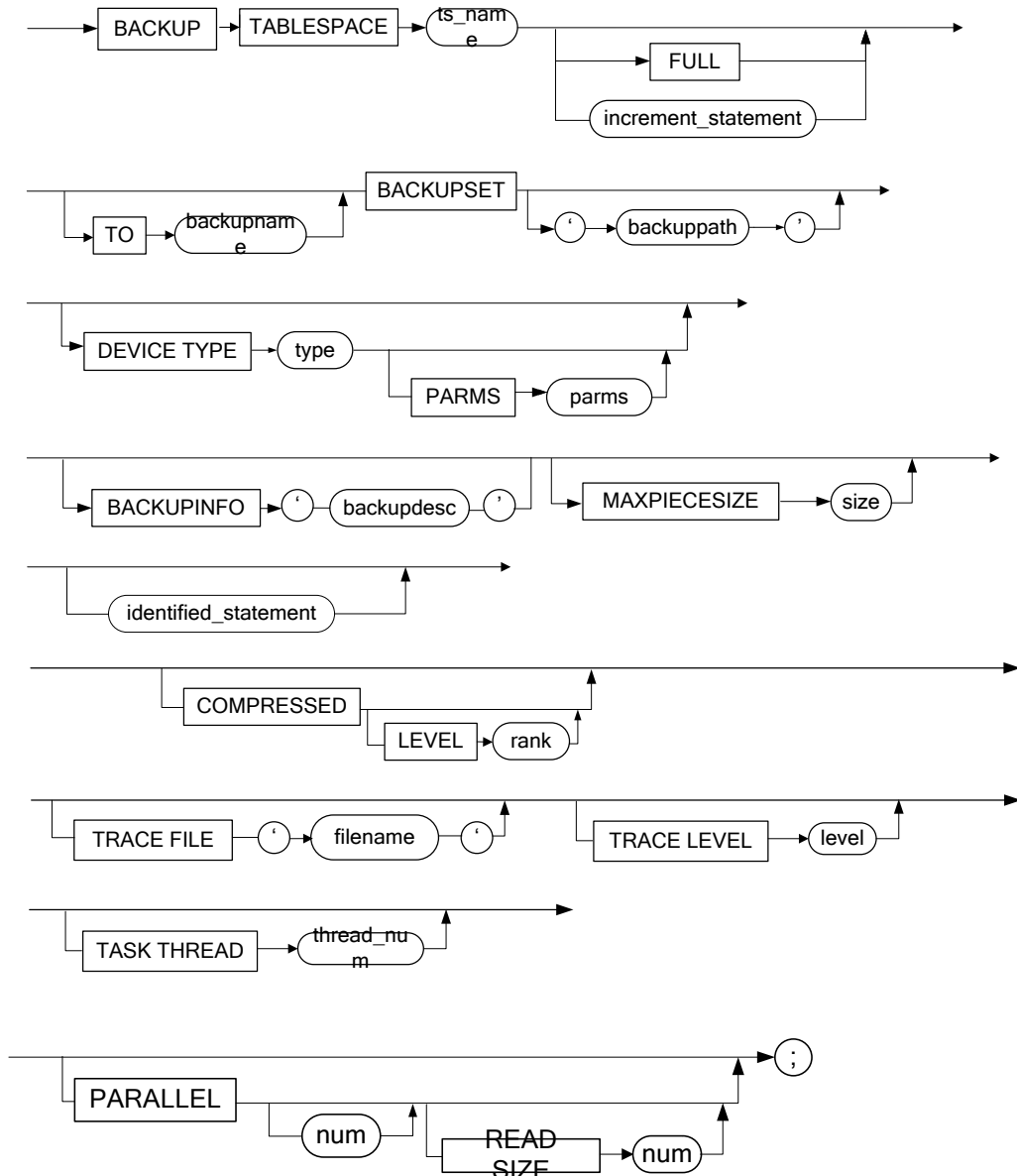
径，则生成在执行码同级目录下。若用户指定，则指定的文件不能为已经存在的文件，否则报错；也不可以为 ASM 文件。

TRACE LEVEL: 有效值 1、2, 默认为 1 表示不启用 TRACE, 此时若指定了 TRACE FILE, 会生成 TRACE 文件, 但不写入 TRACE 信息; 为 2 启用 TRACE 并写入 TRACE 相关内容。

TASK THREAD: 备份过程中数据处理过程线程的个数, 取值范围 0~64, 默认为 4。若指定为 0, 则调整为 1; 若指定超过当前系统主机核数, 则调整为主机核数。线程数 (TASK THREAD) \* 并行数 (PARALLEL) 不得超过 512。

PARALLEL: 指定并行备份的<并行数>和 READ SIZE<拆分块大小> 请参考 [3.2.2.1.1 概述](#)。

图例:



`increment_statement`、`identified_statement`：请参考 3.2.2.1.1 [概述中](#) 的数据库备份图例

### 使用说明：

1. 当备份数据超过限制大小时，会生成新的备份文件，新的备份文件名是初始文件名后加文件编号。
2. 系统处于归档模式下时，才允许进行表空间备份。
3. MOUNT 状态下，不允许进行表空间备份。
4. MPP 环境不允许进行表空间备份。

### 3.2.2.2.2 设置备份选项

表空间备份也可指定备份集路径、介质类型、备份名等备份选项，如何设置这些选项可参考 3.2.2.1.2 小节。本节主要介绍 BASE ON 参数的使用。

#### 增量备份指定基备份目录

BASE ON 参数用于增量备份中，用来指定基备份集的目录。如果不指定该参数，会在备份搜索目录中搜索最近一次的完全备份或增量备份作为这次增量备份的基备份。若需要在特定的备份集基础上执行增量备份就需要使用该参数。

下面以增量备份用户 MAIN 表空间为例，指定 BASE ON 参数执行增量备份：

```
SQL>BACKUP TABLESPACE MAIN BACKUPSET 'ts_full_bak_01';

SQL>BACKUP TABLESPACE MAIN INCREMENT BACKUPSET 'ts_increment_bak_01';

SQL>BACKUP TABLESPACE MAIN INCREMENT BASE ON BACKUPSET'ts_full_bak_01' BACKUPSET
'ts_increment_bak_02';
```

上述示例中，增量备份 ts\_increment\_bak\_02 若不指定备份集 ts\_full\_bak\_01 作为基备份，那么默认会使用最近一次的备份集 ts\_increment\_bak\_01 作为基备份。

### 3.2.2.2.3 备份表空间

同数据库备份一样，表空间备份也可分为完全备份和增量备份。本节主要包括：

- 完全备份
- 增量备份

#### 完全备份

执行表空间备份，表空间必须处于 OPEN 状态，MOUNT 和 SUSPEND 状态下不允许执行表空间备份。表空间备份就是拷贝表空间内所有数据文件有效数据的过程。DM 仅支持表空

间联机备份，完全备份一个表空间步骤如下：

- 1) 配置归档，参考 3.1 节。
- 2) 保证数据库处于 OPEN 状态。
- 3) DISql 中输入备份表空间语句，最简单的不设置其他参数的完全备份表空间语句，

如下所示：

```
SQL>BACKUP TABLESPACE MAIN FULL BACKUPSET '/home/dm_bak/ts_full_bak_01';
```

备份语句中的 FULL 参数可以省略，不指定备份类型会默认指定备份类型为完全备份。

### 增量备份

执行表空间增量备份的主要目的同数据库增量备份一样是为了快速备份数据库中的修改，减少备份时间和避免重复的备份。

增量备份表空间步骤如下：

- 1) 配置归档，参考 3.1 [准备工作](#)。
- 2) 保证数据库处于 OPEN 状态。
- 3) DISql 中输入备份表空间语句，最简单的不设置其他参数的增量备份语句如下：

```
SQL>BACKUP TABLESPACE MAIN INCREMENT WITH BACKUPDIR '/home/dm_bak'BACKUPSET  
'/home/dm_bak/ts_increment_bak_02';
```

备份语句中指定的 INCREMENT 参数表示执行的备份类型为增量备份，不可省略。若要创建累积增量备份，还需要指定 CUMULATIVE 参数，否则缺省为差异增量备份。若基备份不在默认备份目录，WITH BACKUPDIR 参数必须指定，用于搜索基备份集。

## 3.2.2.3 表备份

本节主要介绍使用 DISql 完成表备份及实施一些备份策略，如限制备份集大小、加密备份等。主要包括：

- 概述
- 设置备份选项
- 备份表

### 3.2.2.3.1 概述

与备份数据库与表空间不同，备份表不需要服务器配置归档，DISql 中输入以下即可

备份用户表：

```
SQL>BACKUP TABLE TAB_01 BACKUPSET 'tab_bak_01';
```

备份集“tab\_bak\_01”会生成到默认的备份路径下。如要设置其他备份选项需参考下文的备份表语法。

语法如下：

---

**BACKUP** TABLE <表名>

[TO <备份名>]

BACKUPSET ['<备份集路径>'] [DEVICE TYPE <介质类型> [PARMS '<介质参数>']]

[BACKUPINFO '<备份集描述>']

[MAXPIECESIZE <备份片限制大小>]

[IDENTIFIED BY <加密密码>[WITH ENCRYPTION<TYPE>][ENCRYPT WITH <加密算法>]]

[COMPRESSED [LEVEL <压缩级别>]]

[TRACE FILE '<trace 文件名>'] [TRACE LEVEL <trace 日志级别>]

---

**TABLE：**指定备份的表，只能备份用户表。

**TO：**指定生成备份名称。若未指定，系统随机生成，默认备份名格式为：DB\_备份类型\_表名\_备份时间。其中，备份时间为开始备份的系统时间。

**BACKUPSET：**指定当前备份集生成路径，若指定为相对路径，则在默认备份路径中生成备份集。若不指定具体备份集路径，则在默认备份路径下以约定规则生成默认的表备份集目录。表备份默认备份集目录名生成规则：TAB\_表名\_BTREE\_时间，如TAB\_T1\_BTREE\_20180518\_143057\_123456。表明该备份集为2018年5月18日14时30分57秒123456毫秒时生成的表名为T1的表备份集。若表名超长，使上述完整名称长度大于128个字节，则去掉表名字段，调整为TAB\_BTREE\_时间。

**DEVICE TYPE：**指存储备份集的介质类型，表备份暂时只支持DISK。

**PARMS：**只对介质类型为TAPE时有效。

**BACKUPINFO：**备份的描述信息。最大不超过256个字节。

**MAXPIECESIZE：**最大备份片文件大小上限，以M为单位，最小128M，32位系统最大2G，64位系统最大128G。

**IDENTIFIED BY：**指定备份时的加密密码。密码应用双引号括起来，这样避免一些特殊字符通不过语法检测。密码的设置规则遵行ini参数pwd\_policy指定的口令策略。

**WITH ENCRYPTION:** 指定加密类型，0 表示不加密，不对备份文件进行加密处理；1 表示简单加密，对备份文件设置口令，但文件内容仍以明文存储；2 表示完全数据加密，对备份文件进行完全的加密，备份文件以密文方式存储。当不指定 **WITH ENCRYPTION** 子句时，采用简单加密。

**ENCRYPT WITH:** 加密算法。当不指定 **ENCRYPT WITH** 子句时，使用 **AES256\_CFB** 加密算法。



### 加密算法包括：

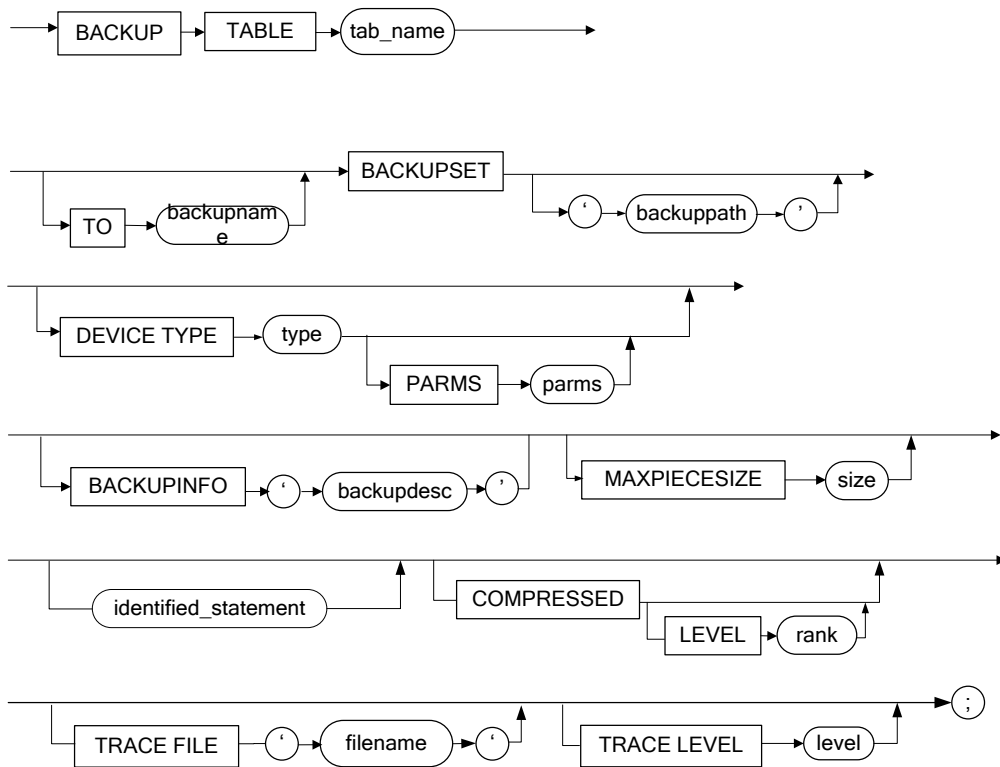
**说明：** **DES\_ECB、DES\_CBC、DES\_CFB、DES\_OFB、DESEDE\_ECB、DESEDE\_CBC、DESEDE\_CFB、DESEDE\_OFB、AES128\_ECB、AES128\_CBC、AES128\_CFB 、AES128\_OFB、AES192\_ECB、AES192\_CBC、AES192\_CFB 、AES192\_OFB、AES256\_ECB、AES256\_CBC、AES256\_CFB 、AES256\_OFB 、RC4**

**COMPRESSED:** 取值范围 0~9。0 表示不压缩，1 表示 1 级压缩，9 表示 9 级压缩。压缩级别越高，压缩越慢，但压缩比越高。若未指定，但指定 **COMPRESSED**，则默认 1；否则，默认 0。

**TRACE FILE:** 指定生成的 **TRACE** 文件。启用 **TRACE**，但不指定 **TRACE FILE** 时，默认在 DM 数据库系统的 **log** 目录下生成 **DM\_SBTTRACE\_年月.LOG** 文件；若使用相对路径，则生成在执行码同级目录下。若用户指定，则指定的文件不能为已经存在的文件，否则报错；也不可以为 **ASM** 文件。

**TRACE LEVEL:** 有效值 1、2，默认为 1 表示不启用 **TRACE**，此时若指定了 **TRACE FILE**，会生成 **TRACE** 文件，但不写入 **TRACE** 信息；为 2 启用 **TRACE** 并写入 **TRACE** 相关内容。

**图例：**



increment\_statement: 请参考 3.2.2.1.1 [概述中的](#) 数据库备份图例

使用说明:

1. 支持对用户的非分区行存储表和堆表进行备份。其中，临时表、物化视图表、物化视图附属表和日志表、特定模式

(DBG\_PKG/INFORMATION\_SCHEMA/INFO\_SCHEM/SYSREP/SYSGEO/SYSJOB/SYSCPT/SYS) 下的表不支持备份。表类型为对象类型的表不支持表备份。表备份不备份表上的注释，default 表达式中函数定义，所以还原时需用户自行确认。表名中包含保留字的表不允还备份还原。

2. 当备份数据超过限制大小时，会生成新的备份文件，新的备份文件名是初始文件名后加文件编号。

3. 表备份时，其所属表空间必须处于联机状态。

4. 目前表备份不支持备份到 TAPE 介质上。

### 3.2.2.3.2 设置备份选项

表备份常用的备份选项有设置备份名、设置备份集路径、指定介质参数、添加备份描述等，设置方式同数据库备份相同，具体参考 3.2.2.1.2 [设置备份选项](#)。

### 3.2.2.3.3 备份表

表备份拷贝指定表所使用的所有数据页到备份集中,并记录各个数据页之间的逻辑关系用来恢复表数据结构。表备份均为联机完全备份,不需要备份归档日志,不存在增量备份之说。当数据库中某张表比较重要而又没必要备份整个数据库或表空间时就可以选择表备份。

完整的备份表步骤如下:

1) 保证数据库处于 OPEN 状态。

2) 创建待备份的表 TAB\_01:

```
SQL>CREATE TABLE TAB_01(C1 INT);
```

3) DISql 中输入备份表语句,简单的备份语句如下:

```
SQL>BACKUP TABLE TAB_01 BACKUPSET '/home/dm_bak/tab_bak_01';
```

### 3.2.2.4 归档备份

本节主要介绍使用 DISql 完成归档备份及实施一些备份策略,如限制备份集大小、加密备份等。主要包括:

- 概述
- 设置备份选项
- 备份归档

#### 3.2.2.4.1 概述

在 DISql 工具中使用 BACKUP 语句可以备份归档日志。使用归档备份的前提:一是,归档文件的 db\_magic、permanent\_magic 值和库的 db\_magic、permanent\_magic 值必须一样;二是,服务器必须配置归档;三是,归档日志必须连续,如果出现不连续的情况,前面的会忽略,仅备份最新的连续部分。如果未收集到指定范围内的归档,则不会备份。联机备份的时候经常会切换归档文件,最后一个归档总是空的,所以最后一个归档不会被备份。

DISql 中输入以下即可备份归档:

```
SQL>BACKUP ARCHIVE LOG ALL BACKUPSET 'arch_bak_01';
```

备份集“arch\_bak\_01”会生成到默认的备份路径下。如要设置其他备份选项需参考下文



的备份归档语法。

语法如下：

---

```

BACKUP <ARCHIVE LOG | ARCHIVELOG>

[ALL | [FROM LSN <lsn>] | [UNTIL LSN <lsn>] | [LSN BETWEEN <lsn> AND <lsn>] | [FROM
TIME '<time>'] | [UNTIL TIME '<time>'] | [TIME BETWEEN '<time>' AND '<time>']
][<notBackedUpSpec>][DELETE INPUT]

[TO <备份名>][<备份集子句>;

<备份集子句>::=BACKUPSET ['<备份集路径>'][DEVICE TYPE <介质类型> [PARMS '<介质参数>']]

[BACKUPINFO '<备份描述>']

[MAXPIECESIZE <备份片限制大小>]

[IDENTIFIED BY <密钥>[WITH ENCRYPTION<TYPE>][ENCRYPT WITH <加密算法>]]

[COMPRESSED [LEVEL <压缩级别>]]

[WITHOUT LOG]

[TRACE FILE '<trace 文件名>'] [TRACE LEVEL <trace 日志级别>]

[TASK THREAD <线程数>][PARALLEL [<并行数>][READ SIZE <拆分块大小>]];

```

---

ALL：备份所有的归档；

FROM LSN：指定备份的起始 lsn。

UNTIL LSN：指定备份的截止 lsn。



归档日志的有效 LSN 范围（起始 lsn, 截止 lsn）可以通过 V\$ARCH\_FILE 查看，或者通过 dmclvt 工具分析日志的结果查看。

说明：

如果用户无法确定准确的 lsn，也可以指定一个模糊的 lsn 值，取值范围（1~9223372036854775807）。只要指定的 FROM LSN、UNTIL LSN 与有效 LSN 范围有重叠部分，就会备份包含重叠部分的完整日志文件。

FROM TIME：指定备份的开始的时间点。

UNTILTIME：指定备份的截止的时间点。

BETWEEN ... AND ...：指定备份的区间。指定区间后，只会备份指定区间内的归档文件。

<notBackedUpSpec>：搜索过滤。搜索过滤仅限于根据备份指定条件能找到的所有归档备份集。1) num TIMES，取值范围为 0~2147483647，指若归档文件已经备份了 num

次，则不再备份；否则备份。如 num=3，则认为已经备份了 3 次的归档文件就不再备份。若 num=0，则认为所有都不需要备份。2) SINCE TIME 'datetime\_String'，指定时间开始没有备份的归档文件进行备份。3) 若以上两种均未指定，则备份所有未备份过的归档日志文件。

**DELETE INPUT:** 用于指定备份完成之后，是否删除归档操作。

**TO:** 指定生成备份名称。若未指定，系统随机生成，默认备份名格式为：ARCH\_备份时间。其中，备份时间为开始备份的系统时间。

**BACKUPSET:** 指定当前备份集生成路径，若指定为相对路径，则在默认备份路径中生成备份集。若不指定具体备份集路径，则在默认备份路径下，以约定归档备份集命名规则生成默认的归档备份集目录。归档备份默认备份集目录名生成规则：ARCH\_LOG\_时间，如 ARCH\_LOG\_20180518\_143057\_123456。表明该备份集为 2018 年 5 月 18 日 14 时 30 分 57 秒 123456 毫秒时生成的归档备份集。

**DEVICE TYPE:** 指存储备份集的介质类型，支持 DISK 和 TAPE，默认 DISK，详见 1.3 介质管理层。

**PARMS:** 只对介质类型为 TAPE 时有效。

**BACKUPINFO:** 备份的描述信息。最大不超过 256 个字节。

**MAXPIECESIZE:** 最大备份片文件大小上限，以 M 为单位，最小 128M，32 位系统最大 2G，64 位系统最大 128G。

**IDENTIFIED BY:** 指定备份时的加密密码。密码应该使用双引号括起来，这样避免一些特殊字符通不过语法检测。密码的设置规则遵行 ini 参数 pwd\_policy 指定的口令策略。

**WITH ENCRYPTION:** 指定加密类型，0 表示不加密，不对备份文件进行加密处理；1 表示简单加密，对备份文件设置口令，但文件内容仍以明文存储；2 表示完全数据加密，对备份文件进行完全的加密，备份文件以密文方式存储。当不指定 WITH ENCRYPTION 子句时，采用简单加密。

**ENCRYPT WITH:** 加密算法。当不指定 ENCRYPT WITH 子句时，使用 AES256\_CFB 加密算法。



加密算法包括:

说明: DES\_ECB、DES\_CBC、DES\_CFB、DES\_OFB、DESEDE\_ECB、  
DESEDE\_CBC、DESEDE\_CFB、DESEDE\_OFB、AES128\_ECB、  
AES128\_CBC、AES128\_CFB 、AES128\_OFB、AES192\_ECB、  
AES192\_CBC、AES192\_CFB 、AES192\_OFB、AES256\_ECB、  
AES256\_CBC、AES256\_CFB 、AES256\_OFB 、RC4

COMPRESSED: 取值范围 0~9。0 表示不压缩, 1 表示 1 级压缩, 9 表示 9 级压缩。压缩级别越高, 压缩越慢, 但压缩比越高。若未指定, 但指定 COMPRESSED, 则默认 1; 否则, 默认 0。

WITHOUT LOG: 只是语法支持, 不起任何作用。

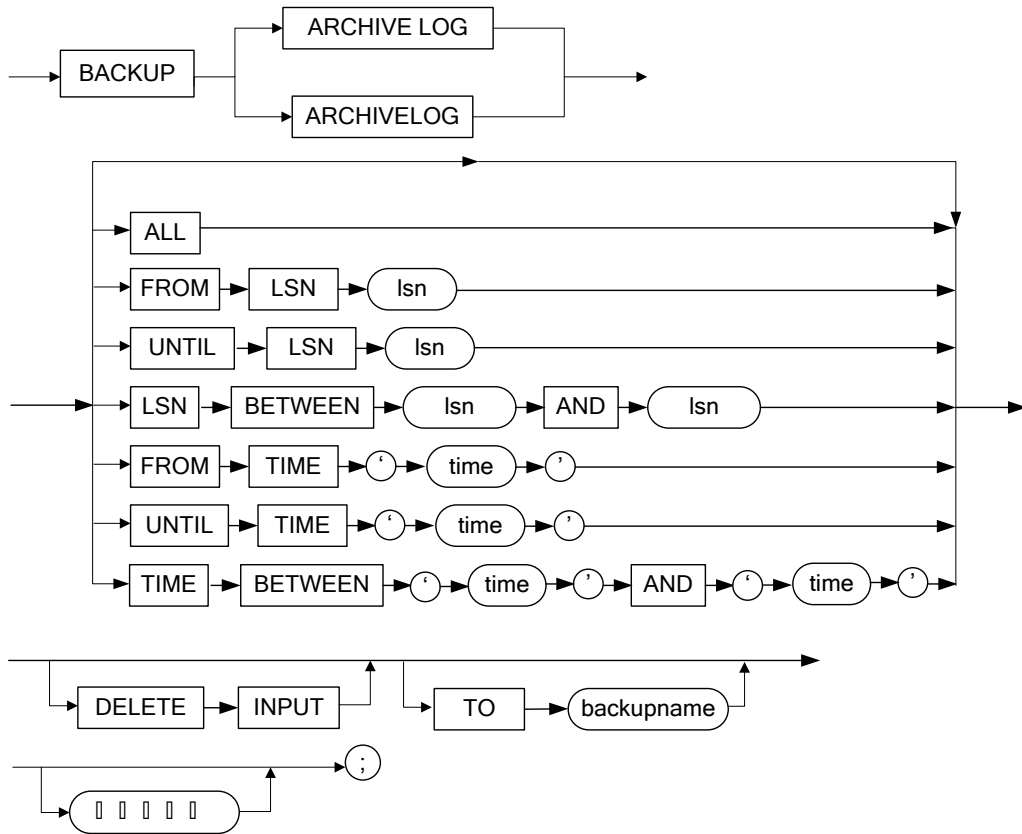
TRACE FILE: 指定生成的 TRACE 文件。启用 TRACE, 但不指定 TRACE FILE 时, 默认在 DM 数据库系统的 log 目录下生成 DM\_SBTTRACE\_年月.LOG 文件; 若使用相对路径, 则生成在执行码同级目录下。若用户指定, 则指定的文件不能为已经存在的文件, 否则报错; 也不可以为 ASM 文件。

TRACE LEVEL: 有效值 1、2, 默认为 1 表示不启用 TRACE, 此时若指定了 TRACE FILE, 会生成 TRACE 文件, 但不写入 TRACE 信息; 为 2 启用 TRACE 并写入 TRACE 相关内容。

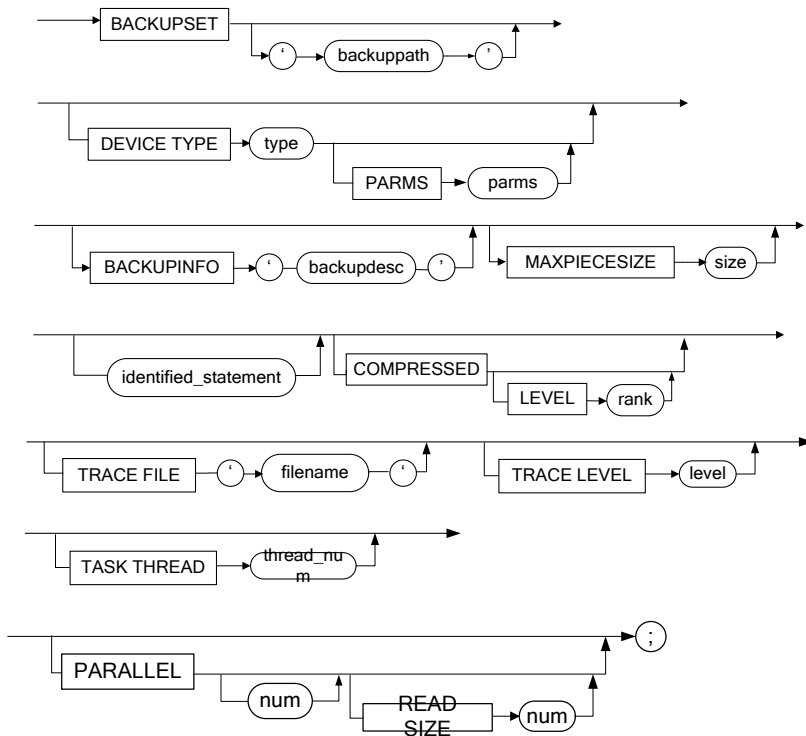
TASK THREAD: 备份过程中数据处理过程线程的个数, 取值范围 0~64, 默认为 4。若指定为 0, 则调整为 1; 若指定超过当前系统主机核数, 则调整为当前主机核数。线程数 (TASK THREAD) \* 并行数 (PARALLEL) 不得超过 512。

PARALLEL: 指定并行备份的<并行数>和 READ SIZE<拆分块大小> 请参考 [3.2.2.1.1 概述](#)。

图例:

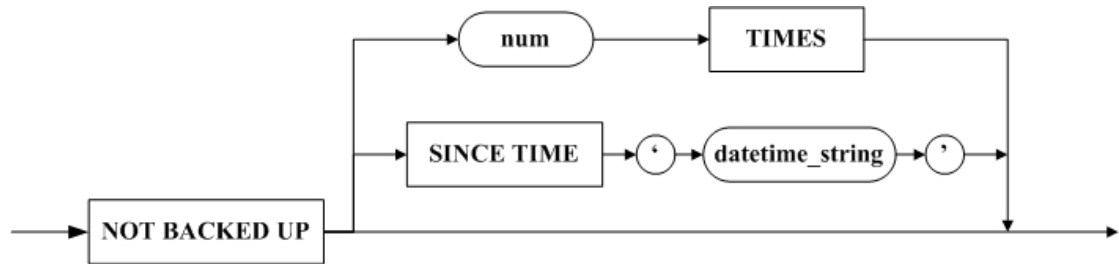


<备份集子句>:



identified\_statement: 请参考 3.2.2.1.1 [概述](#) 中的数据库备份图例

notBackedUpSpec:



### 3.2.2.4.2 设置备份选项

归档备份常用的备份选项有设置备份名、设置备份集路径、指定介质参数、添加备份描述等，设置方式同数据库备份相同，具体参考 3.2.2.1.2 [设置备份选项](#)。

### 3.2.2.4.3 备份归档

归档备份拷贝指定归档目录下的所有的归档文件到备份集中，并记录各个归档文件的属性，文件大小，LSN 区间等。归档备份不存在增量备份之说。当需要保存库的归档时，可以使用归档备份。

完整的备份归档步骤如下：

- 1) 配置归档，参考 3.1.2 [归档配置](#)。
- 2) 数据库处于 OPEN 或者 MOUNT 状态。
- 3) DISql 中输入备份数据库语句。

例如，备份归档日志，通过 LSN BETWEEN ... AND ... 来指定起始和截至 LSN。

首先，确定 LSN 范围。

```
SQL>select ARCH_LSN, CLSN, PATH from V$ARCH_FILE;
```

查询结果为：

ARCH_LSN	CLSN	PATH
38758	40301	d:\arch\ARCHIVE_LOCAL1_20180727105417748.log
40302	40303	d:\arch\ARCHIVE_LOCAL1_20180727111411079.log
40304	40305	d:\arch\ARCHIVE_LOCAL1_20180727111513679.log
.....		
50412	50413	d:\arch\ARCHIVE_LOCAL1_20180727171801098.log
<b>50414</b>	<b>50478</b>	d:\arch\ARCHIVE_LOCAL1_20180727171849712.log

.....

通过查询结果选出备份的起始 LSN 和截至 LSN。比如 50414 50478

其次，备份归档。

```
SQL>BACKUP ARCHIVELOGLSN BETWEEN 50414 AND 50478 BACKUPSET
'/home/dm_bak/arch_bak_time_14-78';
```

### 3.2.3 数据备份高级主题

本节介绍高级的联机备份过程，主要内容包括：

- 加密备份
- 设置跟踪日志文件

DM 的数据库、表空间、表备份和归档备份均支持创建加密备份和备份时设置跟踪日志文件，本节仅以数据库为例说明这两种类型备份的创建。

#### 加密备份

DM 提供加密备份的方式保护用户的备份集，没有权限的用户无法访问加密的备份集。

备份语句中通过指定 IDENTIFIED BY...WITH ENCRYPTION...ENCRYPT WITH... 执行加密备份。其中，IDENTIFIED BY 子句指定加密密码，密码长度为 9 到 48 个字节，若密码长度不符合要求会报错；WITH ENCRYPTION 子句指定加密类型，加密类型分为简单加密和复杂加密，简单加密是对备份文件设置口令，但文件内容仍以明文存储，复杂加密则对备份文件进行完全的加密，备份文件以密文方式存储，用户可根据备份数据的重要程度选择加密类型；ENCRYPT WITH 子句指定加密算法，不同加密算法具体见参数说明，也可通过“SELECT \* FROM V\$CIPHERS”语句查询 DM 支持的加密算法，其中算法 MD5 和 SHA1 不能在此处使用。默认使用的加密算法为 AES256\_CFB。

加密备份过程中 IDENTIFIED BY 子句必须指定，子句 WITH ENCRYPTION 和子句 ENCRYPT WITH 可不指定，此时 WITH ENCRYPTION 默认值为 1，ENCRYPT WITH 默认值为 AES256\_CFB。例如，以下两种加密备份语句都是合法的：

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_for_encrypt' IDENTIFIED BY
'cdb546789';

SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_for_encrypt' IDENTIFIED BY
'cdb546789' ENCRYPT WITH RC4;
```

若指定了加密密码，但加密类型 WITH ENCRYPTION 指定为 0，则为非加密备份，如下所示：

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_for_encrypt' IDENTIFIED BY
"cdb546789" WITH ENCRYPTION 0;
```

下面以数据库完全备份为例，创建加密密码为“cdb546789”，加密算法为“rc4”的复杂类型数据库加密备份，完整步骤如下：

- 1) 配置归档，参考 3.1.2 [归档配置](#)。
- 2) 保证数据库处于 OPEN 状态。
- 3) 备份数据库，输入以下语句：

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_for_encrypt_01' IDENTIFIED
BY "cdb546789" WITH ENCRYPTION 2 ENCRYPT WITH RC4;
```

对于增量备份加密，如果基备份存在加密，则使用的加密算法和加密密码必须与基备份中一致；如果基备份未进行加密处理，则对增量备份使用的加密密码和算法没有特殊要求。

### 设置跟踪日志文件

DM 备份时支持设置跟踪日志文件，日志记录了 SBT 接口的调用过程，用户通过查看日志可跟踪备份还原过程。

与生成跟踪日志文件相关的参数有两个：TRACE FILE 和 TRACE LEVEL。TRACE FILE 用于指定生成的跟踪日志文件路径，TRACE LEVEL 表示是否启用 TRACE。TRACE LEVEL 有效值包括 1 和 2。1 表示不启用 TRACE 功能，2 表示启用，系统默认值为 1。

指定参数 TRACE FILE 但 TRACE LEVEL 值设置为 1 即不启用 TRACE 功能，会生成 TRACE 文件，但不会写入 TRACE 信息。如下所示：

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_for_trace' TRACE
FILE '/home/dm_log/db_trace.log' TRACE LEVEL 1;
```

TRACE LEVEL 值设置为 2 即启用 TRACE 功能，但若 TRACE FILE 没有指定，系统默认在执行码路径的 log 目录下生成 DM\_SBTTRACE\_年月.log 文件。如下所示：

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_for_trace' TRACE LEVEL 2;
```

若 TRACE FILE 使用相对路径，日志文件生成在执行码同级目录下。

以数据库完全备份为例，为备份设置跟踪日志文件的操作步骤如下：

- 1) 配置归档，参考 3.1.2 [归档配置](#)。
- 2) 保证数据库处于 OPEN 状态。

3) 备份数据库, 输入以下语句:

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_for_trac_01' TRACE
FILE '/home/dm_log/db_bak_trace.log' TRACE LEVEL 2;
```

如果指定的 TRACE 文件已存在, 不会覆盖已存在的文件而是在已有文件基础上继续记录日志。

### 3.2.4 管理备份

本章节主要介绍如何使用 DISql 工具管理数据库备份、表空间备份、表备份, 以及归档备份。本节内容主要包括:

- 概述
- 备份目录管理
- 备份集管理
- 备份信息查看

#### 3.2.4.1 概述

管理备份一个重要的目的是删除不再需要的备份, DM 没有提供自动删除过期备份的功能, 删除备份需要手动执行。

备份管理相关系统过程与函数总结如下:

- SF\_BAKSET\_BACKUP\_DIR\_ADD: 添加备份目录。
- SF\_BAKSET\_BACKUP\_DIR\_REMOVE: 指定删除内存中的备份目录。
- SF\_BAKSET\_BACKUP\_DIR\_REMOVE\_ALL: 删除内存中全部的备份目录。
- SF\_BAKSET\_CHECK: 对备份集进行校验。
- SF\_BAKSET\_REMOVE: 删除指定设备类型和指定备份集目录的备份集。
- SF\_BAKSET\_REMOVE\_BATCH: 批量删除满足指定条件的所有备份集。
- SP\_DB\_BAKSET\_REMOVE\_BATCH: 批量删除指定时间之前的数据库备份集。
- SP\_TS\_BAKSET\_REMOVE\_BATCH: 批量删除指定表空间对象及指定时间之前的表空间备份集。
- SP\_TAB\_BAKSET\_REMOVE\_BATCH: 批量删除指定表对象及指定时间之前的表备份集。



- `SP_ARCH_BAKSET_REMOVE_BATCH`: 批量删除指定条件的归档备份集。

备份管理相关动态视图总结如下:

- `V$BACKUPSET`: 显示备份集基本信息。
- `V$BACKUPSET_DBINFO`: 显示备份集的数据库相关信息。
- `V$BACKUPSET_DBF`: 显示备份集中数据文件的相关信息。
- `V$BACKUPSET_ARCH`: 显示备份集的归档信息。
- `V$BACKUPSET_BKP`: 显示备份集的备份片信息。
- `V$BACKUPSET_SEARCH_DIRS`: 显示备份集搜索目录。
- `V$BACKUPSET_TABLE`: 显示表备份集中备份表信息。
- `V$BACKUPSET_SUBS`: 显示并行备份中生成的子备份集信息。



**`SF_BAKSET_BACKUP_DIR_ADD`** 添加备份目录仅对当前会话有效。调用删除

**注意:** 备份等函数或查看动态视图时要先调用 `SF_BAKSET_BACKUP_DIR_ADD` 添加备份目录, 否则仅搜索默认备份路径下的备份集。

各个函数和动态视图的使用, 下文将进行详细介绍。

### 3.2.4.2 备份目录管理

这里的备份目录是指备份集搜索目录, 这些目录被记录在内存中, 当执行动态视图(参见 [3.2.4.4 备份信息查看](#))或批量删除备份集时, 均会从这些指定目录中先搜索所有备份集信息。

本节主要包括:

- `SF_BAKSET_BACKUP_DIR_ADD`
- `SF_BAKSET_BACKUP_DIR_REMOVE`
- `SF_BAKSET_BACKUP_DIR_REMOVE_ALL`

#### ■ `SF_BAKSET_BACKUP_DIR_ADD` 函数

添加备份目录。若添加目录已经存在或者为库默认备份路径, 则认为已经存在, 不添加, 但也不报错。

定义:

---

```
INT SF_BAKSET_BACKUP_DIR_ADD(  
    device_type varchar,
```

---

---

```

        backup_dir varchar(256)
    )

```

---

参数说明：

**device\_type**: 待添加的备份目录对应存储介质类型，DISK 或者 TAPE。目前，无论指定介质类型为 DISK 或者 TAPE，都会同时搜索两种类型的备份集。

**backup\_dir**: 待添加的备份目录。

返回值：

1: 目录添加成功；其它情况下报错。

举例说明：

```
SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK','/home/dm_bak');
```

执行结果

```
1
```

#### ■ SF\_BAKSET\_BACKUP\_DIR\_REMOVE 函数

删除备份目录。若删除目录为库默认备份路径，不进行删除，认为删除失败。若指定目录存在于记录的合法目录中，则删除；不存在或者为空则跳过，正常返回。

定义：

---

```

INT SF_BAKSET_BACKUP_DIR_REMOVE (
    device_type varchar,
    backup_dir varchar(256)
)

```

---

参数说明：

**device\_type**: 待删除的备份目录对应存储介质类型。待删除的备份目录对应存储介质类型，DISK 或者 TAPE。

**backup\_dir**: 待删除的备份目录。

返回值：

1: 目录删除成功、目录不存在或者目录为空；0: 目录为库默认备份路径；其他情况报错。

举例说明：

```
SQL>SELECT SF_BAKSET_BACKUP_DIR_REMOVE('DISK','/home/dm_bak');
```

执行结果

1

#### ■ SF\_BAKSET\_BACKUP\_DIR\_REMOVE\_ALL 函数

清理全部备份目录，默认备份目录除外。

定义：

---

```
INT SF_BAKSET_BACKUP_DIR_REMOVE_ALL ( )
```

---

返回值：

1：目录全部清理成功；其它情况下报错。

举例说明：

```
SQL>SELECT SF_BAKSET_BACKUP_DIR_REMOVE_ALL( ) ;
```

执行结果

1

### 3.2.4.3 备份集校验与删除

本节介绍备份管理中最重要功能，备份集校验和备份集删除。单个备份集删除时并行备份中地子备份集不允许单独删除；在给定备份集搜集目录中发现存在引用删除备份集作为基备份的需要执行级联删除，默认报错。批量删除备份集时，跳过收集到的单独的子备份集。

主要内容如下：

- SF\_BAKSET\_CHECK
- SF\_BAKSET\_REMOVE
- SF\_BAKSET\_REMOVE\_BATCH
- SP\_DB\_BAKSET\_REMOVE\_BATCH
- SP\_TS\_BAKSET\_REMOVE\_BATCH
- SP\_TAB\_BAKSET\_REMOVE\_BATCH
- SP\_ARCH\_BAKSET\_REMOVE\_BATCH

#### ■ SF\_BAKSET\_CHECK 函数

对备份集进行校验。

定义：

---

```
INT SF_BAKSET_CHECK(  
    device_type varchar,  
    bakset_path varchar(256)  
)
```

---

参数说明:

device\_type: 设备类型, disk 或 tape。

bakset\_path: 待校验的备份集目录。

返回值:

1: 备份集目录存在且合法; 否则报错。

举例说明:

```
SQL>BACKUP DATABASE FULL BACKUPSET '/home/dm_bak/db_bak_for_check';  
SQL> SELECT SF_BAKSET_CHECK('DISK','/home/dm_bak/ db_bak_for_check');
```

执行结果

```
1
```

### ■ SF\_BAKSET\_REMOVE 函数

删除指定设备类型和指定备份集目录的备份集。一次只检查一个合法 .meta 文件, 然后删除对应备份集; 若存在非法或非正常备份的 .meta 文件, 则报错或直接返回, 不会接着检查下一个 .meta 文件; 若同一个备份集下还存在其它备份文件或备份集, 则只删除备份文件, 不会删除整个备份集。

定义:

---

```
INT SF_BAKSET_REMOVE (  
    device_type varchar,  
    backsetpath varchar(256),  
    option integer  
)
```

---

参数说明:

device\_type: 设备类型, disk 或 tape。

backsetpath: 待删除的备份集目录。

Option: 删除备份集选项, 0 默认删除, 1 级联删除。可选参数。并行备份集中子备份

集不允许单独删除。目标备份集被其他备份集引用为基备份的，默认删除，报错；级联删除情况下，会递归将相关的增量备份也删除。

返回值：

1：备份集目录删除成功，其它情况下报错。

举例说明：

```
SQL>BACKUP DATABASE FULL BACKUPSET '/home/dm_bak/db_bak_for_remove';

SQL>BACKUP DATABASE INCREMENT BACKUPSET '/home/dm_bak/db_bak_for_remove_incr';

SQL> SELECT SF_BAKSET_REMOVE('DISK','/home/dm_bak/db_bak_for_remove');
```

执行结果

```
[-8202]:[/home/dm_bak/db_bak_for_remove_incr]的基备份，不能删除.
```

```
SQL> SELECT SF_BAKSET_REMOVE('DISK','/home/dm_bak/db_bak_for_remove',1);
```

执行结果

```
1
```

### ■ SF\_BAKSET\_REMOVE\_BATCH 函数

批量删除满足指定条件的所有备份集。

定义：

```
INT SF_BAKSET_REMOVE_BATCH (

    device_type varchar,

    end_time     datetime,

    range        int,

    obj_name     varchar(257)

)
```

参数说明：

device\_type：设备类型，disk 或 tape。指定 NULL，则忽略存储设备的区分。

end\_time：删除备份集生成的结束时间，仅删除 end\_time 之前的备份集，必须指定。

range：指定删除备份的级别。1 代表库级，2 代表表空间级，3 代表表级，4 代表归档备份。若指定 NULL，则忽略备份集备份级别的区分。

**obj\_name:** 待删除备份集中备份对象的名称，仅表空间级和表级有效。若为表级备份删除，则需指定完整的表名（模式.表名），否则，将认为删除会话当前模式下的表备份。若指定为 NULL，则忽略备份集中备份对象名称区分。

返回值：

1: 备份集目录删除成功，其它情况下报错。

举例说明：

```
SQL>BACKUP DATABASE FULL BACKUPSET '/home/dm_bak/db_bak_for_remove';
SQL>BACKUP TABLESPACEMAIN FULL BACKUPSET '/home/dm_bak/ts_bak_for_remove';
SQL>SELECT SF_BAKSET_REMOVE_BATCH ('DISK', now(), NULL, NULL);
```

执行结果

1

### ■ SP\_DB\_BAKSET\_REMOVE\_BATCH 过程

批量删除指定时间之前的数据库备份集。使用该方法前，需要先使用 SF\_BAKSET\_BACKUP\_DIR\_ADD 添加将要删除的备份集目录，否则只删除默认备份路径下的备份集。

定义：

```
SP_DB_BAKSET_REMOVE_BATCH (
    device_type varchar,
    end_time     datetime
)
```

参数说明：

**device\_type:** 设备类型，disk 或 tape。指定 NULL，则忽略存储设备的区分。

**end\_time:** 删除备份集生成的结束时间，仅删除 end\_time 之前的备份集，必须指定。

举例说明：

```
SQL>BACKUP DATABASE FULL BACKUPSET '/home/dm_bak/db_bak_for_batch_del';
SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK', '/home/dm_bak');
SQL>CALL SP_DB_BAKSET_REMOVE_BATCH('DISK', NOW());
```

### ■ SP\_TS\_BAKSET\_REMOVE\_BATCH 过程

批量删除指定表空间对象及指定时间之前的表空间备份集。使用该方法前，需要先使用 SF\_BAKSET\_BACKUP\_DIR\_ADD 添加将要删除的备份集目录，否则只删除默认备份路径下的备份集。

定义：

---

```
SP_TS_BAKSET_REMOVE_BATCH (  
    device_type    varchar,  
    end_time       datetime,  
    ts_name        varchar(128)  
)
```

---

参数说明：

device\_type: 设备类型，disk 或 tape。指定 NULL，则忽略存储设备的区分。

end\_time: 删除备份集生成的结束时间，仅删除 end\_time 之前的备份集，必须指定。

ts\_name: 表空间名，若未指定，则认为删除所有满足条件的表空间备份集。

举例说明：

```
SQL>BACKUP TABLESPACE MAIN BACKUPSET '/home/dm_bak/ts_bak_for_batch_del';  
SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK','/home/dm_bak');  
SQL>CALL SP_TS_BAKSET_REMOVE_BATCH('DISK',NOW(),'MAIN');
```

### ■ SP\_TAB\_BAKSET\_REMOVE\_BATCH 过程

批量删除指定表对象及指定时间之前的表备份集。使用该方法前，需要先使用 SF\_BAKSET\_BACKUP\_DIR\_ADD 添加将要删除的备份集目录，否则只删除默认备份路径下的备份集。

定义：

---

```
SP_TAB_BAKSET_REMOVE_BATCH (  
    device_type    varchar,  
    end_time       datetime,  
    sch_name       varchar(128),  
    tab_name       varchar(128)  
)
```

---

参数说明：

**device\_type**: 设备类型, disk 或 tape。指定 NULL, 则忽略存储设备的区分。

**end\_time**: 删除备份集生成的结束时间, 仅删除 **end\_time** 之前的备份集, 必须指定。

**sch\_name**: 表所属的模式名。

**tab\_name**: 表名, 只要模式名和表名有一个指定, 就认为需要匹配目标; 若均指定为 NULL, 则认为删除满足条件的所有表备份。

举例说明：

```
SQL>CREATE TABLE TAB_FOR_BATCH_DEL(C1 INT);

SQL>BACKUP TABLE TAB_FOR_BATCH_DEL BACKUPSET '/home/dm_bak/tab_bak_for_batch_del';

SQL> SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK','/home/dm_bak');

SQL> CALL SP_TAB_BAKSET_REMOVE_BATCH('DISK',NOW(),'SYSDBA','TAB_FOR_BATCH_DEL');
```

### ■ SP\_ARCH\_BAKSET\_REMOVE\_BATCH 过程

批量删除指定时间之前的归档备份集。使用该方法前, 需要先使用

SF\_BAKSET\_BACKUP\_DIR\_ADD 添加将要删除的备份集目录, 否则只删除默认备份路径下的备份集。

定义：

---

```
SP_ARCH_BAKSET_REMOVE_BATCH (

    device_type varchar,

    end_time     datetime

)
```

---

参数说明：

**device\_type**: 设备类型, disk 或 tape。指定 NULL, 则忽略存储设备的区分。

**end\_time**: 删除备份集生成的结束时间, 仅删除 **end\_time** 之前的备份集, 必须指定。

举例说明：

```
SQL>BACKUP ARCHIVELOG BACKUPSET '/home/dm_bak/arch_bak_for_batch_del';

SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK','/home/dm_bak');

SQL>CALL SP_ARCH_BAKSET_REMOVE_BATCH('DISK', NOW());
```



### 3.2.4.4 备份信息查看

DM 提供了一系列动态视图供用户查看备份集相关信息，在查看之前应先使用 SF\_BAKSET\_BACKUP\_DIR\_ADD 添加备份集目录，否则只显示默认备份路径下的备份集信息，使用方法如下例所示：

```
SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK', '/home/dm_bak');
```

下面逐个介绍 DM 提供的备份相关动态视图。

#### ■ V\$BACKUPSET

V\$BACKUPSET 显示备份集基本信息。

序号	列	数据类型	说明
1	DEVICE_TYPE	VARCHAR(10)	备份集存储介质类型
2	BACKUP_ID	INTEGER	备份 ID
3	PARENT_ID	INTEGER	并行备份的子备份集所属备份的 ID
4	BACKUP_NAME	VARCHAR(512)	备份名
5	BACKUP_PATH	VARCHAR(1024)	备份路径
6	TYPE	INTEGER	0：基备份，1：增量备份，2：表备份，3：归档备份
7	LEVEL	INTEGER	是否脱机备份。0：联机备份，1：脱机备份
8	RANGE#	INTEGER	1：库备份，2：表空间备份，3：表级备份，4：归档备份
9	OBJECT_NAME	VARCHAR(1025)	对象名：数据库名、表空间名或者表名
10	OBJECT_ID	INTEGER	对象 ID，表备份时无效
11	BASE_NAME	VARCHAR(512)	基备份名，表备份时无效
12	BACKUP_TIME	DATETIME(6)	备份时间
13	DESC#	VARCHAR(1024)	备份描述信息
14	ENCRYPT_TYPE	INTEGER	加密类型
15	COMPRESS_LEVEL	INTEGER	压缩级别
16	WITHOUT_LOG	INTEGER	联机数据库备份是否备份日志，表备份时无效

## 备份与还原

17	USE_PWR	INTEGER	增量备份过程中是否使用 PWR 优化, 均不使用, 保留仅为了兼容
18	PKG_SIZE	INTEGER	数据包大写标志, 内部实现
19	BEGIN_LSN	BIGINT	备份的起始 LSN 值, 表备份时无效
20	END_LSN	BIGINT	结束备份的 LSN 值, 表备份时无效
21	BKP_NUM	INTEGER	备份片个数, 即备份集中 .bak 文件个数
22	DBF_NUM	INTEGER	备份集中包含的数据库\表空间数据文件个数, 表备份时无效
23	PARALLEL_NUM	INTEGER	并行备份的并行数, 0 或者 1 为非并行备份集
24	BAKSET_TYPE	INTEGER	备份集类型。0 普通备份集 (NORMAL), 1DDL_CLONE 备份集 (DDL_CLONE), 2 联机拷贝备份集 (COPY)
25	MPP_FLAG	INTEGER	MPP 库备份标识, 0 不是, 1 是
26	MIN_TRX_START_LSN	BIGINT	该参数未使用, 始终为 0
27	MIN_EXEC_VER	INTEGER	备份集适用的最小执行码的版本号, 转换为 16 进制匹配版本号
28	CUMULATIVE	INTEGER	增量备份时, 是否为累积增量备份。1 是, 0 否
29	MIN_DCT_VER	INTEGER	备份集适用的最小字典版本号

下面以创建数据库备份为例, 查看备份集的介质类型、备份路径、备份类型等基本信息:

```
SQL>BACKUP DATABASE FULL BACKUPSET '/home/dm_bak/db_bak_for_info';

SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK', '/home/dm_bak');

SQL>SELECT DEVICE_TYPE, BACKUP_PATH, TYPE, RANGE# FROM V$BACKUPSET;
```

查询结果:

LINEID	DEVICE_TYPE	BACKUP_PATH	TYPE	RANGE#
1	DISK	/home/dm_bak/db_bak_for_info	0	1

### ■ V\$BACKUPSET\_DBINFO

V\$BACKUPSET\_DBINFO 显示备份集的数据库相关信息。

## 备份与还原

序号	列	数据类型	说明
1	DEVICE_TYPE	VARCHAR(10)	备份集存储介质类型
2	BACKUP_ID	INTEGER	备份 ID
3	BACKUP_NAME	VARCHAR(512)	备份名
4	BACKUP_PATH	VARCHAR(1024)	备份路径
5	EXTENT_SIZE	INTEGER	数据文件使用的簇大小
6	PAGE_SIZE	INTEGER	页大小
7	LOG_PAGE_SIZE	INTEGER	日志文件页大小
8	CASE_SENSITIVE	INTEGER	大小写敏感标志
9	DB_MAGIC	INTEGER	数据库的 magic
10	PMNT_MAGIC	INTEGER	永久魔数 (permenant_magic)
11	UNICODE_FLAG	INTEGER	unicode 标志
12	DB_VERSION	INTEGER	数据库版本
13	GLOBAL_VERSION	VARCHAR(512)	数据库全局版本信息
14	ENABLE_POLICY	INTEGER	安全策略
15	ARCH_FLAG	INTEGER	归档是否打开的标志
16	DSC_NODE	INTEGER	高性能集群的节点数目
17	PAGE_CHECK	INTEGER	数据页校验配置
18	RLOG_ENCRYPT	INTEGER	归档日志是否加密
19	EX_CIPHER_NAME	VARCHAR(512)	外部加密算法名称
20	EX_CIPHER_ID	INTEGER	外部加密算法名称对应的 ID
21	EX_HASH_NAME	VARCHAR(512)	外部 HASH 算法名称
22	EX_HASH_ID	INTEGER	外部 HASH 算法名称对应的 ID
23	LENGTH_IN_CHAR	INTEGER	VARCHAR 类型长度是否以字符为单位
24	USE_NEW_HASH	INTEGER	是否使用改进的字符类型 HASH 算法
25	BLANK_PAD_MODE	INTEGER	数据库空格填充模式
26	SRC_DB_MAGIC	INTEGER	源库 DB_MAGIC

表还原时要求目标库的特定建库参数要与源库一致，如页大小、簇大小，通过查看备份文件的数据库信息可确定目标库需要设置哪些建库参数。下面以创建表备份为例，查看备份集的页大小、簇大小、大小写是否敏感、UNICODE\_FLAG 等数据库信息：

```
SQL>BACKUP TABLETAB_01 BACKUPSET '/home/dm_bak/tab_bak_01';

SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK', '/home/dm_bak');

SQL>SELECT    BACKUP_PATH,    PAGE_SIZE,    EXTENT_SIZE,    CASE_SENSITIVE    FROM
V$BACKUPSET_DBINFO WHERE BACKUP_PATH='/home/dm_bak/tab_bak_01';
```

查询结果：

LINEID	BACKUP_PATH	PAGE_SIZE	EXTENT_SIZE	CASE_SENSITIVE
1	/home/dm_bak/db_bak_for_dbinfo	8192	16	1

#### ■ V\$BACKUPSET\_DBF

显示备份集中数据文件的相关信息，表备份时无效。

序号	列	数据类型	说明
1	DEVICE_TYPE	VARCHAR(10)	备份集存储介质类型
2	BACKUP_ID	INTEGER	备份 ID
3	BACKUPNAME	VARCHAR(512)	备份名
4	BACKUPPATH	VARCHAR(1024)	备份路径
5	FILE_SEQ	INTEGER	备份的数据文件序号
6	TS_ID	INTEGER	表空间 ID
7	FILE_ID	INTEGER	数据文件 ID
8	TS_STATE	INTEGER	表空间状态
9	TS_NAME	VARCHAR(512)	表空间名
10	FILE_NAME	VARCHAR(1024)	包含完整路径的数据文件名
11	MIRROR_PATH	VARCHAR(1024)	镜像文件路径
12	FILE_LEN	BIGINT	数据文件占用的字节大小
13	MAX_LIMIT_SIZE	INTEGER	文件最大大小，以 M 为单位
14	AUTO_EXTEND	INTEGER	是否支持自动扩展：1 支持，0 不支持

## 备份与还原

15	NEXT_SIZE	INTEGER	文件每次扩展大小，以 M 为单位
16	START_BKP_SEQ	INTEGER	起始备份片编号
17	START_BKP_OFF	BIGINT	起始备份片偏移
18	END_BKP_SEQ	INTEGER	结束备份片编号
19	END_BKP_OFF	BIGINT	结束备份片偏移

数据库和表空间备份集中记录了备份的数据文件具体信息，如果了解备份集中包含了哪些数据文件且这些数据文件有什么属性，可通过查询 V\$BACKUPSET\_DBF 实现。下面以表空间备份为例，查看备份集中的数据文件信息。

```
SQL>CREATE TABLESPACE TS_FOR_DBF DATAFILE'TS_FOR_DBF_01.DBF' SIZE 128;

SQL>ALTER TABLESPACE TS_FOR_DBF ADD DATAFILE'TS_FOR_DBF_02.DBF' SIZE 128;

SQL>BACKUP TABLESPACE TS_FOR_DBF BACKUPSET '/home/dm_bak/ts_bak_for_dbf';

SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK', '/home/dm_bak');

SQL>SELECT FILE_SEQ, TS_ID, FILE_ID, TS_NAME, FILE_NAME FROM V$BACKUPSET_DBF
WHERE BACKUPPATH = '/home/dm_bak/ts_bak_for_dbf';
```

查询结果：

LINEID	FILE_SEQ	TS_ID	FILE_ID	TS_NAME	FILE_NAME
-----					
1	1	5	0	TS_FOR_DBF	/home/xm/DAMENG/TS_FOR_DBF_01.DBF
2	2	5	1	TS_FOR_DBF	/home/xm/DAMENG/TS_FOR_DBF_02.DBF

### ■ V\$BACKUPSET\_ARCH

显示备份集中归档文件的信息，且仅归档备份才会有数据。

序号	列	数据类型	说明
1	DEVICE_TYPE	VARCHAR(10)	备份集存储介质类型
2	BACKUP_ID	INTEGER	备份 ID
3	BACKUPNAME	VARCHAR(512)	备份名
4	BACKUPPATH	VARCHAR(1024)	备份路径
5	FILE_SEQ	INTEGER	备份的数据文件序号
6	FILE_NAME	VARCHAR(1024)	归档文件路径

## 备份与还原

7	FILE_LEN	BIGINT	归档文件大小
8	DSC_SEQNO	INTEGER	DSC 节点号
9	BEGIN_SEQNO	BIGINT	归档起始 PKG_SEQNO
10	BEGIN_LSN	BIGINT	归档起始 LSN 值
11	END_SEQNO	BIGINT	归档结束 PKG_SEQNO
12	END_LSN	BIGINT	归档结束 LSN 值
13	CREATE_TIME	DATETIME(6)	归档文件的创建时间
14	CLOSE_TIME	DATETIME(6)	归档文件的关闭时间
15	START_BKP_SEQ	INTEGER	起始备份片编号
16	START_BKP_OFF	BIGINT	起始备份片偏移
17	END_BKP_SEQ	INTEGER	结束备份片编号
18	END_BKP_OFF	BIGINT	结束备份片偏移

备份集日志信息可以是联机库备份备份开始到备份结束这段时间数据库产生的日志,也可以是归档备份中备份的归档信息。接下来以数据库备份为例查询备份集日志信息。

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_for_arch';SQL>SELECT
SF_BAKSET_BACKUP_DIR_ADD('DISK', '/home/dm_bak');

SQL>SELECT BACKUPPATH, FILE_SEQ, BEGIN_LSN, END_LSN FROM V$BACKUPSET_ARCH
WHERE BACKUPPATH='/home/dm_bak/db_bak_for_arch';
```

查询结果:

LINEID	BACKUPPATH	FILE_SEQ	BEGIN_LSN	END_LSN
1	/home/dm_bak/db_bak_for_arch6		280037	281471

### ■ V\$BACKUPSET\_BKP

显示备份集的备份片信息。

序号	列	数据类型	说明
1	DEVICE_TYPE	VARCHAR(10)	备份集存储介质类型
2	BACKUP_ID	INTEGER	备份 ID

## 备份与还原

3	BACKUPNAME	VARCHAR(512)	备份名
4	BACKUPPATH	VARCHAR(1024)	备份路径
5	BKP_NTH	INTEGER	备份片文件编号
6	FILE_NAME	VARCHAR(1024)	备份文件名
7	BKP_LEN	BIGINT	备份片长度

以数据库备份为例，查看备份集中的备份片信息。

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_for_bkp';

SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK', '/home/dm_bak');

SQL>SELECT BACKUPPATH, BKP_NTH, FILE_NAME, BKP_LEN FROM V$BACKUPSET_BKP WHERE
BACKUPPATH='/home/dm_bak/db_bak_for_bkp';
```

查询结果：

LINEID	BACKUPPATH	BKP_NTH	FILE_NAME	BKP_LEN
1	/home/dm_bak/db_bak_for_bkp	0	db_bak_for_bkp.bak	15378944
2	/home/dm_bak/db_bak_for_bkp	1	db_bak_for_bkp_1.bak	9216

### ■ V\$BACKUPSET\_SEARCH\_DIRS

显示备份集搜索目录。

序号	列	数据类型	说明
1	DIR	VARCHAR(1024)	备份集搜索目录

查询 V\$BACKUPSET\_SEARCH\_DIRS 显示当前会话已添加的备份目录，即备份集搜索目录。若用户没有添加备份目录，那么仅显示默认的备份目录。

```
SQL>SELECT SF_BAKSET_BACKUP_DIR_REMOVE_ALL();

SQL>SELECT * FROM V$BACKUPSET_SEARCH_DIRS;
```

查询结果：

LINEID	DIR
1	/home/xm/DAMENG/bak

如果添加备份目录，查询结果包括默认备份目录和用户添加的备份目录。

```
SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK', '/home/dm_bak');
```

```
SQL>SELECT * FROM V$BACKUPSET_SEARCH_DIRS;
```

查询结果:

```
LINEID      DIR
-----
1          /home/dm_bak
2          /home/xm/DAMENG/bak
```

#### ■ V\$BACKUPSET\_TABLE

显示表备份集中备份表信息，仅表备份有效。

序号	列	数据类型	说明
1	DEVICE_TYPE	VARCHAR(10)	备份集存储介质类型
2	BACKUP_ID	INTEGER	备份 ID
3	BACKUPNAME	VARCHAR(512)	备份名
4	BACKUPPATH	VARCHAR(1024)	备份路径
5	SCHEMANAME	VARCHAR(512)	备份表所属的模式名
6	USERNAME	VARCHAR(512)	执行表备份的用户名
7	TSNAME	VARCHAR(512)	备份表存储的表空间名
8	TABlename	VARCHAR(512)	备份表名
9	TABLETYPE	INTEGER	表类型
10	INITSQL	VARCHAR(4096)	完整建表语句，忽略引用约束 (语句可能会被截断)
11	DCONS_SQL	VARCHAR(4096)	备份表中被禁用约束的创建语句(语句可能会被截断，DMRMAN 可查看完整语句)
12	DIDX_SQL	VARCHAR(4096)	备份表中无效二级索引的创建语句(语句可能会被截断，DMRMAN 可查看完整语句)
13	BIDX_NUM	INT	备份集中备份的二级索引个数
14	META_VERSION	INT	当前表备份的元信息的版本号

下面创建表备份并查看备份中备份名、备份路径、表名等信息。

```
SQL>CREATE TABLE TAB_FOR_INFO(C1 INT);
```



```
SQL>BACKUP TABLE TAB_FOR_INFO TO TAB_BAK_FOR_INFO BACKUPSET
'/home/dm_bak/tab_bak_for_info';

SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK', '/home/dm_bak');

SQL>SELECT BACKUPNAME, TABLENAME FROM V$BACKUPSET_TABLE WHERE
BACKUPPATH='/home/dm_bak/tab_bak_for_info';
```

查询结果:

LINEID	BACKUPNAME	TABLENAME
1	TAB_BAK_FOR_INFO	TAB_FOR_INFO

#### ■ V\$BACKUPSET\_SUBS

显示并行备份中生成的子备份集信息。

序号	列	数据类型	说明
1	DEVICE_TYPE	VARCHAR(10)	备份集存储介质类型
2	BACKUPNAME	VARCHAR(512)	备份名
3	BACKUP_ID	INTEGER	备份 ID
4	PARENT_ID	INTEGER	子备份集所属主备份集 ID
5	BACKUPPATH	VARCHAR(1024)	各备份集绝对路径
6	BKP_NUM	INTEGER	各备份集中备份片文件个数, 可能为 0
7	DBF_NUM	INTEGER	各备份集中备份数据文件个数, 可能为 0

数据库和表空间支持并行备份, 备份后会在主备份集中生成多个子备份集, 查询视图 V\$BACKUPSET\_SUBS 可获取子备份集中包含的备份片文件个数及备份数据文件个数等信息。以创建数据库并行备份为例, 查看子备份集的相关信息。

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_bak_for_subs' PARALLEL 3;

SQL>SELECT SF_BAKSET_BACKUP_DIR_ADD('DISK', '/home/dm_bak');

SQL>SELECT BACKUPPATH, BKP_NUM, DBF_NUM FROM V$BACKUPSET_SUBS;
```

查询结果:

LINEID	BACKUPPATH	BKP_NUM	DBF_NUM
1	/home/dm_bak/db_bak_for_subs/db_bak_for_subs_0 1	1	1

2	/home/dm_bak/db_bak_for_subs/db_bak_for_subs_1	1
3	/home/dm_bak/db_bak_for_subs/db_bak_for_subs_2	1

### 3.2.5 数据还原

DM 仅支持表的联机还原，数据库、表空间和归档日志的还原必须通过脱机工具 DMRMAN 执行。本章节主要介绍如何使用 Disql 工具还原表。

本章内容主要包括：

- 表还原

#### 3.2.5.1 表还原

##### 3.2.5.1.1 概述

表还原之后不需要恢复操作。Disql 中输入以下简单的 RESTORE 语句就可还原表：

```
SQL>RESTORE TABLE TAB_01 FROM BACKUPSET 'tab_bak_01';
```

语法如下：

---

```
RESTORE TABLE [<表名>][STRUCT]
[WITH INDEX | WITHOUT INDEX] [WITH CONSTRAINT|WITHOUT CONSTRAINT]
FROM BACKUPSET'<备份集路径>' [DEVICE TYPE <介质类型> [PARMS '<介质参数>']]
[IDENTIFIED BY <密码>] [ENCRYPT WITH <加密算法>]
[TRACE FILE '<TRACE 文件名>'] [TRACE LEVEL <TRACE 日志级别>];
```

---

**TABLE：**需要还原的表名称。

**表名：**指定表名还原时数据库中必须存在该表，否则报错，不会从备份集判断是否存在目标表。

**STRUCT：**执行表结构还原，若未指定，则认为是表中数据还原；表数据还原要求还原目标表结构与备份集中完全一致，否则报错，所以表结构还原可以在表数据还原之前执行，减少报错。

**WITH INDEX/WITHOUT INDEX：**指定还原数据后是否重建二级索引，默认重建。

**WITH CONSTRAINT/WITHOUT CONSTRAINT：**指定还原数据后是否重建约束，默认重建。

**BACKUPSET**:表空间备份时指定的备份集路径。若指定为相对路径,会在默认备份目录下搜索备份集。

**DEVICE TYPE**:指存储备份集的介质类型,目前表还原暂时仅支持 DISK,表示备份集存储介质为磁盘。

**PARMS**:介质参数,只对介质类型为 TAPE 时有效。

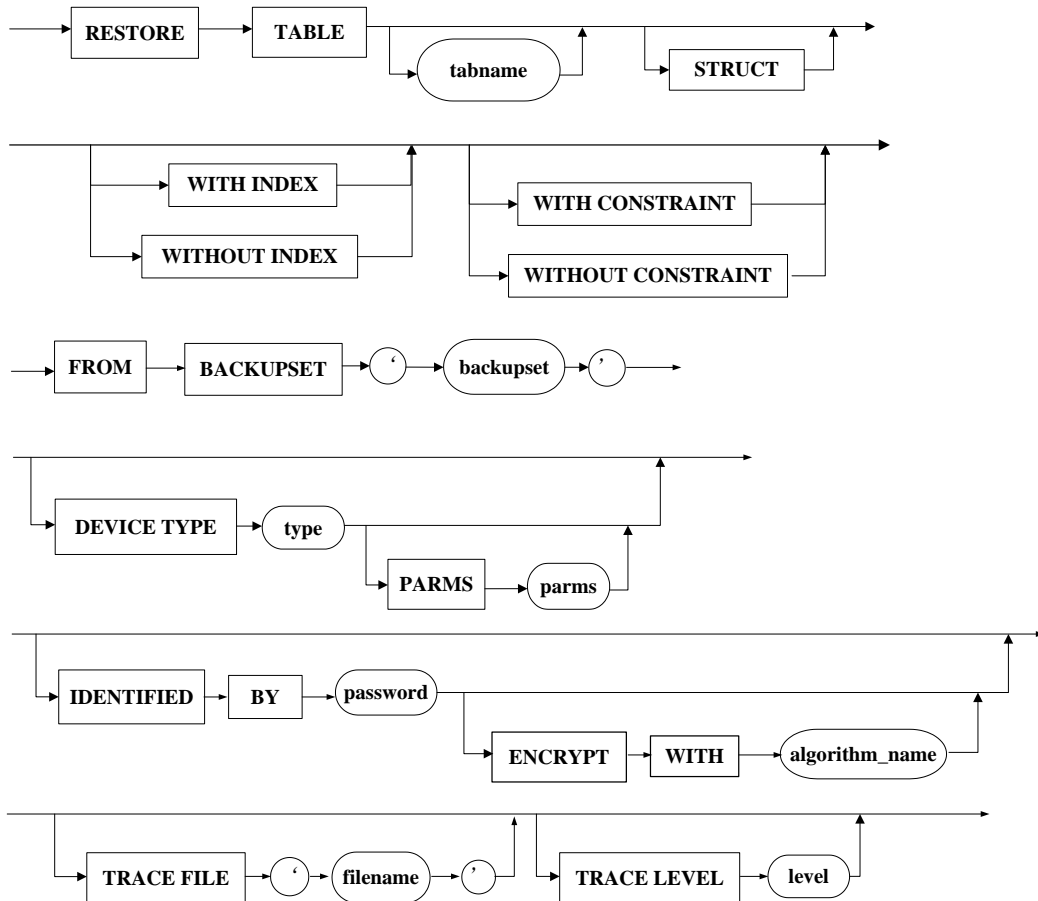
**IDENTIFIED BY**:加密备份表时,用户设置的密码。密码长度为 9 到 48 个字节。

**ENCRYPT WITH**:加密算法。缺省情况下,算法为 AES256\_CFB。

**TRACE FILE**:指定生成的 TRACE 文件名。

**TRACE LEVEL**:有效值 1、2,默认为 1 表示不启用 TRACE,为 2 表示启用 TRACE。

图例:



使用说明:

1. 仅支持对普通用户表进行还原,包括堆表。其中,系统表、临时表、物化视图表、物化视图附属表和日志表、特定模式 (DBG\_PKG/INFORMATION\_SCHEMA/INFO\_SCHEM/SYSREP/SYSGEO/SYSJOB/SYSCP

T/SYS)下的表不支持还原。表列类型为对象类型的表也不支持表还原。若还原表中存在位图连接索引和位图连接虚索引也不支持还原。表名中包含保留字的表不允许备份还原。

2. 备份集路径指备份集所在目录,其中应包含完整备份数据,包括元数据文件(.meta)和备份片文件(.bak)。仅支持从表备份集中还原表。

3. 表名设置为可选参数。若指定,则表必须存在且表定义必须与备份表严格一致;若不指定,则使用备份集中记录的备份表作为还原目标表。

4. 若指定 WITHOUT INDEX,则还原后目标表中无索引信息,即使目标表本来存在,也会被删除;若未指定,则在数据还原后重新创建备份集中索引,目标表原来的索引信息会被删除。

5. 若指定 WITHOUT CONSTRAINT,则还原后目标表中无约束信息,聚集主键除外,即使目标表本来存在,也会被删除;若未指定,则在数据还原后重新创建备份集中约束,目标表原来的约束信息会被删除。

6. 目标表所在的表空间必须处于联机状态。

7. 数据守护环境下,主库允许表备份还原,备库不允许。

8. MOUNT 和 SUSPEND 状态下不允许进行表还原。

9. MPP 环境不允许进行表还原。

10. 若在语句中指定 STRUCT 关键字,则执行表结构还原。表结构还原会根据备份集中备份表还原要求,对目标表定义进行校验,并删除目标表中已存在的二级索引和约束;若未指定 WITHOUT CONSTRAINT,则会将备份集中的 CHECK 约束在目标表上重建。

11. 不指定 STRUCT 关键字,则执行表数据还原,表数据还原默认仅会将备份表中聚集索引上数据进行还原。表数据还原默认仅会在目标表定义与备份表一致且不存在二级索引和约束的情况下执行;若还原时未指定 WITHOUT CONSTRAINT,也允许目标表中存在与备份表中相同的 CHECK 约束。

12. 若在未指定 STRUCT 的情况下,执行还原出现存在二级索引或冗余约束的错误;或者不指定目标表时,报目标不存在的情况,可先执行 STRUCT 还原后,再继续执行实际数据的还原。

13. WITH /WITHOUT INDEX 用于指定还原数据后是否重建二级索引,默认重建;WITH/WITHOUT CONSTRAINT 指定还原数据后是否在目标表上重建约束,默认重建。对于备份表中存在的无效索引和约束,若索引为无效或者不可见,则重建后,也保持无效或者不可见;若约束无效,则重建后,约束也保持无效状态。

14. TRACE FILE 若用户指定，则指定的文件不能为已经存在的文件，否则报错；也不可以为 ASM 文件。

15. 若表中存在大字段列，表备份时 INI 参数 BLOB\_OUTROW\_REC\_STOR 大于 0，而建立还原目标表时 INI 参数 BLOB\_OUTROW\_REC\_STOR 等于 0，那么若大字段列存在行外数据，在执行表还原时会报错，且表数据会丢失。

16. 表还原不检查目标表的缺省表达式（default 值）。

17. 若表列进行了加密，则表还原时不能跨库或跨表还原，只能还原到自身。

### 3.2.5.1.2 表还原

执行表还原，数据库必须处于 OPEN 状态，MOUNT 和 SUSPEND 状态下不允许执行表还原。表还原不需要配置归档，因为表还原是联机完全备份还原，所以不需要借助本地归档日志进行恢复。完整的表备份还原步骤如下：

1) 保证数据库为 OPEN 状态。

2) 创建待备份的表。

```
SQL>CREATE TABLE TAB_FOR_RES_01(C1 INT);
```

3) 备份表数据。

```
SQL>BACKUP TABLE TAB_FOR_RES_01 BACKUPSET '/home/dm_bak/tab_bak_for_res_01';
```

4) 校验备份。此步骤为可选。

```
SQL>SELECT SF_BAKSET_CHECK('DISK','/home/dm_bak/tab_bak_for_res_01');
```

5) 还原表数据。

```
SQL>RESTORE TABLE TAB_FOR_RES FROM BACKUPSET '/home/dm_bak/tab_bak_for_res_01';
```

表还原实质是表内数据的还原，以及索引和约束等的重建。如果备份文件与目标表中都包含索引或约束该如何还原呢？下面以表中包含索引为例说明如何还原表，具体步骤如下：

1) 保证数据库为 OPEN 状态。

2) 创建待备份的表。

```
SQL>CREATE TABLE TAB_FOR_RES_02(C1 INT);
```

3) 创建索引。

```
SQL>CREATE INDEX I_TAB_FOR_RES_02 ON TAB_FOR_RES_02(C1);
```

4) 备份表。

```
SQL>BACKUP TABLE TAB_FOR_RES_02 BACKUPSET '/home/dm_bak/tab_bak_for_res_02';
```

5) 校验备份。此步骤为可选。

```
SQL>SELECT SF_BAKSET_CHECK('DISK','/home/dm_bak/tab_bak_for_res_02');
```

6) 执行表结构还原。表备份和表中都包含索引，如果直接执行表数据还原会报错：还原表中存在二级索引或冗余约束。

```
SQL>RESTORE TABLE TAB_FOR_RES_02 STRUCT FROM BACKUPSET
'/home/dm_bak/tab_bak_for_res_02';
```

7) 执行表数据还原。

```
SQL>RESTORE TABLE TAB_FOR_RES_02 FROM BACKUPSET
'/home/dm_bak/tab_bak_for_res_02';
```

### 3.2.6 数据还原高级主题

本节介绍使用 DISql 工具联机数据还原的高级过程，DISql 工具仅支持对表进行还原。

本节主要包括：

- 表还原

#### 3.2.6.1 表还原

前面的 [3.2.5.1 表还原](#) 已经介绍了表备份还原的基本常用操作，本节将介绍一些表还原的高级过程。主要包括：

- 指定还原时不重建索引
- 指定还原时不重建约束

##### 指定还原时不重建索引

表备份时会默认备份表中的索引，还原时使用 `RESTORE TABLE...WITHOUT INDEX...` 语句可选择还原索引。完整示例如下：

- 1) 保证数据库为 OPEN 状态。
- 2) 准备数据。创建待备份的表及索引。

```
SQL>CREATE TABLE TAB_FOR_IDX_01(C1 INT);
SQL>CREATE INDEX I_TAB_FOR_IDX_01 ON TAB_FOR_IDX_01 (C1 INT);
```

- 3) 备份表数据。

```
SQL>BACKUP TABLE TAB_FOR_IDX_01 BACKUPSET '/home/dm_bak/tab_bak_for_res_01';
```

4) 校验备份。此步骤为可选。

```
SQL>SELECT SF_BAKSET_CHECK('DISK','/home/dm_bak/tab_bak_for_res_01');
```

5) 还原表数据，但不重建索引。

```
SQL>RESTORE TABLE TAB_FOR_RES WITHOUT INDEX FROM BACKUPSET
'/home/dm_bak/tab_bak_for_res_01';
```

#### 指定还原时不重建约束

表备份时会默认备份表中的索引定义，还原时使用 `RESTORE TABLE...WITHOUT CONSTRAINT...` 语句可选择还原时不重建约束。完整示例如下：

1) 保证数据库为 OPEN 状态。

2) 准备数据。创建待备份的表及索引。

```
SQL>CREATE TABLE TAB_FOR_CONS_01(C1 INT);
```

```
SQL>CREATE INDEX I_TAB_FOR_CONS_01 ON TAB_FOR_CONS_01 (C1 INT);
```

3) 备份表数据。

```
SQL>BACKUP TABLE TAB_FOR_CONS_01 BACKUPSET '/home/dm_bak/tab_bak_for_res_01';
```

4) 校验备份。此步骤为可选。

```
SQL>SELECT SF_BAKSET_CHECK('DISK','/home/dm_bak/tab_bak_for_res_01');
```

5) 还原表数据，但不还原约束。

```
SQL>RESTORE TABLE TAB_FOR_CONS_01 WITHOUT CONSTRAINT FROM BACKUPSET
'/home/dm_bak/tab_bak_for_res_01';
```

### 3.3 使用脱机工具 DMRMAN 进行备份还原

DMRMAN (DM RECOVERY MANAGER) 是 DM 的脱机备份还原管理工具，由它来统一负责库级脱机备份、脱机还原、脱机恢复等相关操作，该工具支持命令行指定参数方式和控制台交互方式执行，降低了用户的操作难度。

通过 DMRMAN 工具执行脱机操作过程中，仅会使用通过关键字 `DATABASE` 指定的目标库的本地归档配置信息，不会对本地归档配置文件中其他类型归档配置信息进行校验。例如：`DATABASE` 指定目标库为 `/opt/dmdbms/data/dm.ini`，为单机环境且配置参数 `ARCH_INI` 为 1，那么只会相应的读取 `dmarch.ini` 中本地归档日志信息。此时，即使

dmarch.ini 中配置了 REALTIME、MARCH、TIMELY 等其他类型的归档日志，也不会去校验 dmmal.ini、dmtimer.ini 是否有效。

本章节内容主要包括：

- DMRMAN 概述
- 启动和配置 DMRMAN
- 数据备份
- 管理备份
- 数据库还原和恢复
- 归档还原
- 归档修复
- 表空间还原
- 查看操作日志



注意：

应使用与 DM 数据库版本配套的 DMRMAN 工具进行操作。当使用 DM7 的 DMRMAN 工具操作 DM8 数据库时，无法检测 DM8 数据库实例是否处于启动状态。

### 3.3.1 DMRMAN 概述

DMRMAN 是 DM 提供的命令行工具，无需额外地安装。DMRMAN 的结构比较复杂，为了更好地使用它我们需要了解它的结构。接下来将对 DMRMAN 的重要组成部分进行介绍。

#### 源库

源库是待备份的数据库。使用 DMRMAN 工具只可对数据库进行脱机备份，联机备份需要通过 DISql 工具实现。DMRMAN 可以针对整个数据库执行脱机完全备份和增量备份，数据库可以配置归档也可以不配置。

#### 目标库

目标库是待还原的数据库，即是用来做还原的库，也称为目标还原库。目标库可以通过 dm.ini 文件指定的数据库，也可以是目标文件目录。通过 dm.ini 指定库时，dm.ini 及其配置中的 CTL\_PATH 必须有效，且库必须处于关闭状态；指定文件目录时，指定目录作为 SYSTEM\_PATH 处理，可以存在，也可以不存在，但必须有效。目前目标库的还原仅支持脱机还原，可通过 DMRMAN 和 CONSOLE 工具实现。



### DMRMAN 客户端

DMRMAN 客户端是 DM 软件的一部分，用来执行目标数据库的备份和恢复操作。DMRMAN 客户端是一个命令行工具，命令行的好处是管理员可以编写复杂的脚本，将 DMRMAN 和操作系统的任务调度结合起来可以实现备份的自动化。

### 备份集

利用 DMRMAN 工具将数据库中的一个或多个数据文件、数据库信息等备份到一个称为“备份集”的逻辑结构中，备份集的格式是特定的只能由 DMRMAN 创建和访问。一个备份集中会包含一个或多个二进制文件，这个二进制文件被称为备份片。备份数据库会产生一个备份集，这个备份集可能包含一个或者多个备份片。备份片的个数由数据库中数据文件的大小和文件系统对文件大小的限制共同决定。如果备份的数据文件大小超过了文件系统对文件大小的限制就会产生多个备份片。

在对数据库进行备份时可以指定备份片的大小来适应存储设备的容量。例如，一个数据库的产生的备份集大小约为 40GB，一盘磁带的容量为 10GB，这时可以指定备份片大小为 10G，备份后备份集将包含 4 个备份片，正好使用 4 盘磁带。

## 3.3.2 启动和配置 DMRMAN

本节介绍如何启动命令行工具 DMRMAN 并进行交互。主要内容包括：

- 启动和退出 DMRMAN
- 输入 DMRMAN 命令
- DMRMAN 环境配置

### 3.3.2.1 启动和退出 DMRMAN

安装 DM 数据库后，DMRMAN 可执行程序与数据库其他可执行程序一样位于安装路径的执行码目录下。比如，Linux 上数据库的执行码目录为 /opt/dmdbms/bin，转到执行码目录直接在操作系统的命令行中输入以下命令就可启动 DMRMAN。若配置了环境变量 DM\_HOME，可直接命令行执行：

```
./dmrman
```

启动后控制台中输入 exit 命令即可退出 DMRMAN 环境：

```
DMRMAN>exit
```

### 3.3.2.2 输入 DMRMAN 命令

DMRMAN 工具支持控制台输入命令、命令行设置参数两种操作方式。下面分别对这两种操作方式进行介绍。

#### DMRMAN 控制台输入命令

当 DMRMAN 客户端已经启动且可以输入命令时会显示命令提示符。如下所示：

```
RMAN>
```

输入以下用于执行的 DMRMAN 命令：

```
RMAN>HELP
```

```
RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini';
```

大多数的 DMRMAN 命令会设置多个参数，命令的结尾可以加分号结束，也可以不使用分号。

DMRMAN 控制台还可以执行脚本。如创建一个名为 `cmd_file.txt` 的文件，文件中包含 `"BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'"` 命令，保存到 `/home/dm_cmd` 目录下，执行方式如下：

```
RMAN> `/home/dm_cmd/cmd_file.txt
```

DMRMAN 是按行解析命令，如果输入一行不完整的命令执行会报错，如一条完整的命令分三行输入会导致报错：

```
BACKUP
```

```
DATABASE
```

```
' /opt/dmdbms/data/DAMENG/dm.ini';
```

#### DMRMAN 命令行设置参数执行

DMRMAN 命令行设置参数执行又可分为命令行指定脚本、命令行指定语句两种执行方式。DMRMAN 支持的参数如表 3.3.1 所示。

表 3.3.1 DMRMAN 参数

参数	含义	备注
CTLFILE	指定执行语句所在的文件路径，不能和 CTLSTMT 同时使用。脚本文件格式支持 *.txt	可选参数
CTLSTMT	指定待执行语句，不能和 CTLFILE 同时使用。如：CTLSTMT="BACKUP	可选参数

## 备份与还原

	DATABASE '/home/dmdbms/data/DAMENG/dm.ini'"	
DCR_INI	指定 dmdcr.ini 路径，用于 ASM 存储时访问 ASM 服务；若未指定，则认为不存在 ASM 存储使用。可单独使用，也可与其他参数配合使用。主要用于 DMDSC 环境	可选参数
USE_AP	指定备份还原执行策略。取值 1、2。默认为 1。  1：使用 DMAP 辅助进程方式执行备份还原；执行备份还原时要求先启动 DMAP 服务。  2：无辅助进程方式，由 DMSERVER 进程自身完成备份还原，不再依赖 DMAP 服务；配置成 2 的情况下，不能执行第三方备份（指定 DEVICE TYPE 为 TAPE）。	可选参数
HELP	打印帮助信息	可选参数

命令行指定脚本执行，要求我们先创建一个包含 DMRMAN 命令的文件，然后设置 CTLFILE 参数，参数后面指定文件的路径。例如，创建一个名为 cmd\_rman.txt 的文件，保存到 /home/dm\_cmd 目录下，文件中包含一行如下所示的 DMRMAN 命令：

```
BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini' ;
```

设置 CTLFILE 参数启动 DMRMAN 工具，指定脚本文件中的命令将被执行：

```
./dmrman CTLFILE=/home/dm_cmd/cmd_rman.txt
```

命令执行完后 DMRMAN 工具会自动退出。

命令行执行语句执行，是 DMRMAN 工具设置 CTLSTMT 参数，参数后面执行需要执行的 DMRMAN 命令，使用方法如下所示：

```
./dmrman CTLSTMT="BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini' ;"
```



**CTLFILE 中存在多条语句或 CTLSTMT 中一次输入多条语句时，每条语句都**

**注意：**必须以分号结尾；执行单条语句，语句结尾可以加分号也可以不加。

**CTLSTMT 中，待执行语句外层双引号一定要加。执行完毕后工具自动退出。**

**CTLFILE 和 CTLSTMT 不能同时指定。若使用 DMASM 文件系统，可通过 DCR\_INI 指定目标联机的 ASM 服务器。**

### 3.3.2.3 DMRMAN 环境配置

使用 CONFIGURE 命令进行 DMRMAN 的默认参数配置，配置默认的存储介质类型、备份集搜集目录、归档日志搜集目录、跟踪日志文件。

- 显示和清除现有参数的默认配置
- 配置存储介质类型：DISK 或 TAPE
- 配置备份集搜集目录
- 配置归档日志搜集目录
- 配置跟踪日志文件

语法如下：

---

```
CONFIGURE |

CONFIGURE CLEAR |

CONFIGURE DEFAULT <sub_conf_stmt>

<sub_conf_stmt> ::=

DEVICE [[TYPE<介质类型> [PARMS <第三方参数>]]|CLEAR] |

TRACE [[FILE <跟踪日志文件路径>][TRACE LEVEL <跟踪日志等级>]|CLEAR] |

BACKUPDIR [[ADD|DELETE] '<基备份搜索目录>' {','<基备份搜索目录>' }|CLEAR] |

ARCHIVEDIR [[ADD|DELETE] '<归档日志目录>' {','<归档日志目录>' }

{'<归档日志目录>' {','<归档日志目录>' } }|CLEAR]
```

---

CONFIGURE：查看设置的默认值。

CLEAR：清理参数的默认值。

DEVICE TYPE：备份集存储的介质类型，DISK 或者 TAPE，默认 DISK，详见 1.3 介质管理层。

PARMS：介质参数，供第三方存储介质（TAPE 类型）管理使用。

BACKUPDIR：默认搜集备份的目录，可以设置为不存在但是在系统中是有效的路径。

ARCHIVEDIR：默认搜集归档的目录，可以设置为不存在但是在系统中是有效的路径。

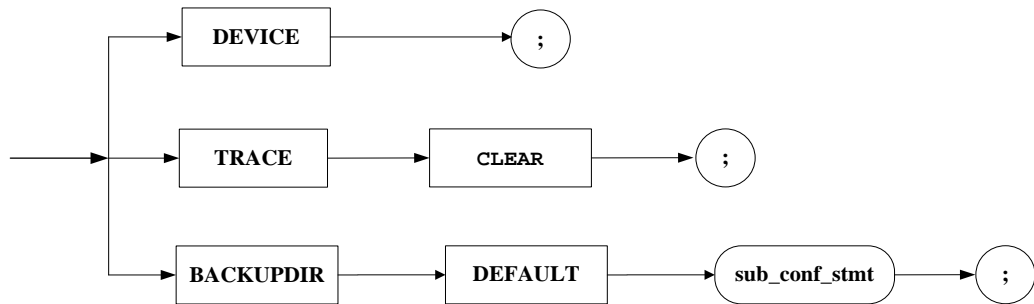
ADD：添加默认备份集搜索目录或归档日志目录，若已经存在，则替换原来的。

DELETE：删除指定默认备份集搜索目录或者归档日志目录。

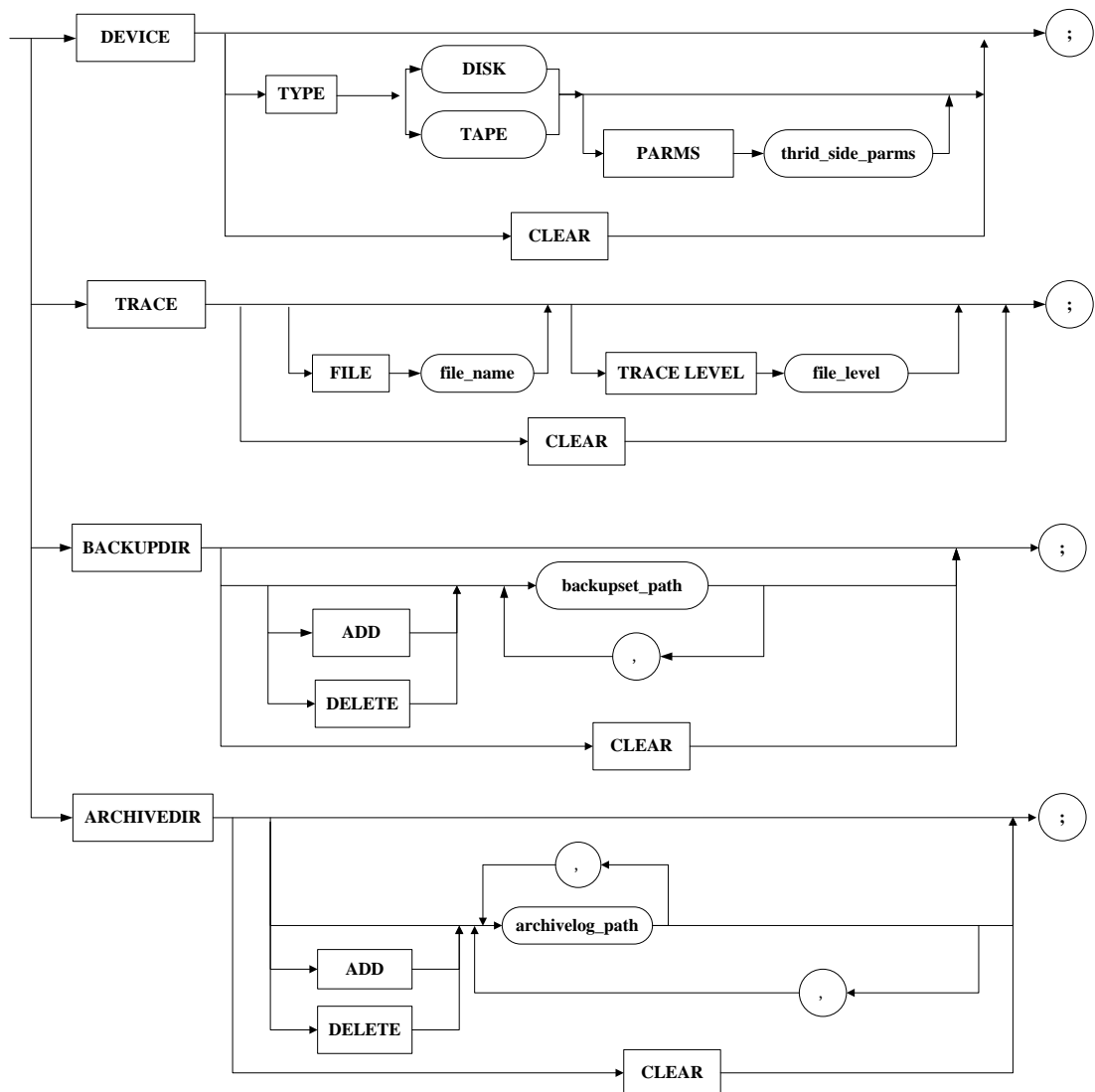
TRACE：介质存储过程中使用的跟踪日志配置，包括文件路径（TRACE FILE）和日志级别（TRACE LEVEL），其中日志级别有效值（1/2，不写/写；默认为 1，不写）。不支持

asm 类型文件。若用户指定，则指定的文件不能为已经存在的文件，否则报错；也不可以为 ASM 文件。

图例：



sub\_conf\_stmt:



使用说明：

1. 设置的参数默认值仅在此 DMRMAN 实例存活期间有效。
2. DMRMAN 命令中如果指定了相同的参数，会覆盖 CONFIGURE 的默认设置。

### 显示和清除所有默认配置项

使用 CONFIGURE 命令就可显示 DMRMAN 配置项的当前值。示例如下：

```
RMAN>CONFIGURE ;

THE DMRMAN DEFAULT SETTING:


DEFAULT DEVICE:

    MEDIA : DISK

DEFAULT TRACE :

    FILE  :

    LEVEL : 1

DEFAULT BACKUP DIRECTORY:

    TOTAL COUNT :0

DEFAULT ARCHIVE DIRECTORY:

    TOTAL COUNT :0


RMAN>CONFIGURE CLEAR;
```

使用 CONFIGURE...CLEAR 命令可恢复任意一个配置项到默认值。示例如下：

```
RMAN>CONFIGURE DEFAULT DEVICE CLEAR;
```

或使用 CONFIGURE...CLEAR 命令清除所有的默认配置。示例如下：

```
RMAN>CONFIGURE CLEAR;
```

### 配置存储介质类型：DISK 或 TAPE

备份时如果没指定备份介质类型参数，则会使用 CONFIGURE 中配置的默认介质类型。

DMRMAN 预先配置的介质类型为 DISK，不需要特别指定。

修改介质类型默认配置的步骤如下：

- 1) 启动 DMRMAN。
- 2) 执行 CONFIGURE 命令查看介质类型当前的默认值。

```
RMAN>CONFIGURE ;
```

- 3) 执行 CONFIGURE DEFAULT DEVICE TYPE 修改默认值。

```
RMAN>CONFIGURE DEFAULT DEVICE TYPE TAPE PARMS 'command';
```

### 配置备份集搜集目录

备份集搜索目录用于增量备份还原中搜索基备份。单个目录最大长度为 256 个字节，可配置的备份集搜索目录没有限制。如果不指定备份集搜集目录只会在库的默认备份目录和当前备份执行备份集目录的上级目录下搜索备份集。

配置备份集搜索目录步骤如下：

```
RMAN>CONFIGURE DEFAULT BACKUPDIR;
```

```
RMAN>CONFIGURE DEFAULT BACKUPDIR '/home/dm_bak1','/home/dm_bak2';
```

若要增加或删除部分备份集搜索目录，不需要对所有的目录重新进行配置，只要添加或删除指定的目录即可。

```
RMAN>CONFIGURE DEFAULT BACKUPDIR ADD '/home/dm_bak3';
```

```
RMAN>CONFIGURE DEFAULT BACKUPDIR DELETE '/home/dm_bak3';
```

### 配置归档日志搜集目录

归档日志搜索目录用于增量备份还原中搜索归档日志。单个目录最大长度为 256 个字节，可配置的归档日志搜索目录没有限制。

配置归档日志搜索目录步骤如下：

```
RMAN>CONFIGURE DEFAULT ARCHIVEDIR;
```

```
RMAN>CONFIGURE DEFAULT ARCHIVEDIR '/home/dm_arch1','/home/dm_arch2';
```

若要增加或删除部分归档日志搜索目录，不需要对所有的目录重新进行配置，只要添加或删除指定的目录即可。

```
RMAN>CONFIGURE DEFAULT ARCHIVEDIR ADD '/home/dm_arch3';
```

```
RMAN>CONFIGURE DEFAULT ARCHIVEDIR DELETE '/home/dm_arch3';
```

### 配置跟踪日志文件

跟踪日志文件记录了 SBT 接口的调用过程，用户通过查看日志可跟踪备份还原过程（见 [3.2.3 数据备份高级主题](#)）。DMRMAN 备份还原命令中不支持设置跟踪日志文件，只能用 CONFIGURE 命令配置，默认配置不记录跟踪日志。

配置跟踪日志文件步骤如下：

1) 显示 TRACE 文件的默认配置。

```
RMAN>CONFIGURE DEFAULT TRACE;
```

2) 配置默认 TRACE 文件。

```
RMAN>CONFIGURE DEFAULT TRACE FILE '/home/dm_trace/trace.log';
```

3) 配置默认 TRACE 级别。

```
RMAN>CONFIGURE DEFAULT TRACE LEVEL 2;
```

其中，TRACE 文件和 TRACE 级别也可以同时设置：

```
RMAN>CONFIGURE DEFAULT TRACE FILE '/home/dm_trace/trace.log ' TRACE LEVEL 2;
```

### 3.3.3 数据备份

本节介绍使用 DMRMAN 工具如何执行数据库和归档的备份操作。

本节内容主要包括：

- 备份数据库
- 备份归档

#### 3.3.3.1 备份数据库

##### 3.3.3.1.1 概述

在 DMRMAN 工具中使用 BACKUP 命令你可以备份整个数据库。使用 DMRMAN 脱机备份数据库需要设置归档和关闭数据库实例。在 DMRMAN 中输入以下命令即可备份数据库：

```
RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini';
```

命令执行完后会在默认的备份路径下生成备份集目录，默认的备份路径为 dm.ini 中 BAK\_PATH 的配置值，若未配置，则为 SYSTEM\_PATH 下的 bak 目录。这是最简单的脱机数据库备份语句，如果要设置其他备份选项请参考下文的语法及使用说明。

语法如下：

---

```
BACKUP DATABASE '<INI 文件路径>' [[[FULL][DDL_CLONE]]] | INCREMENT
[CUMULATIVE][WITH BACKUPDIR '<基备份搜索目录>{'<基备份搜索目录>'}]] [BASE ON
BACKUPSET '<基备份集目录>']]
[TO <备份名>] [BACKUPSET '<备份集目录>'] [DEVICE TYPE <介质类型>[PARMS '<介质参数>']]
[BACKUPINFO '<备份描述>'] [MAXPIECESIZE <备份片限制大小>]
[IDENTIFIED BY <加密密码>[WITH ENCRYPTION<TYPE>][ENCRYPT WITH <加密算法>]]
[COMPRESSED [LEVEL <压缩级别>]][WITHOUT LOG]
```

---



---

```
[TASK THREAD <线程数>][PARALLEL [<并行数>][READ SIZE <拆分块大小>]];
```

---

**DATABASE:** 必选参数。指定还原目标库的 INI 文件路径。

**FULL:** 备份类型。FULL 表示完全备份，可不指定，DMRMAN 会默认为完全备份。

**DDL\_CLONE:** 数据库克隆。该参数只能用于完全备份中，表示仅拷贝所有的元数据不拷贝数据。如对于数据库中的表来说，只备份表的定义不备份表中数据。

**INCREMENT:** 备份类型。INCREMENT 表示增量备份，若要执行增量备份必须指定该参数。

**WITH BACKUPDIR:** 用于增量备份中，指定基备份的搜索目录，最大长度为 256 个字节。若不指定，自动在默认备份目录下搜索基备份。如果基备份不在默认的备份目录下，增量备份必须指定该参数。

**CUMULATIVE:** 用于增量备份中，指明为累积增量备份类型，若不指定则缺省为差异增量备份类型。

**BASE ON BACKUPSET:** 用于增量备份中，为增量备份指定基备份集目录。，如果没有指定基备份集，则会自动搜索一个最近可用的备份集作为基备份集。

**TO:** 指定生成备份名称。若未指定，系统随机生成，默认备份名格式为：DB\_库名\_备份类型\_备份时间。

**BACKUPSET:** 指定当前备份集生成目录。若指定为相对路径，则在默认备份路径中生成备份集。

**DEVICE TYPE:** 指存储备份集的介质类型，支持 DISK 和 TAPE，默认 DISK，详见 1.3 介质管理层。

**PARMS:** 只对介质类型为 TAPE 时有效。

**BACKUPINFO:** 备份的描述信息。最大不超过 256 个字节。

**MAXPIECESIZE:** 最大备份片文件大小上限，以 M 为单位，最小 128M，32 位系统最大 2G，64 位系统最大 128G。

**IDENTIFIED BY:** 指定备份时的加密密码。密码长度为 9 到 48 个字节。密码应用双引号括起来，这样避免一些特殊字符通不过语法检测。密码的设置规则遵行 ini 参数 pwd\_policy 指定的口令策略。

**WITH ENCRYPTION:** 指定加密类型，0 表示不加密，不对备份文件进行加密处理；1 表示简单加密，对备份文件设置口令，但文件内容仍以明文存；2 表示完全数据加密，对备

份文件进行完全的加密，备份文件以密文方式存储。

**ENCRYPT WITH:** 加密算法。缺省情况下，算法为 AES256\_CFB。具体可以使用的加密算法参考联机数据库备份章节的参数说明。

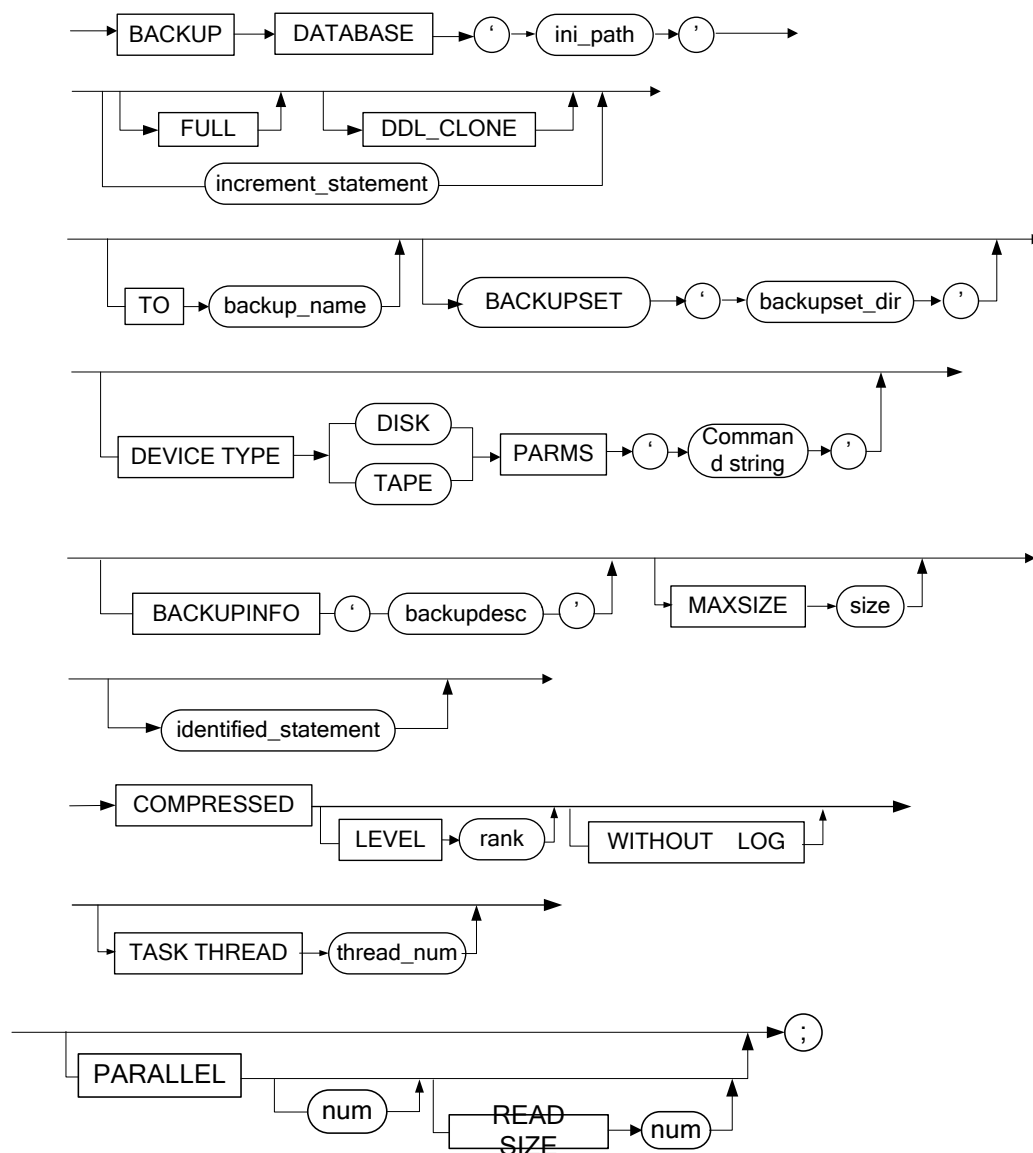
**COMPRESSED:** 取值范围 0~9。0 表示不压缩，1 表示 1 级压缩，9 表示 9 级压缩。压缩级别越高，压缩越慢，但压缩比越高。若未指定，但指定 COMPRESSED，则默认 1；否则，默认 0。

**WITHOUT LOG:** 脱机数据库备份是否备份日志。如果使用，则表示不备份，否则表示备份。如果使用了 WITHOUT LOG 参数，则使用 DMRMAN 工具还原时，必须指定 WITH ARCHIVEDIR 参数。

**TASK THREAD:** 备份过程中数据处理过程线程的个数，取值范围 0~64，默认为 4。若指定为 0，则调整为 1；若指定大于当前系统主机核数，则调整为当前主机核数。线程数（TASK THREAD）\*并行数（PARALLEL）不得超过 512。

**PARALLEL:** 指定并行备份的<并行数>和 READ SIZE<拆分块大小> 请参考 [3.2.2.1.1 概述](#)。

**图例：**



increment\_statement、identified\_statement: 请参考 3.2.2.1.1[概述](#) 中的数据库备份图例。

使用说明：

1. 备份成功后会在<备份集目录>或者备份默认目录下生成备份集。备份集中包括一个备份元数据文件，后缀.meta，一个或多个备份片文件，后缀.bak。
2. 对于并行备份的备份集，备份集中还包括其他子备份集目录，但每个子备份集目录中也都包含一个 meta 文件，0 个或者多个备份片文件。
3. DDL\_CLONE 库备份集不能作为增量备份的基备份，仅能用于库级还原。
4. 脱机备份的数据库可以正常退出库，也可以是故障退出的数据库。若是故障退出的数据库，则备份前，需先进行归档修复。
5. 在执行脱机数据库备份过程中，如果报错归档不完整，则需要检查库是不是异常退

出。如果库是异常退出，则需要先进行归档修复。

6. STANDBY 模式下的库不支持脱机备份。

7. 备份还原回来的库，服务器没有经过重启的，不支持脱机备份。

8. STANDBY 模式切换为 PRIMARY 模式后，服务器没有经过重启的库，不支持脱机备份。

### 3.3.3.1.2 备份数据库

本节主要描述使用 DMRMAN 如何执行基本的脱机数据库备份及实施一些备份策略，包括：

- 设置备份选项
- 创建完全备份
- 创建增量备份

#### 设置备份选项

备份命令如果仅指定了必选参数如 "BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'", 那么 DMRMAN 会根据配置的环境及内置的参数默认值自动指定备份介质类型、备份路径、备份片大小等参数。用户备份时也可以指定这些参数来覆盖默认值，常见的备份选项有设置备份集路径、指定备份名、限制备份片大小、添加描述信息、并行备份等。DMRMAN 脱机备份数据库的语法与使用 DISQL 联机备份类似，关于备份选项的详细介绍及使用参见 [3.2.2.1.2 设置备份选项](#)。

#### 创建完全备份

执行数据库备份要求数据库处于脱机状态。与联机备份数据库不同的是，脱机备份不需要配置归档。一个完整的创建脱机数据库备份的示例如下：

- 1) 启动 DMRMAN 命令行工具。
- 2) 保证数据库处于脱机状态。
- 3) DMRMAN 中输入以下命令：

```
DMRMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini' FULL BACKUPSET  
'/home/dm_bak/db_full_bak_01';
```

命令中的 FULL 参数表示执行的备份为完全备份，也可以不指定该参数，DMRMAN 默认执行的备份类型为完全备份。

### 创建增量备份

增量备份指基于指定的库的某个备份（完全备份或者增量备份），备份自该备份以来所有发生修改了的数据页。脱机增量备份要求两次备份之间数据库必须有操作，否则备份会报错。关于如何制定备份策略请参考 3.2.2.1.3 [备份数据库](#)。

增量备份示例如下：

- 1) 启动 DMRMAN 命令行工具。
- 2) 保证数据库处于脱机状态。
- 3) DMRMAN 中输入以下命令：

```
DMRMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini' INCREMENT WITH BACKUPDIR  
'/home/dm_bak'BACKUPSET '/home/dm_bak/db_increment_bak_02';
```

命令中的 INCREMENT 参数表示执行的备份为增量备份，增量备份该参数不可省略。如果增量备份的基备份不在默认备份目录，必须指定 WITH BACKUPDIR 参数用于搜索基备份集，或者使用 CONFIGURE...BACKUPDIR 命令配置默认的基备份集搜索目录（参考 3.3.2.3 [DMRMAN 环境配置](#)）。

DSC 环境下的脱机库备份与联机库备份存在相同的使用限制，详细情况请参考 3.2.2.1.4 [DSC 环境使用说明](#)。由于脱机备份无法推进检查点，当不满足该约束时，备份将失败。

## 3.3.3.2 备份归档

### 3.3.3.2.1 概述

在 DMRMAN 工具中使用 BACKUP 命令你可以备份库的归档。使用 DMRMAN 备份归档需要设置归档，否则会报错。关闭数据库实例，在 DMRMAN 中输入以下命令即可备份归档：

```
DMRMAN>BACKUP ARCHIVE LOG DATABASE '/opt/dmdbms/data/DAMENG/dm.ini';
```

命令执行完后会在默认的备份路径下生成备份集目录，默认的备份路径为 dm.ini 中 BAK\_PATH 的配置值，若未配置，则使用 SYSTEM\_PATH 下的 bak 目录。这是最简单的脱机归档备份语句，如果要设置其他备份选项请参考下文的语法及使用说明。

语法如下：

---

**BACKUP**<ARCHIVE LOG | ARCHIVELOG>

---

---

```
[ALL | [FROM LSN <lsn 值>] | [UNTIL LSN <lsn 值>] | [LSN BETWEEN < lsn 值> AND < lsn 值>] | [FROM TIME '时间串'] | [UNTIL TIME '时间串'] | [TIME BETWEEN '时间串' AND '时间串']] [<notBackedUpSpec>][DELETE INPUT]

DATABASE '<INI 文件路径>'

[TO <备份名>] [BACKUPSET '<备份集目录>'] [DEVICE TYPE <介质类型>][PARMS '<介质参数>']

[BACKUPINFO '<备份描述>'] [MAXPIECESIZE <备份片限制大小>]

[IDENTIFIED BY <加密码>][WITH ENCRYPTION<TYPE>][ENCRYPT WITH <加密算法>]]

[COMPRESSED [LEVEL <压缩级别>]][TASK THREAD <线程数>][PARALLEL [<并行数>][READ SIZE <拆分块大小>]];

<notBackedUpSpec>::=NOT BACKED UP
| NOT BACKED UP numTIMES
| NOT BACKED UP SINCE TIME 'datetime_string'
```

---

ALL: 备份所有的归档;

FROM LSN , UNTIL LSN: 备份的起始和截止 lsn。请参考 [3.2.2.4 归档备份](#)

FROM TIME: 指定备份的开始时间点。例如, '2018-12-10'。

UNTIL TIME: 指定备份的截止时间点。

BETWEEN ...AND ...: 指定备份的区间, 仅仅指备份区间内的归档文件。

<notBackedUpSpec>: 搜索过滤。搜索过滤仅限于根据备份指定条件能找到的所有归档备份集。1) num TIMES, 指若归档文件已经备份了 num 次, 则不再备份; 否则备份。如 num=3, 则认为已经备份了 3 次的归档文件就不再备份。若 num=0, 则认为所有都不需要备份。2) SINCE TIME 'datetime\_String', 指定时间开始没有备份的归档文件进行备份。3) 若以上两种均未指定, 则备份所有未备份过的归档日志文件。

DELETE INPUT: 用于指定备份完, 是否删除归档操作。

DATABASE: 必选参数。指定备份目标库的 INI 文件路径。

TO: 指定生成备份名称。若未指定, 系统随机生成, 默认备份名格式为: ARCH\_备份时间。

BACKUPSET: 指定当前备份集生成目录。若指定为相对路径, 则在默认备份路径中生成备份集。

DEVICE TYPE: 指存储备份集的介质类型, 支持 DISK 和 TAPE, 默认 DISK, 详见 1.3 介质管理层。

PARMS: 只对介质类型为 TAPE 时有效。

BACKUPINFO: 备份的描述信息。最大不超过 256 个字节。

MAXPIECESIZE: 最大备份片文件大小上限, 以 M 为单位, 最小 128M, 32 位系统最大 2G, 64 位系统最大 128G。

IDENTIFIED BY: 指定备份时的加密密码。密码应用双引号括起来, 这样避免一些特殊字符通不过语法检测。密码的设置规则遵行 ini 参数 `pwd_policy` 指定的口令策略。

WITH ENCRYPTION: 指定加密类型, 0 表示不加密, 不对备份文件进行加密处理; 1 表示简单加密, 对备份文件设置口令, 但文件内容仍以明文方式存储; 2 表示完全数据加密, 对备份文件进行完全的加密, 备份文件以密文方式存储。

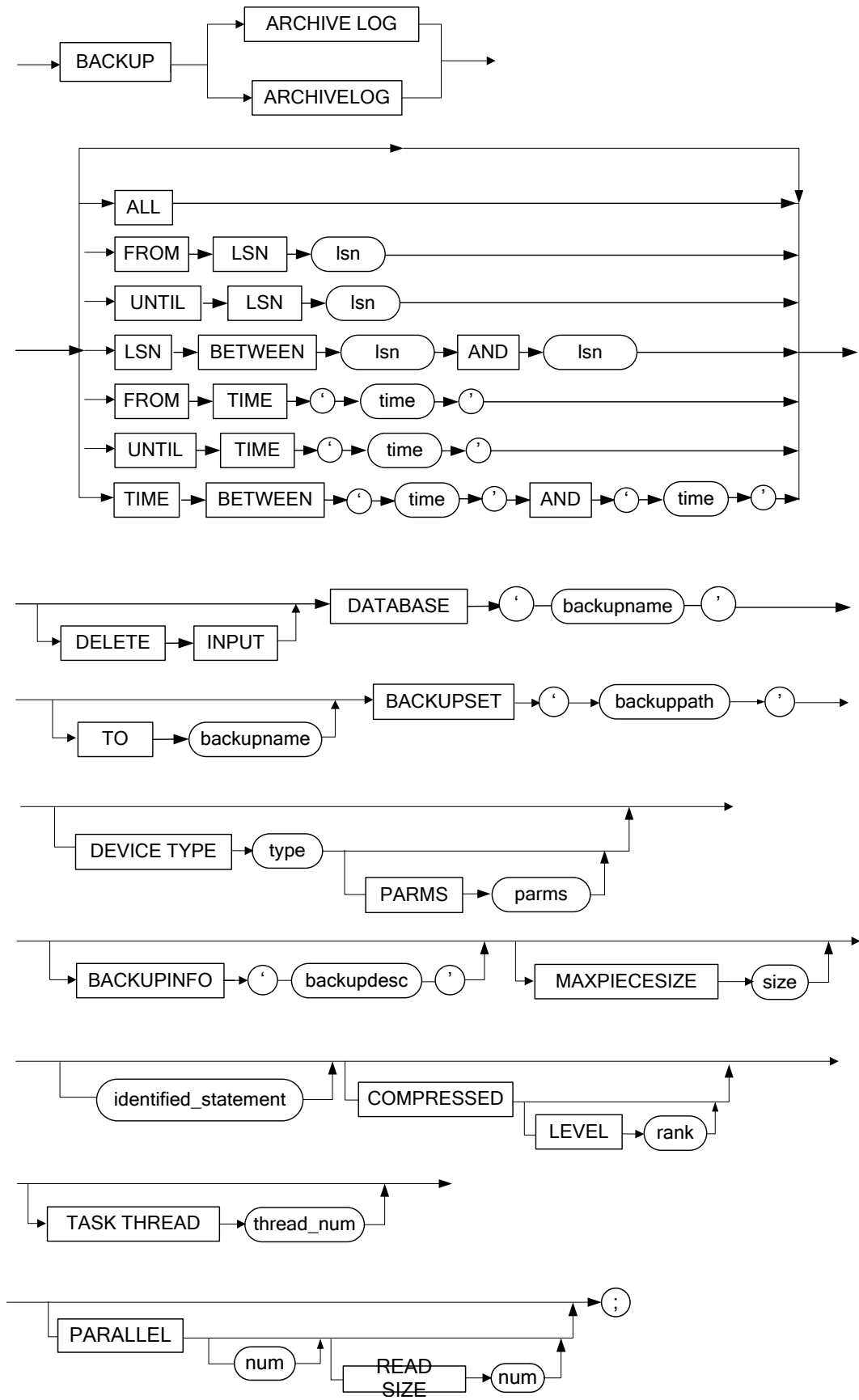
ENCRYPT WITH: 加密算法。缺省情况下, 算法为 AES256\_CFB。具体可以使用的加密算法参考联机数据库备份章节的参数说明。

COMPRESSED: 取值范围 0~9。0 表示不压缩, 1 表示 1 级压缩, 9 表示 9 级压缩。压缩级别越高, 压缩越慢, 但压缩比越高。若未指定, 但指定 COMPRESSED, 则默认 1; 否则, 默认 0。

TASK THREAD: 备份过程中数据处理过程线程的个数, 取值范围 0~64, 默认为 4。若指定为 0, 则调整为 1; 若指定超过当前系统主机核数, 则调整为主机核数。线程数 (TASK THREAD) \* 并行数 (PARALLEL) 不得超过 512。

PARALLEL: 指定并行备份的<并行数>和 READ SIZE<拆分块大小> 请参考 [3.2.2.1.1 概述](#)。

图例:



identified\_statement: 请参考 3.2.2.1.1 [概述](#) 中数据库备份图例

notBackedUpSpec: 请参考 3.2.2.4.1 [概述](#) 中归档备份图例



### 3.3.3.2.2 备份归档

本节主要描述使用 DMRMAN 如何执行基本的脱机归档备份及实施一些备份策略,包括:

- 设置备份选项
- 创建归档备份
- 创建设置条件的归档备份

#### 设置备份选项

备份命令如果仅指定了必选参数如 "BACKUP ARCHIVE LOG DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'", 那么 DMRMAN 会根据配置的环境及内置的参数默认值自动指定备份介质类型、备份路径、备份片大小等参数。用户备份时也可以指定这些参数来覆盖默认值,常见的备份选项有设置备份集路径、指定备份名、限制备份片大小、添加描述信息、并行备份等。DMRMAN 脱机备份归档的语法与使用 DISQL 联机备份类似,关于备份选项的详细介绍及使用参见 [3.2.2.3.2 设置备份选项](#)。

#### 创建归档备份

执行归档备份要求数据库处于脱机状态。与联机备份数据库一样,脱机归档备份需要配置归档。一个完整的创建脱机数据库备份的示例如下:

- 1) 启动 DMRMAN 命令行工具。
- 2) 保证数据库处于脱机状态。
- 3) DMRMAN 中输入以下命令:

```
DMRMAN>BACKUP ARCHIVE LOG ALL DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'BACKUPSET  
'/home/dm_bak/arch_all_bak_01';
```

命令中的 ALL 参数表示执行的备份为备份所有的归档,也可以不指定该参数,DMRMAN 默认执行的备份类型为 ALL 类型归档备份。

#### 创建设置条件的归档备份

设置条件的归档备份指通过设置 LSN 或者时间点,控制归档需要备份归档的范围。脱机设置条件归档备份。

增量备份示例如下:

- 1) 启动 DMRMAN 命令行工具。
- 2) 保证数据库处于脱机状态。
- 3) DMRMAN 中输入以下命令:

```

RMAN>BACKUP ARCHIVE LOG LSN BETWEEN 50000 AND 120000 DATABASE
'/opt/dmdbms/data/DAMENG/dm.ini' BACKUPSET '/home/dm_bak/db_increment_bak_02';

```

命令中的 LSN BETEAWN start AND end 参数表示执行的备份为执行备份的区间，或者设置 FROM LSN(TIME) 以及 UNTIL LSN(TIME)。

### 3.3.3.3 数据备份高级主题

本节介绍使用 DMRMAN 工具执行高级的数据库脱机备份过程，主要包括：

- 加密备份
- 设置跟踪日志文件

#### 加密备份

DMRMAN 同 Disql 工具一样可使用加密的方式备份数据库，没有权限的用户无法访问加密的备份集。

DMRMAN 备份命令中通过指定 IDENTIFIED BY...WITH ENCRYPTION...ENCRYPT WITH...命令执行加密备份。其中，加密备份相关参数的含义与联机备份中的含义相同（参考 3.2.3 [数据备份高级主题](#)）。

加密备份过程中参数 IDENTIFIED BY 必须指定，参数 WITH ENCRYPTION 和参数 ENCRYPT WITH 可不指定。加密备份时不指定 WITH ENCRYPTION 参数，该参数默认为 1，不指定 ENCRYPT WITH 参数，该参数默认值为 AES256\_CFB。

例如，以下两种加密备份命令都是合法的：

```

RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini' BACKUPSET
'/home/dm_bak/db_bak_for_encrypt_01' IDENTIFIED BY "cdb546";
RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini' BACKUPSET
'/home/dm_bak/db_bak_for_encrypt_02' IDENTIFIED BY "cdb546" ENCRYPT WITH RC4;

```

若指定了加密密码，但加密类型 WITH ENCRYPTION 参数指定为 0，则为非加密备份，如下所示：

```

RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini' BACKUPSET
'/home/dm_bak/db_bak_for_encrypt_03' IDENTIFIED BY "cdb546789" WITH ENCRYPTION
0;

```

下面以数据库完全备份为例，创建加密密码为“cdb546789”，加密算法为“rc4”的复

杂加密类型的数据库加密备份，完整步骤如下：

- 1) 保证数据库处于关闭状态。
- 2) 备份数据库。启动 DMRMAN 工具并输入以下命令。

```
RMAN>BACKUP          DATABASE          '/opt/dmdbms/data/DAMENG/dm.ini'BACKUPSET  
'/home/dm_bak/db_bak_for_encrypt_04' IDENTIFIED BY "cdb546" WITH ENCRYPTION 2  
ENCRYPT WITH RC4;
```

对于增量备份加密，如果基备份存在加密，则使用的加密算法和加密密码必须与基备份中一致；如果基备份未进行加密处理，则对增量备份使用的加密密码和算法没有特殊要求。

### 设置跟踪日志文件

DMRMAN 备份时可选择生成跟踪日志文件，跟踪日志记录了 SBT 接口的调用过程，用户通过查看日志可跟踪备份还原过程。

与生成跟踪日志文件相关的参数有两个，TRACE FILE 和 TRACE LEVEL。TRACE FILE 用于指定生成的跟踪日志文件路径。与 Disql 工具不同的是，DMRMAN 不可在备份时指定参数生成跟踪文件，只能使用 CONFIGURE 命令进行事先配置。

使用 CONFIGURE DEFAULT...TRACE FILE...TRACE LEVEL 命令启用 TRACE 功能并设 TRACE 文件路径（参考 3.3.2.3 [DMRMAN 环境配置](#)），以下命令生成 TRACE 文件到 /home/dm\_trace 目录：

```
RMAN>CONFIGURE DEFAULT TRACE FILE '/home/dm_trace/trace.log ' TRACE LEVEL 2;
```

指定参数 TRACE FILE 但 TRACE LEVEL 值设置为 1 即不启用 TRACE 功能，会生成 TRACE 文件，但不会写入 TRACE 信息。如下所示：

```
RMAN> CONFIGURE DEFAULT TRACE FILE'/home/dm_log/db_trace.log' TRACE LEVEL 1;
```

TRACE LEVEL 值设置为 2 即启用 TRACE 功能，但若 TRACE FILE 没有指定，系统默认在执行码路径的 log 目录下生成 DM\_SBTTRACE\_年月.log 文件。如下所示：

```
RMAN> CONFIGURE DEFAULT TRACE LEVEL 2;
```

若 TRACE FILE 使用相对路径，日志文件则生成在执行码同级目录下。

为数据库脱机备份设置跟踪日志文件的操作步骤如下：

- 1) 保证数据库处于关闭状态。
- 2) 使用 CONFIGURE 命令配置生成跟踪日志文件。

```
RMAN>CONFIGURE DEFAULT TRACE FILE '/home/dm_trace/trace.log ' TRACE LEVEL 2;
```

- 3) 备份数据库。启动 DMRMAN 工具输入以下命令：

```
RMAN>BACKUP          DATABASE          '/opt/dmdbms/data/DAMENG/dm.ini' BACKUPSET  
'/home/dm_bak/db_bak_for_trace_01' ;
```

查看 `/home/dm_trace/trace.log` 文件即可跟踪本次备份的 SBT 接口调用过程。

如果指定的 TRACE 文件已存在，不会覆盖已存在的文件而是在已有文件基础上继续记录日志。

### 3.3.4 管理备份

本章节主要介绍如何使用 DMRMAN 工具管理数据库备份、表空间备份及表备份。本节内容主要包括：

- 概述
- 备份集查看
- 备份集校验
- 备份集删除
- 备份集导入
- 备份集映射文件导出

#### 3.3.4.1 概述

管理备份一个重要的目的是删除不再需要的备份。DMRMAN 工具提供 SHOW、CHECK、REMOVE、LOAD 等命令分别用来查看、校验、删除和导入备份集。下文将对这些命令进行详细介绍。若命令中指定了 `dm.ini`，则要求 `dm.ini` 配置正确。

#### 3.3.4.2 备份集查看

##### 3.3.4.2.1 概述

DMRMAN 中使用 SHOW 命令可以查看备份集的信息，包括：

- ✓ 备份集的数据库信息
- ✓ 备份集的元信息
- ✓ 备份集中文件信息（如备份数据文件 DBF 和备份片文件）

- ✓ 备份集中表信息（仅对表备份集有效）

若指定具体备份集目录，则会生成相应的备份集链表信息。

语法如下：

---

```

SHOW    BACKUPSET    '< 备 份 集 目 录 >'    [<device_type_stmt>][RECURSIVE]
[<database_bakdir_lst_stmt>] [<info_type_stmt>] [<to_file_stmt>];

| SHOW BACKUPSETS [<device_type_stmt>] <database_bakdir_lst_stmt>
[<info_type_stmt>] [<use_db_magic_stmt>] [<to_file_stmt>];

<database_bakdir_lst_stmt>::= DATABASE '<INI_PATH>' |
WITH BACKUPDIR '<备份集搜索目录>'{','<备份集搜索目录>'} |
DATABASE '<INI_PATH>' WITH BACKUPDIR '<备份集搜索目录>'{','<备份集搜索目录>'}

<device_type_stmt>::= DEVICE TYPE DISK|TAPE [PARMS '<介质参数>']

<info_type_stmt>::= INFO DB[,META][,FILE] [,TABLE]

<use_db_magic_stmt>::= USE DB_MAGIC <db_magic>

<to_file_stmt>::= TO '<输出文件路径>' [FORMAT TXT | FORMAT XML]

```

---

**BACKUPSET:** 指定显示目标备份集信息，若同时指定 **RECURSIVE**，则显示以该备份集为最新备份集递归显示完整的备份集链表；否则，仅显示指定备份集本身信息。若为完全备份，则仅显示该备份集自身信息。

**DATABASE:** 指定数据库 `dm.ini` 文件路径，若指定，则该数据库的默认备份目录作为备份集搜索目录之一。

**WITH BACKUPDIR:** 备份集搜索目录，最大长度为 256 个字节。若查看完全备份的备份集信息且指定的备份集路径为相对路径可通过设置此参数搜索备份集；若查看的为增量备份集信息，设置该参数除上述功能外还用于搜索基备份集。

**<device\_type\_stmt>:** 指定介质类型和介质参数，介质类型支持 **DISK** 和 **TAPE**，默认 **DISK**。

**<info\_type\_stmt>:** 指定显示备份集信息内容，可以组合指定，若未指定，则显示全部。具体说明如下：

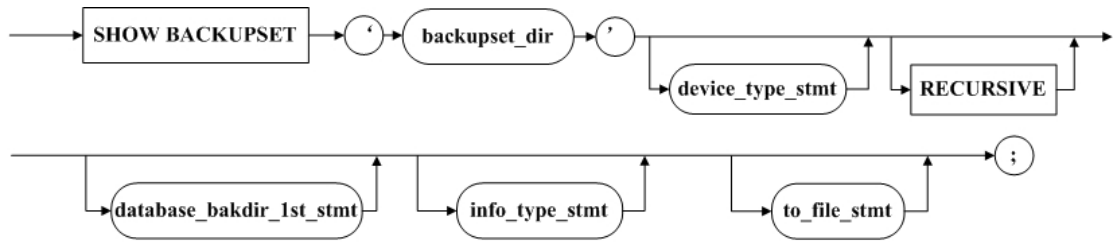
- ✓ **DB** 表示仅显示备份集的数据库信息；
- ✓ **META** 表示仅显示备份集的元信息；
- ✓ **FILE** 表示仅显示备份集中文件信息，如备份数据文件 **DBF** 和备份片文件；

✓ TABLE 表示显示备份集中表信息，仅对表备份集有效。

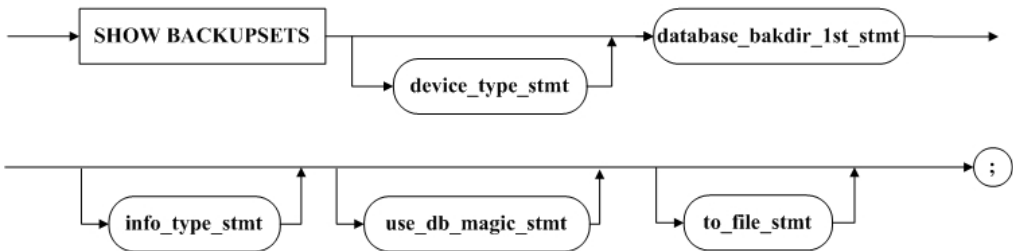
<use\_db\_magic\_stmt>: SHOW BACKUPSETS 可以指定仅显示指定 DB\_MAGIC 即指定数据库的备份集信息。

<to\_file\_stmt>: 指定备份集信息输出的目标文件路径，若不指定，仅控制台打印。文件格式有两种类型，TXT 和 XML 格式，默认是 TXT 格式。不支持输出到 DMASN 文件系统中。指定的文件不能为已经存在的文件，否则报错。

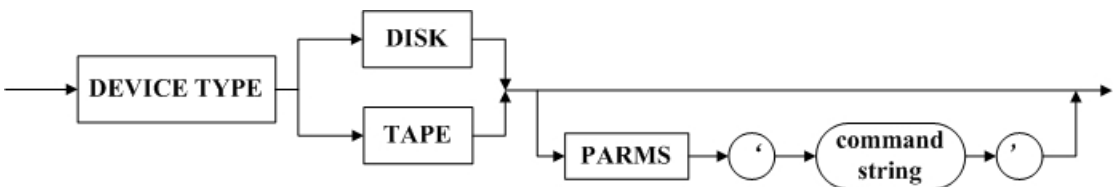
图例：



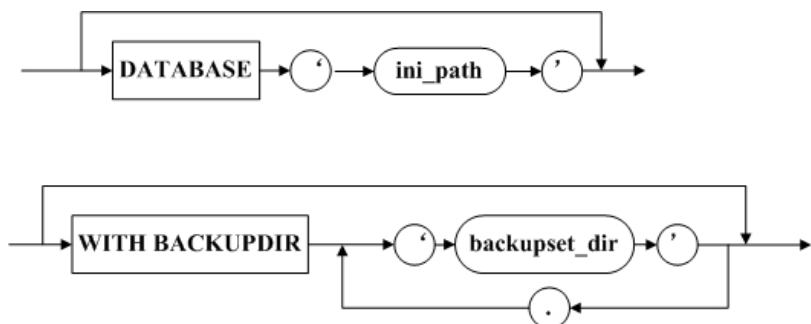
或



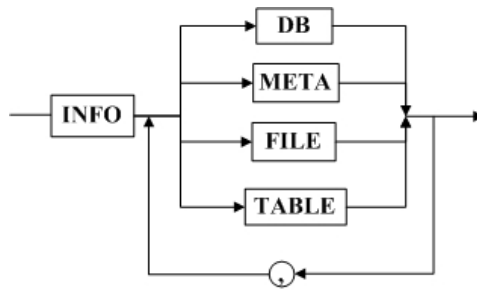
<device\_type\_stmt>:



<database\_bakdir\_1st\_stmt>:



<info\_type\_stmt>:



<use\_db\_magic\_stmt>:



### 3.3.4.2.2 备份信息查看

SHOW 命令支持查看单个备份集信息也支持批量查看多个备份集的信息。本节主要举例说明如何执行简单的常用的查看备份集信息操作，包括：

- 查看指定的备份集信息
- 批量显示备份集信息
- 查看指定数据库所有备份集的信息
- 指定显示部分备份集信息
- 以 xml 格式输出备份信息到文件

#### 查看指定的备份集信息

SHOW BACKUPSET...命令用于显示特定的备份集信息，每次只能显示一个备份集。

当仅需要查看某个特定备份集信息时可以使用此命令。执行步骤如下：

```

RMAN> show backupset '/home/dm_bak/DB_FULL_DAMENG_20190522_133248_000770'

show backupset '/home/dm_bak/DB_FULL_DAMENG_20190522_133248_000770'

total 0 packages processed...

<backupset [DEVICE TYPE:DISK, BACKUP_PATH:
/home/dm_bak/DB_FULL_DAMENG_20190522_133248_000770] info start .....>

<DB INFO>

system path:          /home/DAMENG

pmnt_magic:           1018589430
  
```

```

src_db_magic:          1408921336
db_magic:              1408921336
dsc node:              1
sys mode:              0
page check:            0
rlog encrypt:          0
external cipher[id/name]: 0/
external hash[id/name]: 0/
length in char:        0
use new hash:          1
page size:             8KB
extent size:           16
case sensitive:        1
log page size:         512B
unicode_flag/charset:  0
data version:          0x7000A
sys version:           V8.1.0.160-Build(2019.05.21-107084)ENT
enable policy:         0
archive flag:          1
blank_pad_mode:        0
crc_check:             TRUE

<META INFO>
backupset sig:         BA
backupset version:     0x4009
database name:         DAMENG
backup name:           DB_FULL_DAMENG_20190522_133248_000770
backupset description:
n_magic:               0x68305D27
parent n_magic:        0xFFFFFFFF

```



```

meta file size :      86528
compressed level:      0
encrypt type:         0
parallel num:         1
backup range:         database
mpp_timestamp:        1558503168
bakset_type:          NORMAL
mpp_flag:             FALSE
backup level:         online
backup type:          full
without log:          FALSE
end_lsn:              1274185
base begin_lsn:       -1
base end_lsn:         -1
base n_magic:         0xFFFFFFFF
base name:
base backupset:
backup time:          2019-05-22 13:32:55
min exec ver:         0x0701060C
pkg size:             0x02000000

<EP INFO>
EP[0]:
begin_pkg_seq:        6132
begin_lsn:            1268123
end_pkg_seq:          6191
end_lsn:              1274185

<FILE INFO>
backupset directory:  /home/dm_bak/DB_FULL_DAMENG_20190522_133248_000770

```

```

backupset name:          DB_FULL_DAMENG_20190522_133248_000770

backup data file num:    6

backup piece num:        2

<backup_piece_list>

$file_seq |$size(KB) |$pos_desc

|$content_type

0          |42410      |DB_FULL_DAMENG_20190522_133248_000770.bak

|DATA

1          |740        |DB_FULL_DAMENG_20190522_133248_000770_1.bak

|LOG

<data_file_list>

$file_seq |$group_id |$group_name      |$file_id |$file_path

|$mirror_path          |$file_len

1          |0          |SYSTEM           |0         |/home/DAMENG/SYSTEM.DBF |

|201326592

2          |0          |SYSTEM           |1         |

|F:\tmp\db_dev2\DAMENG\SYSTEM_01.dbf |104857600

3          |1          |ROLL             |0         |/home/DAMENG/ROLL.DBF   |

|1073741824

4          |1          |ROLL             |1         |/home/DAMENG/ROLL_01.dbf |

|104857600

5          |4          |MAIN             |0         |/home/DAMENG/MAIN.DBF   |

|231735296

<arch_file_list>

$file_seq |$dsc_seq |$file_len          |$begin_seqno      |$begin_lsn

|$end_seqno      |$end_lsn

6          |0          |753152            |6132              |1268123

```

```
|6191                |1274185

<backupset [DEVICE TYPE:DISK, BACKUP_PATH:
/home/dm_bak/DB_FULL_DAMENG_20190522_133248_000770] info end .>

show backupsets successfully.

time used: 249.062(ms)
```

显示的备份集信息分为三类,依次是元数据信息(META INFO)、文件信息(FILE INFO)和数据库信息(DB INFO)。用户可根据自身需要只显示部分备份集信息,本节下文会有介绍。

### 批量显示备份集信息

SHOW BACKUPSETS...命令用于批量显示指定搜索目录下的备份集信息。如需要查看的多个备份集不在同一个目录下,可通过多次指定 WITH BACKUPDIR 参数同时查看所有的备份集。操作如下:

```
RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'
BACKUPSET'/home/dm_bak1/db_bak_for_show_01';

RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'
BACKUPSET'/home/dm_bak2/db_bak_for_show_01';

RMAN>SHOW BACKUPSETS WITH BACKUPDIR '/home/dm_bak1','/home/dm_bak2';
```

### 查看指定数据库所有备份集的信息

如果指定的备份搜索目录下包含不同数据库的备份集,而我们只想查看某个特定数据库的所有备份集信息,此时可以使用 SHOW BACKUPSETS...USE DB\_MAGIC 命令实现。具体操作如下:

- 1) 备份不同数据库至同一目录下。

```
RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'
BACKUPSET'/home/dm_bak/db_bak_for_show_db_magic_01';

RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG2/dm.ini' BACKUPSET'/home/dm_bak/
db_bak_for_show_db_magic_02';
```

- 2) 查看指定数据库备份集的信息,获取 DB\_MAGIC 信息。

```
RMAN>SHOW BACKUPSET'/home/dm_bak/db_bak_for_show_db_magic_01';
```

通过查看备份集信息可知,数据库/opt/dmdbms/data/DAMENG 的 DB\_MAGIC 值为

1447060265。

3) 查看指定目录下数据库/opt/dmdbms/data/DAMENG的所有备份集信息。

```
RMAN>SHOW BACKUPSETS WITH BACKUPDIR '/home/dm_bak' USE DB_MAGIC 1447060265;
```

#### 指定查看备份集的元数据信息

SHOW BACKUPSET...INFO META 命令用于查看备份集的元数据信息。操作步骤如下：

```
RMAN> show backupset '/home/dm_bak/DB_FULL_DAMENG_20190522_133248_000770' info
meta

show backupset '/home/dm_bak/DB_FULL_DAMENG_20190522_133248_000770'

total 0 packages processed...

<backupset [DEVICE TYPE:DISK, BACKUP_PATH:
/home/dm_bak/DB_FULL_DAMENG_20190522_133248_000770] info start .....>

<META INFO>

backupset sig:          BA
backupset version:      0x4009
database name:          DAMENG
backup name:             DB_FULL_DAMENG_20190522_133248_000770
backupset description:

n_magic:                 0x68305D27
parent n_magic:          0xFFFFFFFF
meta file size :        86528
compressed level:        0
encrypt type:             0
parallel num:             1
backup range:             database
mpp_timestamp:           1558503168
ddl_clone:               FALSE
mpp_flag:                 FALSE
backup level:             online
```

```

backup type:          full
without log:          FALSE
end_lsn:              1274185
base begin_lsn:       -1
base end_lsn:         -1
base n_magic:         0xFFFFFFFF
base name:
base backupset:
backup time:          2019-05-22 13:32:55
min exec ver:         0x0701060C
pkg size:             0x02000000

<EP INFO>
EP[0]:
begin_pkg_seq:        6132
begin_lsn:            1268123
end_pkg_seq:          6191
end_lsn:              1274185

<backupset [DEVICE TYPE:DISK, BACKUP_PATH:
/home/dm_bak/DB_FULL_DAMENG_20190522_133248_000770] info end .>

show backupsets successfully.

time used: 249.062(ms)

```

这里获取的是单个备份集的元数据信息，也可以使用 `SHOW BACKUPSETS...INFO META` 命令批量获取元数据信息。同样地，我们可以使用类似的命令指定获取备份集的数据库信息、文件信息和表信息，或这些信息的任意组合。

### 以 xml 格式输出备份信息到文件

DMRMAN 可以将显示的备份集信息输出到文件，目前支持的格式包括 TXT 和 XML，默认为 TXT 文件格式。使用 `SHOW BACKUPSETS...TO 'file_path' FORMAT XML` 命令可将备份信息以 XML 格式显示并输出到文件，如下所示：

```

RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'

BACKUPSET'/home/dm_bak1/db_bak_for_xml_01';

RMAN>SHOW BACKUPSET'/home/dm_bak1/db_bak_for_xml_01' TO

'/home/dm_info/bkp_info.txt' FORMAT XML;

```

### 3.3.4.3 备份集校验

DMRMAN 中使用 CHECK 命令对备份集进行校验，校验备份集是否存在及合法。

语法如下：

**CHECK** BACKUPSET '<备份集目录>'

[DEVICE TYPE<介质类型> [PARMS '<介质参数>']][DATABASE '<INI\_PATH>']

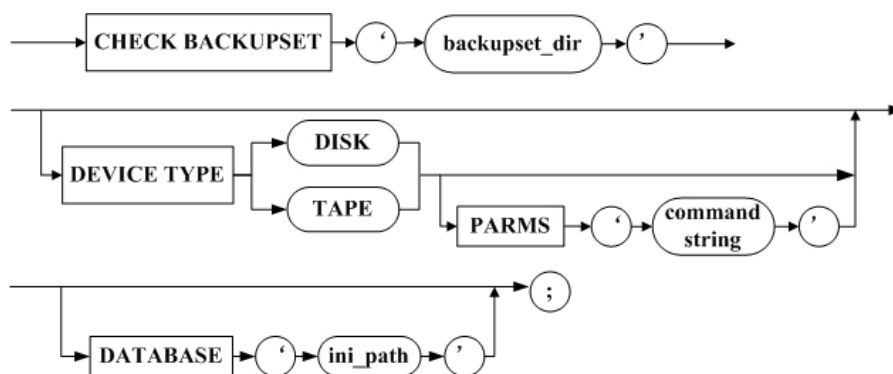
BACKUPSET: 指定目标校验备份集目录。

DEVICE TYPE: 指存储备份集的介质类型，支持 DISK 和 TAPE，默认 DISK，详见 1.3 介质管理层。

PARMS: 介质类型为 TAPE 时，第三方介质（tape 类型）管理实现所需的参数字符串。

INI\_PATH: 数据库 dm.ini 文件路径，若指定，则该数据库的默认备份目录作为备份集搜索目录之一。

图例：



#### 校验特定的备份集

CHECK BACKUPSET...命令用于校验特定备份集，每次只能检验一个备份集。

```

SQL>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'

BACKUPSET'/home/dm_bak/db_bak_for_check_01';

RMAN>CHECK BACKUPSET '/home/dm_bak/db_bak_for_check_01';

```

若备份集在默认备份路径下，可指定相对路径校验备份集，如下所示：

```

RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'

BACKUPSET'db_bak_for_check_02';

RMAN>CHECK BACKUPSET 'db_bak_for_check_02' DATABASE

'/opt/dmdbms/data/DAMENG/dm.ini';

```

### 3.3.4.4 备份集删除

备份集删除是备份管理的主要功能，本节主要对如何删除备份进行描述。内容包括：

- 概述
- 删除备份集

#### 3.3.4.4.1 概述

DMRMAN 中使用 REMOVE 命令删除备份集，可删除指定备份集，也可批量删除备份集。单个备份集删除时并行备份中的子备份集不允许单独删除；在指定备份集搜集目录中，发现存在引用目标备份集作为基备份的需要执行级联删除，默认报错。批量删除备份集时，跳过收集到的单独的子备份集。

语法如下：

---

```

REMOVE BACKUPSET '<备份集目录>'

[DEVICE TYPE< 介 质 类 型 > [PARMS '< 介 质 参 数 >']][<
database_bakdir_lst_stmt>][CASCADE];

REMOVE [

    DATABASE |

    TABLESPACE[<ts_name>] |

    TABLE "<schema_name>". "<tab_name>" |

    ARCHIVELOG|

    ARCHIVE LOG

    ] BACKUPSETS [<device_type_stmt>]<database_bakdir_lst_stmt>

{[UNTIL TIME '<截止时间串>'] | [BEFORE n]}

<device_type_stmt>::= DEVICE TYPE<介质类型> [PARMS '<介质参数>']

```

---

---

```
<database_bakdir_lst_stmt>::= DATABASE '<INI_PATH>' |  
WITH BACKUPDIR '<备份集搜索目录>' {, '<备份集搜索目录>' } |  
DATABASE '<INI_PATH>' WITH BACKUPDIR '<备份集搜索目录>' {, '<备份集搜索目录>' }
```

---

**BACKUPSET:** 指定待删除的备份集目录。

**DATABASE:** 指定数据库 dm.ini 文件路径，若指定，则该数据库的默认备份目录作为备份集搜索目录之一。

**DEVICE TYPE:** 指存储备份集的介质类型，支持 DISK 和 TAPE，默认 DISK。DISK 表示备份集存储介质磁盘，TAPE 表示存储介质为磁带。



**说明:** 目前 DM 的介质管理不支持 TAPE 类型介质的备份集删除，若使用支持此操作的第三方介质管理，则可指定 **DEVICE TYPE TAPE** 子句。

**PARMS:** 介质类型为 TAPE 时，第三方介质管理实现所需的参数字符串。

**CASCADE:** 当目标备份集已经被其他备份集引用为基备份集，默认不允许删除，若指定 CASCADE，则递归删除所有引用的增量备份。

**DATABASE | TABLESPACE | TABLE | ARCHIVELOG | ARCHIVE LOG:** 指定删除备份集的类型，分别为库级、表空间级、表级备份，以及归档级别，其中 ARCHVELOG 和 ARCHIVE LOG 等价。若不指定，全部删除。指定 TABLESPACE 时，若指定目标表空间名，则仅会删除满足条件的指定表空间名称的表空间备份集，否则，删除所有满足条件的表空间级备份集；指定 TABLE 时，若指定目标表名，则仅会删除满足条件的指定表名的表备份集；否则，删除所有满足条件的表备份集。

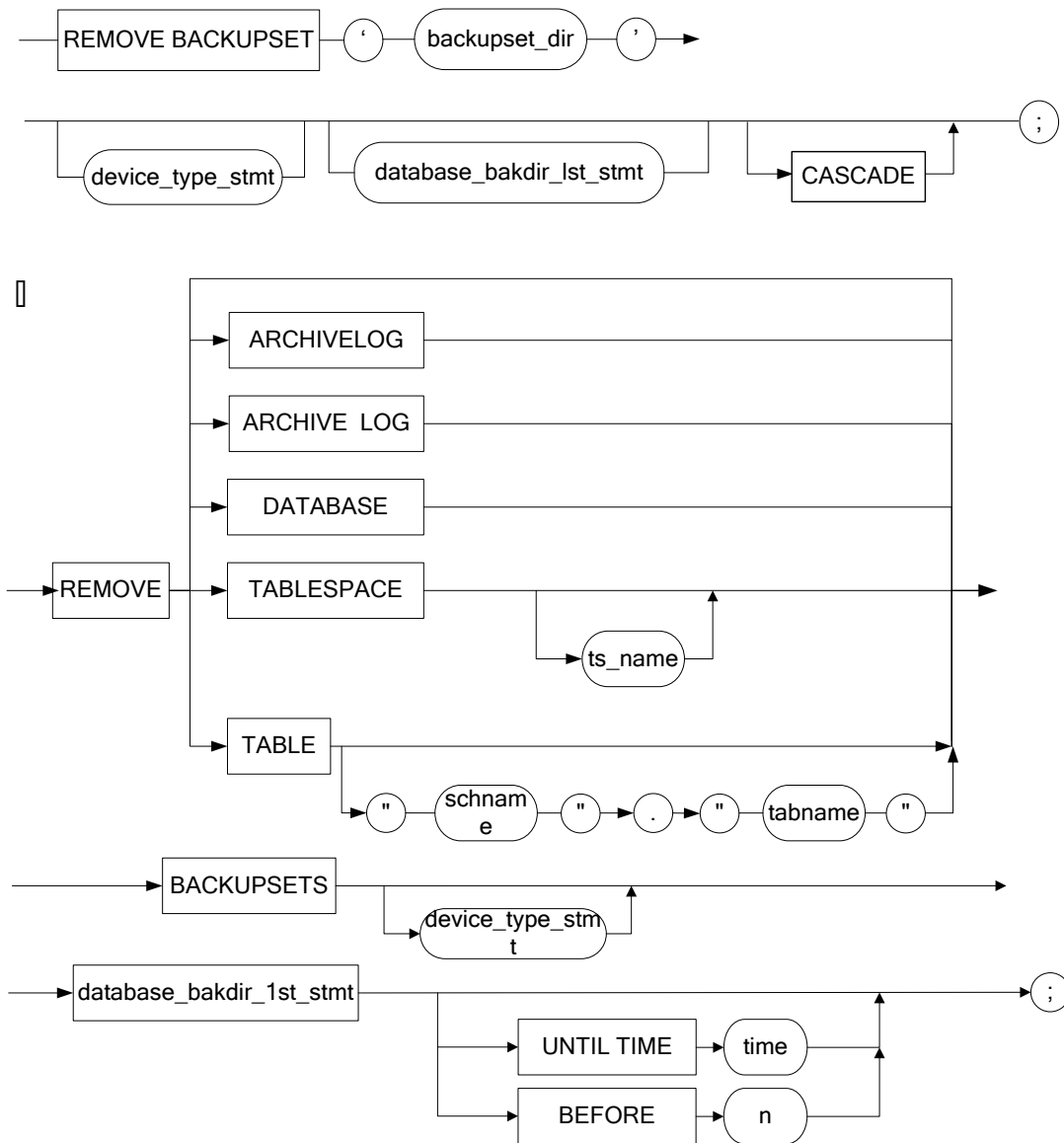
**UNTIL TIME:** 删除备份集生成的最大时间，即删除指定时间之前的备份集，若未指定，则删除所有备份集。

**BEFORE n:** 删除距离当前时间前 n 天产生的备份集；n 取值范围 0~365，单位：天。

**WITH BACKUPDIR:** 备份集搜索目录，用于搜索指定目录下的所有备份集。

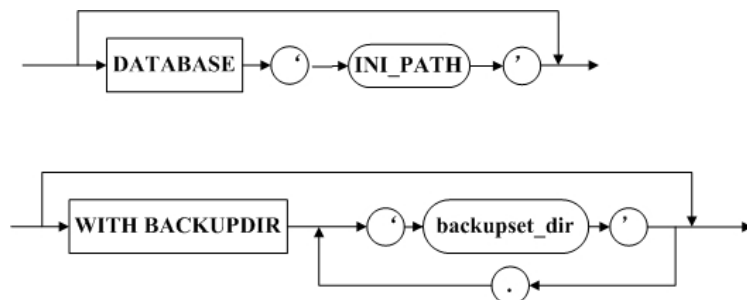
**图例:**





<device\_type\_stmt>: 请参考 3.3.4.2.1 [概述](#)

<database\_bakdir\_lst\_stmt>:



### 3.3.4.4.2 删除备份集

本节举例说明如何使用 DMRMAN 的 REMOVE 命令删除备份集，主要包括：

- 删除特定的备份集
- 批量删除所有备份集
- 批量删除指定时间之前的备份集

### 删除特定的备份集

使用 `REMOVE BACKUPSET...` 命令可删除特定备份集，每次只能删除一个备份集。若删除备份集已经被引用为其他备份集的基备份且未指定 `CASCADE`，则报错。

```
RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'
BACKUPSET '/home/dm_bak/db_bak_for_remove_01';
RMAN>REMOVE BACKUPSET '/home/dm_bak/db_bak_for_remove_01';
```

如果备份集在数据库默认备份目录下还可使用以下方式删除备份集：

```
RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini' BACKUPSET'db_bak_for_remove_01';
RMAN>REMOVE BACKUPSET 'db_bak_for_remove_01' DATABASE
'/opt/dmdbms/data/DAMENG/dm.ini';
```

如果备份集为其他备份集的基备份且备份集都在数据库默认备份目录下还可使用以下方式删除备份集：

```
RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini' BACKUPSET'db_bak_for_remove_01';
--增量备份之前，启动数据库，以便成功生成增量备份
RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini' INCREMENT
BACKUPSET'db_bak_for_remove_01_incr';
RMAN>REMOVE BACKUPSET 'db_bak_for_remove_01_incr' DATABASE
'/opt/dmdbms/data/DAMENG/dm.ini' CASCADE;
```

### 批量删除所有备份集

使用 `REMOVE BACKUPSETS...` 命令可批量删除备份集。批量删除可选择删除的备份类型，数据库备份、表空间备份、表备份，以及归档备份，不指定则全部删除。下面示例为删除 `/home/dm_bak` 目录下的所有备份集，可以是联机生成的备份集，也可以脱机 `DMRMAN` 工具生成的备份集。

```
RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'
BACKUPSET '/home/dm_bak/db_bak_for_remove_02';
RMAN>REMOVE BACKUPSETS WITH BACKUPDIR '/home/dm_bak';
```

### 批量删除指定时间之前的备份集

REMOVE BACKUPSETS...UNTIL TIME 命令用来批量删除指定时间的备份集。通常情况下，用户并不想删除指定目录下所有的备份集，这时可以选择只删除指定时间之前的备份。如何确定删除备份的时间点，需要结合用户的备份计划合理指定。若用户每周做一次完全备份，每天进行增量备份，那么删除的时间可指定为 7 天前的某个特定时间。假设今天的日期为 2019-6-8，要删除 7 天前 /home/dm\_bak 目录下的所有备份在 DMRMAN 中输入以下命令：

```
DMRMAN>REMOVE BACKUPSETS WITH BACKUPDIR '/home/dm_bak' UNTIL TIME'2019-6-1
00:00:00';
```

### 3.3.4.5 备份集导出

备份集导出是备份管理的主要功能，本节主要对如何导出备份集进行描述。内容包括：

- 概述
- 导出备份集

#### 3.3.4.5.1 概述

DMRMAN 中使用 LOAD 命令导出备份集。

语法如下：

---

```
LOAD BACKUPSETS FROM DEVICE TYPE<介质类型> [PARMS '<介质参数>'] [WITH BACKUPDIR '<
备份集搜索目录>' {, '<备份集搜索目录>'} ] TO BACKUPDIR '<备份集存放目录>';
```

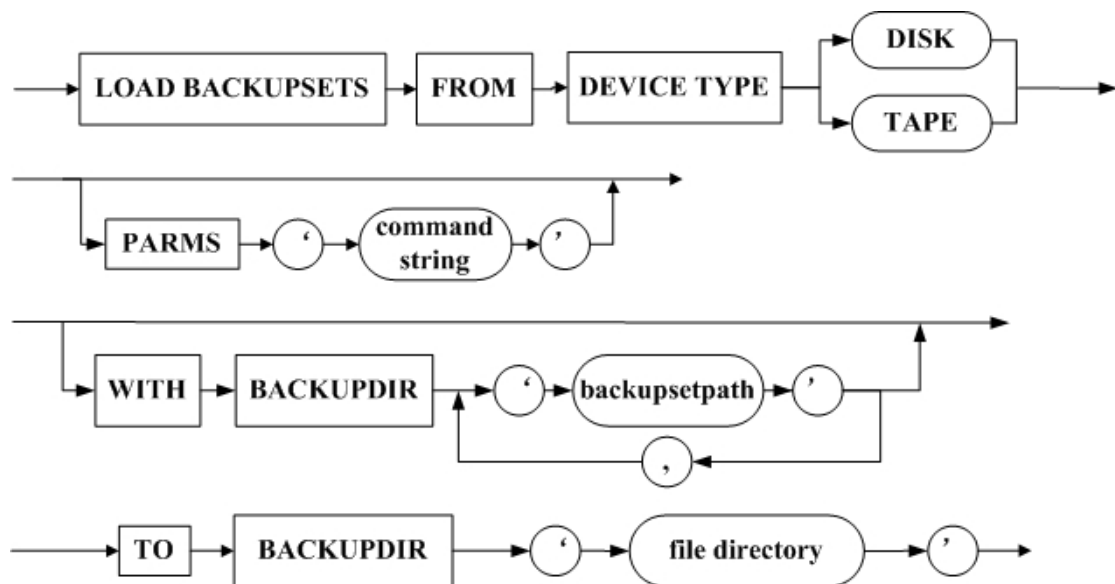
---

介质类型：指存储备份集的设备类型，暂支持 DISK 和 TAPE。

介质参数：介质类型为 TAPE 时，第三方介质管理实现所需的参数字符串。

备份集存放目录：指从 TAPE 上导出的备份集 meta 文件存放到本地磁盘的目标目录，要求为空或不存在。

图例：



### 3.3.4.5.2 导出备份集

本节举例说明如何使用 DMRMAN 的 LOAD 命令导出备份集，主要包括：

- 导出磁盘上备份集
- 导出磁带上所有的备份集

#### 导出磁盘上备份集

导出磁盘上备份集的 meta 文件到指定 /mnt/dmsrc/bak\_dir。

```
RMAN>LOAD BACKUPSETS FROM DEVICE TYPE DISK TO BACKUPDIR '/mnt/dmsrc/bak_dir';
```

#### 导出磁带上所有的备份集

导出磁带 /dev/nst0 上所有备份集的 meta 文件到目录 /mnt/hgfs/dmsrc/bak\_dir 中。

```
RMAN>LOAD BACKUPSETS FROM DEVICE TYPE TAPE TO BACKUPDIR '/mnt/hgfs/dmsrc/bak_dir';
LOAD BACKUPSETS FROM DEVICE TYPE TAPE TO BACKUPDIR '/mnt/hgfs/dmsrc/bak_dir';
load backupsets failed.error code: -10000
[-10000]: [错误码: -20022]磁带打开失败
```

退出 dmrman，设置环境变量 TAPE，值为 /dev/nst0。

```
[root@192 debug]# export TAPE=/dev/nst0
[root@192 debug]# echo $TAPE
/dev/nst0
```

启动 dmrman，再次执行：

```
[root@192 debug]# ./dmrman

dmrman V8.1.0.166-Build(2019.06.10-107924)ENT

RMAN>LOAD BACKUPSETS FROM DEVICE TYPE TAPE TO BACKUPDIR '/mnt/hgfs/dmsrc/bak_dir';

LOAD BACKUPSETS FROM DEVICE TYPE TAPE TO BACKUPDIR '/mnt/hgfs/dmsrc/bak_dir';

load meta file [SBT_TEST_T-20140909192629000000-4966] to path

[/mnt/hgfs/dmsrc/bak_dir/0/0.meta]...

load meta file [SBT_TEST_T-20140909192629000000-4966] to path

[/mnt/hgfs/dmsrc/bak_dir/0/0.meta]...success

load meta file [SBT_TEST_T2-20140909192746000000-9983] to path

[/mnt/hgfs/dmsrc/bak_dir/1/1.meta]...

load meta file [SBT_TEST_T2-20140909192746000000-9983] to path

[/mnt/hgfs/dmsrc/bak_dir/1/1.meta]...success

load backupsets successfully.
```

退出 dmrman，查看本地磁盘目录/mnt/hgfs/dmsrc/bak\_dir:

```
[root@192 debug]# ls -l /mnt/hgfs/dmsrc/bak_dir

total 0

drwxrwxrwx 1 root root 0 Sep 11 00:23 0

drwxrwxrwx 1 root root 0 Sep 11 00:23 1

[root@192 debug]# ls -l /mnt/hgfs/dmsrc/bak_dir/0

total 12

-rwxrwxrwx 1 root root 24576 Sep 11 00:23 0.meta

[root@192 debug]# ls -l /mnt/hgfs/dmsrc/bak_dir/1

total 12

-rwxrwxrwx 1 root root 24576 Sep 11 00:23 1.meta
```

### 3.3.4.6 备份集映射文件导出

备份集映射文件，又称为 mapped file。备份集映射文件导出，是备份管理的主要功能，是将备份集中各数据文件的原始路径或者调整后的路径生成到一个本地文件中，可通过关键字 MAPPED FILE 应用于表空间和库的还原操作中。本节主要对如何将备份集中数据

文件路径导出到本地进行描述。内容包括：

- 概述
- 备份集映射文件导出

### 3.3.4.6.1 概述

DMRMAN 中使用 DUMP 命令导出映射文件。不支持导出到 DMASM 文件系统中。

语法如下：

---

```
DUMP BACKUPSET '<备份集目录>'

[DEVICE TYPE<介质类型> [PARMS '<介质参数>']]

[DATABASE '<INI_PATH>' | TO '<SYSTEM_DIR>']

MAPPED FILE '<映射文件路径>';
```

---

**备份集目录：**待导出映射文件的目标备份集，仅支持库级和表空间级备份。

**介质类型：**指存储备份集的设备类型，暂支持 DISK 和 TAPE。

**介质参数：**介质类型为 TAPE 时，第三方介质管理实现所需的参数字符串。

**INI\_PATH：**备份集还原到目标库的 INI 路径。若指定，则根据 INI 对应库的系统目录调整数据文件路径；若不指定，则保持备份集中数据文件的原始路径。

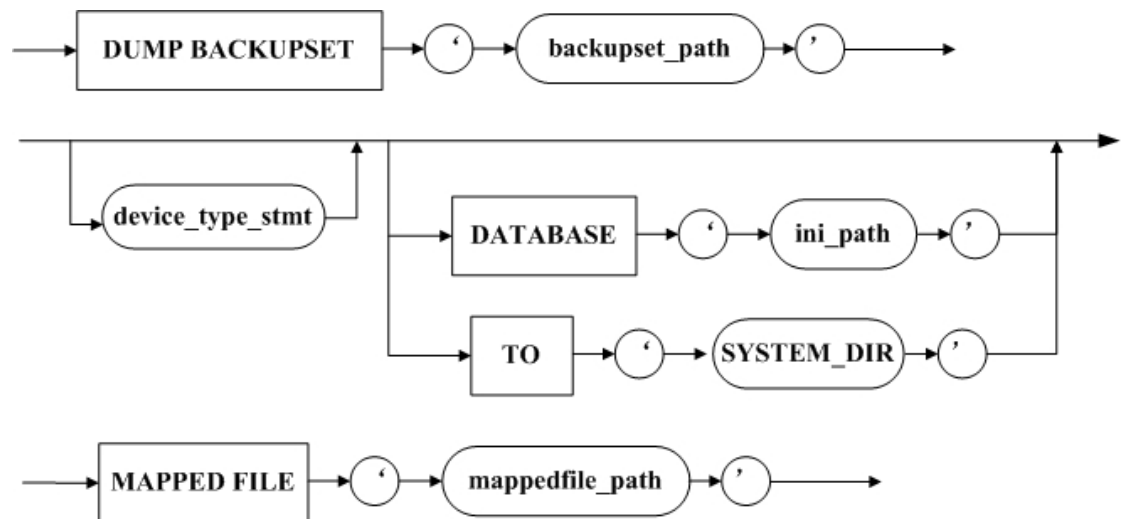
**SYSTEM\_DIR：**数据库目录中 SYSTEM.DBF 数据文件所在目录，作为数据库系统目录处理。



**数据文件路径指定 INI\_PATH 或者 SYSTEM\_DIR 调整策略：**若指定 **INI\_PATH**，则取出配置中 **SYSTEM\_PATH** 作为数据库系统目录，若指定 **SYSTEM\_DIR**，则直接作为数据库系统目录。对于库备份集，与还原过程（2.1.2.1 节）中数据库文件路径构造策略描述一致。对于表空间级备份集仅支持原库还原，若指定 **INI\_PATH** 或者 **SYSTEM\_DIR**，则认为需要构造，构造策略与库备份集中数据文件路径构造策略一致。

**映射文件路径：**输出到本地的目标映射文件路径。用户指定的文件不能为已经存在的文件，否则报错。文件生成之后，允许手动调整数据文件路径到其他路径，真正使用映射文件时，会再次校验。不支持导出到 DMASM 文件系统中。

图例：



### 3.3.4.6.2 导出备份集映射文件

本节举例说明如何使用 DMRMAN 的 DUMP 命令导出备份集数据文件路径到映射文件，主要内容包

- 导出原始路径
- 导出指定 INI\_PATH 调整后的路径

#### 导出原始路径

导出备份集中数据文件的原始路径。

```

RMAN>DUMP      BACKUPSET '/mnt/dmsrc/db_bak' DEVICE      TYPE      DISK      MAPPED      FILE
'/mnt/dmsrc/db_bak_mapped.txt';
    
```

#### 导出指定 INI\_PATH 调整后的路径

指定 INI\_PATH，导出调整后的数据文件路径到映射文件。

```

RMAN>DUMP      BACKUPSET '/mnt/dmsrc/db_bak' DEVICE      TYPE      DISKDATABASE
'/opt/dmdbms/data/DAMENG/dm.ini'MAPPED FILE  '/mnt/dmsrc/db_bak_mapped.txt';
    
```

## 3.3.5 数据库还原和恢复

本章节介绍在数据文件损坏后如何使用 DMRMAN 还原和恢复数据库，主要内容包

- 数据库还原
- 数据库恢复

### 3.3.5.1 执行数据库还原和恢复

#### 3.3.5.1.1 数据库还原

使用 RESTORE 命令完成脱机还原操作，在还原语句中指定库级备份集，可以是脱机库级备份集，或是联机库级备份集。数据库的还原包括数据库配置文件还原和数据文件还原，目前可能需要还原的数据库配置文件包括 dm.ini、dm.ct1、服务器秘钥文件 (dm\_service.private 或者 dm\_external.config，若备份库指定 usbkey 加密，则无秘钥文件)、联机日志文件。

语法如下：

---

```
RESTORE DATABASE <restore_type>[WITH CHECK] FROM BACKUPSET '<备份集目录>'

[DEVICE TYPE DISK|TAPE[PARMS '<介质参数>']]

[IDENTIFIED BY <密码> [ENCRYPT WITH <加密算法>]]

[WITH BACKUPDIR '<基备份集搜索目录>'{'<基备份集搜索目录>'}]

[MAPPED FILE '<映射文件>'] [TASK THREAD <任务线程数>]

[RENAME TO '<数据库名>'];

<restore_type>::=<type1>|<type2>

<type1>::='<INI 文件路径>' [REUSE DMINI] [OVERWRITE]

<type2>::= TO '<system_dbf 所在路径>' [OVERWRITE]
```

---

**DATABASE:** 指定还原库目标的 dm.ini 文件路径。

**WITH CHECK:** 指定还原前校验备份集数据完整性。缺省不校验。

**BACKUPSET:** 指定用于还原目标数据库的备份集目录。若指定为相对路径，会在默认备份目录下搜索备份集。

**DEVICE TYPE:** 指存储备份集的介质类型，支持 DISK 和 TAPE，默认为 DISK，详见 1.3 介质管理层。

**PARMS:** 介质参数，供第三方存储介质（TAPE 类型）管理使用。

**IDENTIFIED BY:** 指定备份时使用的加密密码，供还原过程解密使用。

**ENCRYPT WITH:** 指定备份时使用的加密算法，供还原过程解密使用，若未指定，则使用默认算法。

**WITH BACKUPDIR:** 指定备份集搜索目录。



**MAPPED FILE:** 指定存放还原目标路径的文件，参见 [3.3.5.2.1 数据库还原](#)。当<备份集目录>和<映射文件>指定的路径不一致时，以<映射文件>指定的路径为主

**TASK THREAD:** 指定还原过程中用于处理压缩和解密任务的线程个数。若未指定，则默认为 4；若指定为 0，则调整为 1；若指定超过当前系统主机核数，则调整为主机核数。

**RENAME TO:** 指定还原数据库后是否更改库的名字，指定时将还原后的库改为指定的数据库名，默认使用备份集中的 db\_name 作为还原后库的名称。

<restore\_type>:

■ <type1>

1>指定 dm.ini 还原，要求 dm.ini 中 CTL\_PATH 必须配置正确，且内容有效；若配置 CTL\_PATH 文件故障，且想利用 dm.ini 优化配置，则可选择 type2 还原后，用指定 dm.ini 覆盖还原后 dm.ini，并修改 CTL\_PATH 为当前控制文件路径即可；

2>除 dm.ini 文件外，其他文件均可不存在；但 dm.ini 参数配置必须正确，且配置的 dm.ctl 文件必须是有效的控制文件；

3>数据库配置文件中除已经存在的 dm.ini 外，先删除控制文件中的数据文件，然后根据 overwrite 选项，若指定 overwrite，其他文件（这些文件不在控制文件中，所以未删除）均采用删除重建的处理，避免存在非法的文件，否则如果这些文件已经存在，则报错；

4>若指定 REUSE DMINI，则会将备份集中备份的 dm.ini 中除路径相关的 INI 参数外，均拷贝到当前 dm.ini 上。

■ <type2>说明:

1>所有文件均可不在，system\_dbf 所在路径需为有效路径，若不存在，restore 过程中会自动创建；

2>所有前面提到到数据库配置文件均会在指定的 system\_dbf 所在路径还原，但非单机环境中相关其他文件均不修改或者重建，如 MPP 中 dmmpp.ini、dmmal.ini 等；

3>若未指定 OVERWRITE，若 system\_dbf 所在路径中存在待还原的库配置文件，则报错；若指定，则将已经存在的文件删除重建；

4>由于 DSC 环境中 dm.ini 可能存在多个，且可能不在一个主库上，或者即使在一个主库上也可能不在 system\_dbf 所在路径中，故暂时不支持 DSC 环境的指定目录还原。

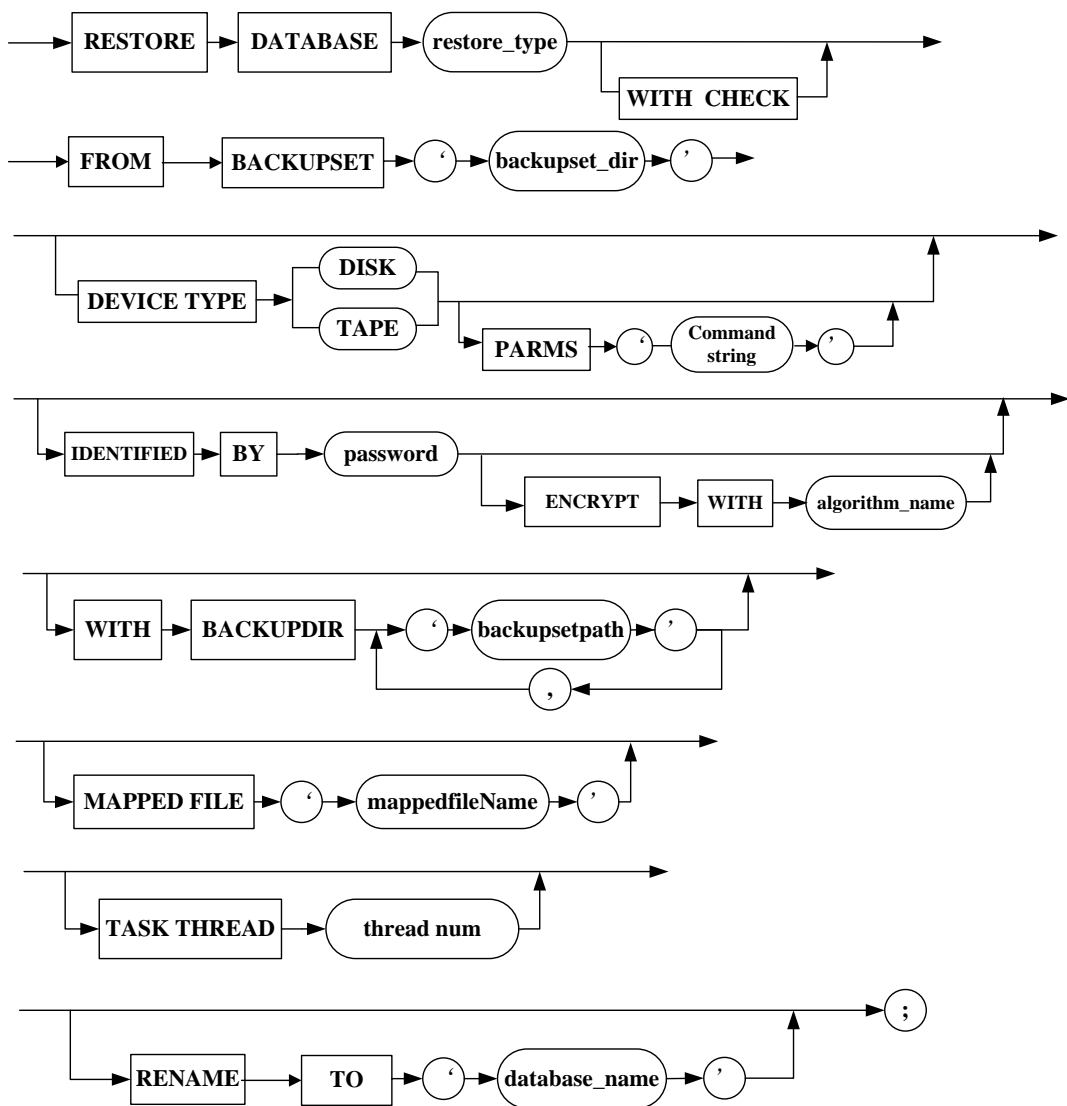
■ 联机日志

上述<type1>和<type2>中的还原后的联机日志文件至少会有两个，因为源库中的日

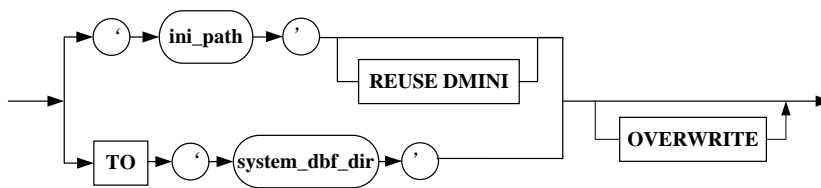
志文件可能大于等于两个，如果小于两个（被误删的情况）的则补齐为两个。已经存在的联机日志配置，使用原路径，若文件大小非法，则使用缺省大小 256M 重建；缺少的使用缺省命名和缺省大小 256M 重建。联机日志文件的命名规则：

- 1>单机：db\_name+(file\_id+1).log，其中 fil\_id + 1，占 2 个位置，如：db\_name=DAMENG，则相应的名称为 DAMENG01.log，DAMENG02.log；
- 2>DSC：dsc+(ep\_no + 1)\_(file\_id+1).log，其中 ep\_no + 1 和 file\_id+1 均占 2 个位置，如 0 号节点日志名称为 dsc01\_01.log，dsc01\_02.log。

图例：



restore\_type:



使用说明：

通过 RESTORE 命令还原后的数据库不可用，需进一步执行 RECOVER 命令，将数据库恢复到备份结束时的状态。

数据库备份集分为联机 and 脱机两种类型。通常情况下，用户会在联机的情况下备份数据库，因此下面以联机数据库备份为例说明使用 DMRMAN 如何执行数据库还原操作。

- 1) 联机备份数据库，保证数据库运行在归档模式及 OPEN 状态。

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_full_bak_for_restore';
```

- 2) 准备目标库。还原目标库可以是已经存在的数据库，也可使用 dminit 工具初始化一个新库。如下所示：

```
./dminit path=/opt/dmdbms/data db_name=DAMENG_FOR_RESTORE
```



如果还原目标库与故障库是同一个，建议先执行故障库的归档修复操作。

注意：具体详见 [3.3.5.4 归档修复](#)。

- 3) 校验备份，校验待还原备份集的合法性。校验备份有两种方式，联机和脱机，此处使用脱机校验。

```
RMAN>CHECK BACKUPSET '/home/dm_bak/db_full_bak_for_restore';
```

- 4) 还原数据库。启动 DMRMAN，输入以下命令：

```
RMAN>RESTORE DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' FROM
BACKUPSET '/home/dm_bak/db_full_bak_for_restore';
```



可通过调整 dm.ini 中的检查点和 REDO 日志相关参数，降低检查点频率，  
小窍门：增大 REDO 日志包大小来提升还原性能。

### 3.3.5.1.2 数据库恢复

使用 RECOVER 命令完成数据库恢复工作，可以是基于备份集的恢复工作，也可以是使用本地归档日志的恢复工作。如果还原后，数据已经处于一致性状态了，则可以使用更新

DB\_MAGIC 方式恢复，前提是不需要重做日志。

语法如下：

---

```
RECOVER DATABASE '<INI 文件路径>' <with_archdir_lst_stmt>

[USE DB_MAGIC <db_magic>] [UNTIL TIME '<时间串>'] [UNTIL LSN <LSN>]; |

RECOVER DATABASE '<INI 文件路径>' FROM BACKUPSET '<备份集目录>' [DEVICE TYPE
DISK|TAPE[PARMS '<介质参数>']] [IDENTIFIED BY <密码> [ENCRYPT WITH <加密算法>]];

<with_archdir_lst_stmt> ::=

WITH ARCHIVEDIR '<归档日志目录>' {, '<归档日志目录>' }
```

---

**DATABASE:** 指定还原库目标的 dm.ini 文件路径。

**USE DB\_MAGIC:** 指定本地归档日志对应数据库的 DB\_MAGIC，若不指定，则默认使用目标恢复数据库的 DB\_MAGIC。

**UNTIL TIME:** 恢复数据库到指定的时间点。如果指定的结束时间早于备份结束时间，忽略 UNTIL\_TIME 参数，重做所有小于备份结束 LSN（END\_LSN）的 REDO 日志，将系统恢复到备份结束时点状态。

**UNTIL LSN:** 恢复数据库到指定的 LSN。如果指定的 UNTIL\_LSN 小于备份结束 LSN（END\_LSN），则报错：恢复过程指定 UNTIL LSN\*\*\*小于备份结束 LSN\*\*\*。

**BACKUPSET:** 指定用于还原目标数据库的备份集目录。

**DEVICE TYPE:** 指存储备份集的介质类型，支持 DISK 和 TAPE，默认为 DISK，详见 1.3 介质管理层。

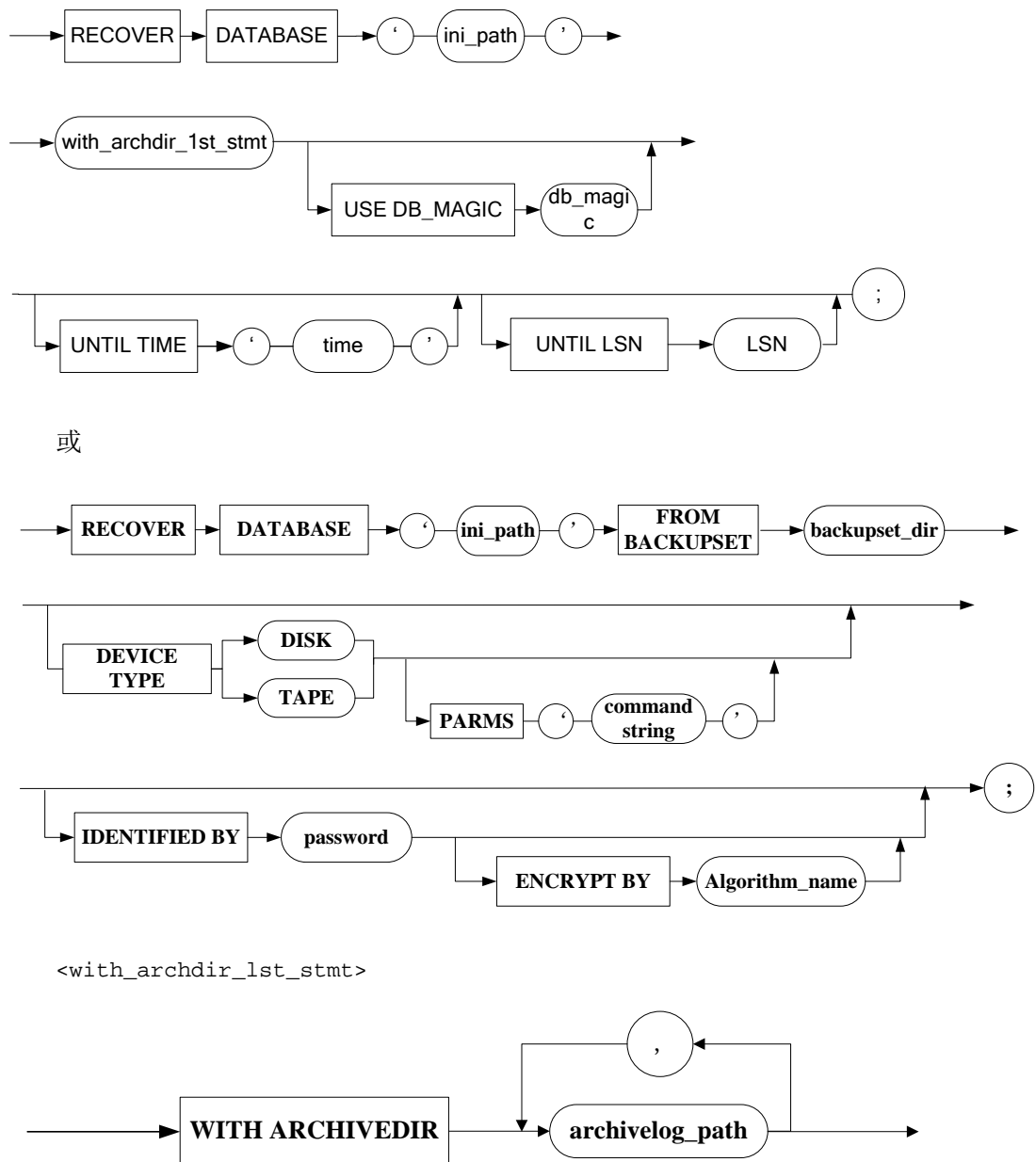
**PARMS:** 介质参数，供第三方存储介质（TAPE 类型）管理使用。

**IDENTIFIED BY:** 指定备份时使用的加密密码，供还原过程解密使用。

**ENCRYPT WITH:** 指定备份时使用的加密算法，供还原过程解密使用，若未指定，则使用默认算法。

**WITH ARCHIVEDIR:** 本地归档日志搜索目录。

图例：



数据库恢复是指重做 REDO 日志，有两种方式：从备份集恢复，即重做备份集中的 REDO 日志；从归档恢复，即重做归档中的 REDO 日志。数据库恢复后，还需要执行数据库更新操作，将数据库调整为可正常工作的库才算完成。下面逐一进行说明。

### 从备份集恢复

如果备份集在备份过程中生成了日志，且这些日志在备份集中有完整备份（如联机数据库备份），在执行数据库还原后，可以重做备份集中备份的日志，将数据库恢复到备份时的状态，即从备份集恢复。完整的示例如下：

- 1) 启动 DISql 联机备份数据库。

```
SQL>BACKUP DATABASE BACKUPSET
```

```
'/home/dm_bak/db_full_bak_for_recover_backupset';
```

2) 准备目标库，可以使用备份库，也可以重新生成库。重新生成库操作如下：

```
./dminit path=/opt/dmdbms/data db_name=DAMENG_FOR_RESTORE auto_overwrite=1
```

3) 启动 RMAN，校验备份。

```
RMAN>CHECK BACKUPSET '/home/dm_bak/db_full_bak_for_recover_backupset';
```

4) 还原数据库。

```
RMAN>RESTORE DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' FROM  
BACKUPSET '/home/dm_bak/db_full_bak_for_recover_backupset';
```

5) 恢复数据库。

```
RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' FROM  
BACKUPSET '/home/dm_bak/db_full_bak_for_recover_backupset';
```

### 从归档恢复

从归档恢复是利用重做本地归档日志来恢复数据的过程。从归档恢复允许恢复到指定的时间点及指定的 LSN 值。若同时指定了时间点和 LSN，则以较早的为结束点。



由于从本地归档恢复允许使用 `USE DB_MAGIC` 指定待收集归档的 `DB_MAGIC`，  
**警告：**那么就会存在跨库恢复情况，所以，归档日志的正确使用完全由用户保证。

因此，为了保证用户能正确使用从归档恢复，除了下文这两种情况，其他情况  
(可能导致数据被破坏) 不建议用户使用从本地归档恢复。

可以使用从归档恢复的情况：

1、执行过从备份集还原，以及执行过从备份集还原恢复的库，本地归档日志属于备份集备份时的库，本地归档日志的 `DB_MAGIC` 与备份集中记录 `DB_MAGIC` 值；

2、未执行过还原的库，但恢复目标库与本地归档日志来源于同一个库，且目标库与原来的库分离后没有再单独执行过操作或者故障重启过。比如一个数据库 A 因故障关闭后，拷贝一份作为数据库 B，此时数据库 A 和 B 完全相同。此后，数据库 A 故障重启，并正常执行其他操作，生成了新的本地归档日志。如果想利用 A 的本地归档日志去将 B 恢复到 A 的状态，那么可以利用 `DMRMAN` 工具执行从本地归档恢复。若中间 B 启动过或者执行过其他操作，则均不能再使用 A 的归档日志进行恢复操作。

利用归档恢复数据库至最新状态的完整示例如下：

1) 启动 `DISql` 联机备份数据库，以及备份从检查点开始的本地归档日志。

```
SQL>BACKUP ARCHIVELOG FROM LSN 421401 BACKUPSET
```

```
'/home/dm_bak/arch_bak_lsn_421401';
```

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_full_bak_for_recover_arch';
```



小窍门:

查看动态视图 V\$RLOG 中的 CKPT\_LSN 列, 可以得出当前检查点 LSN。如:

```
SQL> select CKPT_LSN from V$RLOG;

行号          CKPT_LSN
-----
1             421401
```

2) 准备目标库, 可以使用备份库, 也可以重新生成库。

如果使用原备份库, 且备份库故障, 需要先执行目标库归档修复:

```
RMAN>REPAIR ARCHIVELOG DATABASE 'opt/dmdbms/data/dm.ini';
```

如果使用新生成的库, 生成之后需要先重启一下数据库实例, 才可以被还原。重新生成的库不需要执行归档修复 (repair) 操作。重新生成库操作如下:

```
./dmnit path=/opt/dmdbms/data db_name=DAMENG_FOR_RESTORE auto_overwrite=1
```

3) 启动 RMAN, 校验备份。

```
RMAN>CHECK BACKUPSET '/home/dm_bak/db_full_bak_for_recover_arch';
```

4) 还原数据库。

```
RMAN>RESTORE DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' FROM
BACKUPSET '/home/dm_bak/db_full_bak_for_recover_arch';
```

5) 查看备份集的数据库信息, 获取源库的 DB\_MAGIC。



小窍门:

若还原后, 立即执行恢复, 可以不用获取源库 DB\_MAGIC。因为 DMRMAN

执行库级备份集还原后, 会将备份集中的 DB\_MAGIC 刷入还原后的库中。

```
RMAN>SHOW BACKUPSET '/home/dm_bak/db_full_bak_for_recover_arch' INFO DB;

show backupset '/home/dm_bak/db_full_bak_for_recover_arch' info db;

total 0 packages processed...
```

```
<backupset [DEVICE TYPE:DISK, BACKUP_PATH:
/home/dm_bak/db_full_bak_for_recover_arch] info start .....>
```

```
<DB INFO>
```

```

system path:          DAMENG
pmnt_magic:           1018589430
src_db_magic:          1447060265
db_magic:              1447060265
dsc node:              1
sys mode:              0
page check:           0
rlog encrypt:          0
external cipher[id/name]: 0/
external hash[id/name]: 0/
length in char:        0
use new hash:          1
page size:             8KB
extent size:           16
case sensitive:         1
log page size:         512B
unicode_flag/charset:  0
data version:          0x7000A
sys version:           V8.1.0.160-Build(2019.05.21-107084)ENT
enable policy:          0
archive flag:           1
blank_pad_mode:         0
crc_check:              TRUE

<backupset [DEVICE TYPE:DISK, BACKUP_PATH:
/home/dm_bak/db_full_bak_for_recover_arch] info end .>

show backupsets successfully.

time used: 51.969(ms)

```

6) 利用归档恢复数据库。由步骤 5 显示的备份集信息可知，源库的 DB\_MAGIC 值为 1447060265。



```
RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' WITH  
ARCHIVEDIR'/home/dm_arch/arch' USE DB_MAGIC 1447060265;
```



小窍门:

若还原后，立即执行恢复，可以不用指定 **DB\_MAGIC** 。

7) 若执行后归档恢复过程中，出现归档不足的错误，则利用归档校验工具 **dmrchk**，查看归档目录中归档连续性情况，然后再利用备份的本地归档日志，还原到归档目录后，再次执行恢复操作。

```
./dmrchk arch_path=/home/dm_arch/arch  
  
.....  
  
ARCH_PATH 的归档文件链表是不连续的。  
  
breakpoint 0:  
  
the left file /home/dm_arch/arch/ARCHIVE_LOCAL1_20180112161534890.log, clsn:  
421311  
  
the right file /home/dm_arch/arch/ARCHIVE_LOCAL1_20180113140135670.log,  
arch_lsn: 423208  
  
breakpoint 1:  
  
the left file /home/dm_arch/arch/ARCHIVE_LOCAL1_20180114085528401.log, clsn:  
424273  
  
the right file /home/dm_arch/arch/ARCHIVE_LOCAL1_20180114092356075.log,  
arch_lsn: 425285  
  
the rchk tool running cost 949.011 ms
```

DMRMAN 查看归档备份的范围:

```
RMAN>show backupset '/home/dm_bak/arch_bak_lsn_421401' INFO META;  
  
show backupset '/home/dm_bak/arch_bak_lsn_421401 ' INFO META;  
  
total 0 packages processed...  
  
<backupset [DEVICE TYPE:DISK, BACKUP_PATH: /home/dm_bak/arch_bak_lsn_421401]  
info start .....>
```

```

<META INFO>

backupset sig:          BA

backupset version:      0x4009

database name:          DAMENG

backup name:            ARCH_20190523_101415_000381

backupset description:

n_magic:                0x257F3E68

parent n_magic:         0xFFFFFFFF

meta file size :       82432

compressed level:       0

encrypt type:           0

parallel num:           1

backup range:           archivelog

mpp_timestamp:          1558577655

arch start lsn:         421300

arch end lsn:           425550

backup level:           online

backup type:             archive

without log:            TRUE

end_lsn:                425550

base begin_lsn:         -1

base end_lsn:           -1

base n_magic:           0xFFFFFFFF

base name:

base backupset:

backup time:            2019-05-23 10:14:17

min exec ver:           0x0701060C

pkg size:               0x02000000

<EP INFO>

```

```

EP[0]:

begin_pkg_seq:      6526

begin_lsn:          425450

end_pkg_seq:        6585

end_lsn:            425550


<backupset [DEVICE TYPE:DISK, BACKUP_PATH: /home/dm_bak/arch_bak_lsn_421401]

info end .>

show backupsets successfully.

time used: 45.602(ms)

```

可见本地归档缺失部分(421311,423208]、(424273,425285]在归档备份范围内[421300,425550]中,执行归档还原后,再次执行利用本地归档恢复操作。

```

RMAN>RESTORE ARCHIVELOGFROM BACKUPSET '/home/dm_bak/arch_bak_lsn_421401' ALL TO
ARCHIVEDIR '/home/dm_arch/arch';

```

### 3.3.5.1.3 数据库更新

数据库更新是指更新数据库的 DB\_MAGIC,并将数据库调整为可正常工作状态,与数据库恢复一样使用 RECOVER 命令完成。数据库更新发生在重做 REDO 日志恢复数据库后,或者目标库不需要执行重做日志已经处于一致状态的情况。

语法如下:

---

```
RECOVER DATABASE '<INI 文件路径>' UPDATE DB_MAGIC;
```

---

DATABASE: 指定还原库目标的 dm.ini 文件路径。

图例:



以使用正常退出数据库的脱机备份的备份集还原为完整示例,进行说明如下:

- 1) 启动 RMAN 备份数据库,保证数据库处于正常退出的脱机状态。

```

RMAN>BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini' BACKUPSET

```

```
 '/home/dm_bak/db_full_bak_for_recover_dbmagic';
```

- 2) 准备目标库，可以使用备份库，也可以重新生成库或者直接指定目录还原。重新生成库操作如下：

```
./dminit path=/opt/dmdbms/data db_name=DAMENG_FOR_RESTORE auto_overwrite=1
```

- 3) 校验备份。

```
RMAN>CHECK BACKUPSET '/home/dm_bak/db_full_bak_for_recover_dbmagic';
```

- 4) 还原数据库。

```
RMAN>RESTORE DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' FROM  
BACKUPSET '/home/dm_bak/db_full_bak_for_recover_dbmagic ';
```

- 5) 恢复数据库。

```
RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' FROM  
BACKUPSET '/home/dm_bak/db_full_bak_for_recover_dbmagic ';
```

- 6) 更新数据库。

```
RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' UPDATE  
DB_MAGIC;
```

### 3.3.5.2 执行数据库还原和恢复：高级场景

#### 3.3.5.2.1 数据库还原

本节主要介绍数据库还原的高级过程，指定映射文件还原。

##### 指定映射文件还原

还原后的数据文件默认地生成到还原目标库的路径下，如果用户想生成数据文件到特定的路径，就需要指定映射文件参数来实现。

映射文件（mappedfile）用于指定存放还原目标路径，即备份集里面的数据文件的路径。可以手动修改自动生成的映射文件。当参数 BACKUPSET 和 MAPPED FILE 指定的路径不一致时，以 MAPPED FILE 中指定的路径为主。映射文件可用于库级脱机还原和表空间还原。

使用 DUMP 命令可以将指定备份集还原目标信息生成到目标映射文件中，该文件可被重新编辑后，用于数据库的还原过程。

语法如下：

---

```
DUMP BACKUPSET '<备份集目录>' [DEVICE TYPE DISK|TAPE [PARMS '介质参数']]
```

```
[DATABASE '<INI 文件路径>'] MAPPED FILE '<映射文件>';
```

---

BACKUPSET：待生成映射文件的备份集目录。

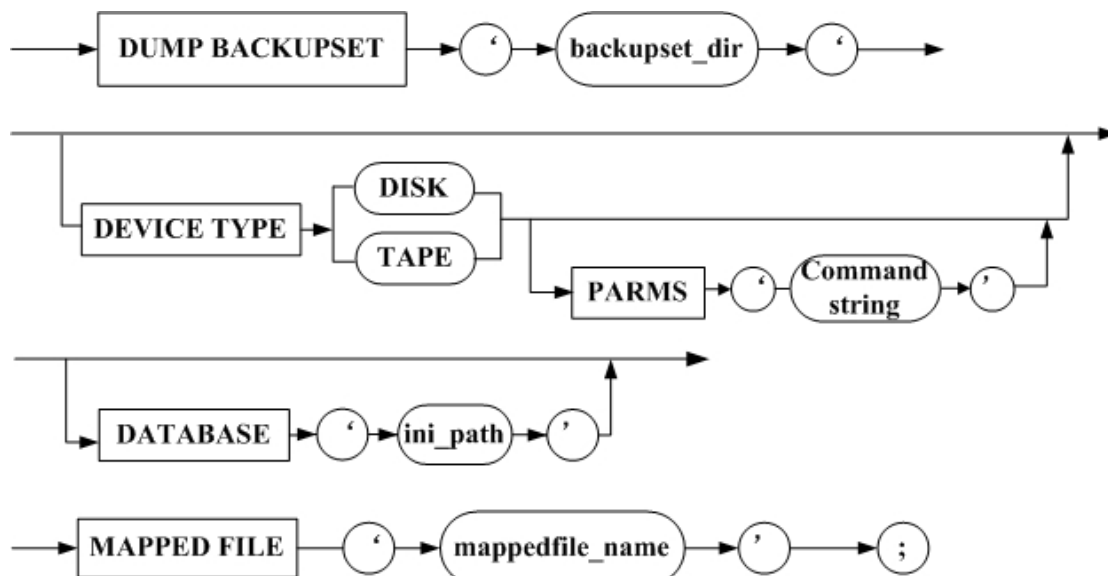
DEVICE TYPE：指存储备份集的介质类型，支持 DISK 和 TAPE，默认为 DISK。见 1.3 介质管理层。

PARMS：介质参数，供第三方存储介质（TAPE 类型）管理使用。

DATABASE：指定目标还原库的 dm.ini 文件路径。

MAPPED FILE：生成映射文件路径。若指定 DATABASE 参数，则生成内容调整为与指定数据库相适应的数据文件目标还原路径；否则，仅将备份集中备份时记录的路径输出。

图例：



下面以脱机备份还原为例说明使用映射文件还原的具体步骤。

1) 启动 RMAN，脱机备份数据库。

```
RMAN> BACKUP DATABASE '/opt/dmdbms/data/DAMENG/dm.ini'
BACKUPSET'/home/dm_bak/db_bak_for_map_01';
```

2) 生成映射文件 map\_file\_01.txt 至/home/dm\_mapfile 目录。此处指定生成映射文件中的数据文件路径与数据库/opt/dmdbms/data/DAMENG\_FOR\_RESTORE 中的数据文件一致。若不指定，与备份集中的源数据库的数据文件路径一致。

```
RMAN> DUMP BACKUPSET'/home/dm_bak/db_bak_for_map_01' DATABASE
'/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' MAPPED
```

```
FILE '/home/dm_mapfile/map_file_01.txt';
```

生成的映射文件内容如下：

```
/*
*****
Delete the unnecessary modified groups
Modify the data_path or mirror_path only in one group
*****
**=====**
/*[DAMENG_SYSTEM_FIL_0]*/
fil_id      = 0
ts_id       = 0
ts_name     = SYSTEM
data_path   = /opt/dmdbms/data/DAMENG_FOR_RESTORE/SYSTEM.DBF
mirror_path =

**=====**
/*[DAMENG_ROLL_FIL_0]*/
fil_id      = 0
ts_id       = 1
ts_name     = ROLL
data_path   = /opt/dmdbms/data/DAMENG_FOR_RESTORE/ROLL.DBF
mirror_path =

**=====**
/*[DAMENG_MAIN_FIL_0]*/
fil_id      = 0
ts_id       = 4
ts_name     = MAIN
data_path   = /opt/dmdbms/data/DAMENG_FOR_RESTORE/MAIN.DBF
mirror_path =

```

```

/***** END *****/

```

如果需要恢复数据文件或镜像文件到指定路径，可手动编辑映射文件中表空间对应的 data\_path 属性。例如，要还原 MAIN 表空间中的数据文件 MAIN.DBF 到 /home/dm\_dbf 路径下，修改组 DAMENG\_MAIN\_FIL\_0 的内容如下：

```

/*[DAMENG_MAIN_FIL_0]*/

fil_id      = 0

ts_id       = 4

ts_name     = MAIN

data_path   = /home/dm_dbf/MAIN.DBF

```

3) 指定映射文件还原。还原前可选择对备份文件进行校验。

```

RMAN> RESTORE DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' FROM
BACKUPSET '/home/dm_bak/db_bak_for_map_01' MAPPED
FILE '/home/dm_mapfile/map_file_01.txt';

```

### 3.3.5.2.2 数据库恢复

前面章节我们已经对数据库恢复中的更新 DB\_MAGIC、从备份集恢复和从归档恢复作了简单介绍。本节将介绍数据库恢复中比较典型且复杂的场景，主要包括：

- 恢复数据库到指定时间点/LSN
- 主备环境下指定 DB\_MAGIC 收集归档
- DSC 环境下的数据库恢复
- 多次故障恢复后使用不同数据库归档恢复

#### 恢复数据库到指定时间点/LSN

恢复数据库到指定时间点/LSN 是从归档恢复的一种方式，也称为不完全恢复。从归档恢复允许恢复到指定的时间点及指定的 LSN 值。若同时指定了时间点和 LSN，则以较早的为结束点。用户可以通过指定一个时间点/LSN，使数据库恢复到这个指定的时间点/LSN。

例如，用户在下午 5 点做了一个误操作，删除了某些重要数据；我们可以指定恢复时间点到下午 4:59 分，恢复被误删除的数据。

下面以联机数据库备份为例说明如何恢复数据库到指定的时间点/LSN。

1) 准备数据。

```
SQL>CREATE TABLE TAB_FOR_RECOVER_01(C1 INT);  
  
SQL>INSERT INTO TAB_FOR_RECOVER_01 VALUES(1);  
  
SQL>COMMIT;
```

2) 备份数据库。

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_full_bak_for_time_lsn';
```

3) 正确操作数据库，产生一些归档。

```
SQL>CREATE TABLE TAB_FOR_RECOVER_02(C1 INT);  
  
SQL>INSERT INTO TAB_FOR_RECOVER_02 VALUES(1);  
  
SQL>COMMIT;
```

使用 SELECT SYSDATE 命令查询此时的时间为：2018-11-16 10:56:40.624931；

使用 SELECT FILE\_LSN FROM V\$RLOG 命令查询此时的 LSN 为：50857

4) 误操作数据库。此步骤误删除了表 TAB\_FOR\_RECOVER\_01 中数据。

```
SQL>DELETE FROM TAB_FOR_RECOVER_01;  
  
SQL>COMMIT;
```

操作步骤同步骤 3)，此时的时间和 LSN 分别为：2018-11-16 10:57:20.977265、50861。

5) 还原数据库。步骤 4) 为误操作，因此我们需要将数据库恢复到步骤 3) 的状态。首先需要还原数据库：

```
RMAN> RESTORE DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' FROM  
BACKUPSET '/home/dm_bak/db_full_bak_for_time_lsn';
```

6) 恢复数据库到指定时间点/LSN。还原后数据库的数据与备份时一致，如果要恢复数据库至步骤 3) 的状态可以指定 UNTIL TIME 或 UNTIL LSN 参数重做部分归档。

使用 RECOVER DATABASE...UNTIL TIME 命令恢复到指定的时间：

```
RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' WITH  
ARCHIVEDIR'/home/dm_arch/arch' UNTIL TIME'2018-11-16 10:56:40.624931';
```

或者使用 RECOVER DATABASE...UNTIL LSN 命令恢复到指定的 LSN：

```
RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' WITH  
ARCHIVEDIR'/home/dm_arch/arch' UNTIL LSN 50857;
```

### 主备环境下指定 DB\_MAGIC 收集归档

主备环境下，如果当前节点的归档缺失，使用归档进行恢复时，会由于归档日志不连续



而报错，无法将数据库恢复到最新状态。如果另一节点保存有该部分日志，则可以使用该节点日志进行恢复。默认情况下，恢复阶段收集归档时只收集与待恢复库 DB\_MAGIC、PERMANENT\_MAGIC 一致的归档文件，而在主备环境中 PERMANENT\_MAGIC 是一致的，此时可通过指定 DB\_MAGIC 跳过这一限制。

- 1) 搭建主备环境并备份任意节点。

```

RMAN>BACKUP DATABASE '/opt/dmdbms/data0/DAMENG/dm.ini' BACKUPSET
'/home/dm_bak/db_full_bak';

```

- 2) 启动主备，登录主节点插入部分数据后退出，然后删除该节点下所有归档日志。
- 3) 指定备份集还原到主节点。

```

RMAN>RESTORE DATABASE '/opt/dmdbms/data0/DAMENG/dm.ini' FROM BACKUPSET
'/home/dm_bak/db_full_bak';

```

- 4) 指定 DB\_MAGIC，使用备库归档进行恢复。由于该节点下所有归档已被删除，无法恢复到最新状态，此时需要借助备库存在的归档。

```

RMAN>RECOVER DATABASE '/opt/dmdbms/data0/DAMENG/dm.ini' WITH ARCHIVEDIR
'/opt/dmdbms/data1/arch' USE DB_MAGIC 1447060264;

```

### DMDSC 环境下的数据库恢复

DM 支持 DMDSC 环境下的备份还原。DMDSC 的备份还原与普通的单机不同的是，它包含多个节点，有关 DMDSC 的详细介绍及环境搭建参见《DM8 共享存储集群》。下面以从归档恢复为例说明 2 节点（DSC0、DSC1）DMDSC 环境下的备份恢复：

- 1) 搭建 DMDSC 环境，每个节点都需要配置双向的远程归档。归档配置示例如下：  
DSC0 实例的 dmarch.ini 配置：

```

[ARCHIVE_LOCAL1]

ARCH_TYPE          = LOCAL

ARCH_DEST          = /dmdata/dameng/arch_dsc0

ARCH_FILE_SIZE     = 128

ARCH_SPACE_LIMIT   = 0

[ARCH_REMOTE1]

ARCH_TYPE          = REMOTE

ARCH_DEST          = DSC1

ARCH_INCOMING_PATH = /dmdata/dameng/arch_dsc1

```

```
ARCH_FILE_SIZE      = 128
```

```
ARCH_SPACE_LIMIT    = 0
```

DSC1 实例的 dmarch.ini 配置:

```
[ARCHIVE_LOCAL1]
```

```
ARCH_TYPE           = LOCAL
```

```
ARCH_DEST           = /dmdata/dameng/arch_dsc1
```

```
ARCH_FILE_SIZE      = 128
```

```
ARCH_SPACE_LIMIT    = 0
```

```
[ARCH_REMOTE1]
```

```
ARCH_TYPE           = REMOTE
```

```
ARCH_DEST           = DSC0
```

```
ARCH_INCOMING_PATH  = /dmdata/dameng/arch_dsc0
```

```
ARCH_FILE_SIZE      = 128
```

```
ARCH_SPACE_LIMIT    = 0
```



远程归档必须双向配置, 单向配置时目标实例上不会接收归档日志, 归档状

注意: 态将会变成无效状态。

2) 启动 Disql, 联机备份数据库。备份其中任意一个节点即可备份整个 DMDSC 环境。

```
SQL>BACKUP DATABASE BACKUPSET '/home/dm_bak/db_full_bak_for_dsc';
```

3) 还原数据库。还原数据库之前可选择对备份文件进行校验。需要注意的是, 待还原的目标库可以是单机库, 也可以是 DMDSC 库, 且节点个数允许不同。

```
RMAN> RESTORE DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' FROM  
BACKUPSET '/home/dm_bak/db_full_bak_for_dsc';
```

4) 恢复数据库。DMDSC 库恢复要求各节点归档完整性由用户保证, 即各节点的本地归档都能够访问到, 若本地存在 REMOTE 归档, 则可以使用 REMOTE 归档代替远程节点的本地归档。

```
RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' WITH  
ARCHIVEDIR '/dmdata/dameng/arch_dsc0', '/dmdata/dameng/arch_dsc1';
```

5) 数据库更新

```
RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' UPDATE  
DB_MAGIC
```

### 多次故障恢复后使用不同数据库归档恢复

在实际应用中可能会遇到以下还原场景：

- 1) 创建一个数据库 D1。
- 2) 操作数据库并执行数据库备份 B1。
- 3) 继续操作数据库的过程中数据库故障，此时生成的归档为 A1。
- 4) 利用备份 B1 和归档 A1 将数据库 D1 恢复到目标库 D2，此时数据库 D2 为 D1 故障前的状态。
- 5) 启动数据库 D2，操作数据库过程中数据库第二次故障，此时生成的归档为 A2。

归档 A1、A2 属于不同的数据库，使用备份 B1 和归档 A1、A2 是否可以恢复数据库到第二次故障发生前的状态呢？答案是肯定的。下面将举例详细说明如何使用不同数据库的归档恢复数据库到最新状态。具体步骤如下：

- 1) 创建源库 D1，即待备份的数据库，然后启动数据库并配置归档，归档目录为 /home/dm\_arch/arch。

```
./dmunit path=/opt/dmdbms/data  
./dmserver /opt/dmdbms/data/DAMENG/dm.ini  
--连接数据库配置归档  
SQL>ALTER DATABASE MOUNT;  
SQL>ALTER DATABASE ADD ARCHIVELOG 'DEST = /home/dm_arch/arch, TYPE = local,  
FILE_SIZE = 1024, SPACE_LIMIT = 2048';  
SQL> ALTER DATABASE ARCHIVELOG;  
SQL>ALTER DATABASE OPEN;
```

- 2) 操作数据库的同时备份数据库，备份集为 B1。此处以向表中循环插入数据为例来操作数据库。

```
SQL>DROP TABLE TAB_FOR_RECOVER;  
SQL>CREATE TABLE TAB_FOR_RECOVER(C1 INT);  
SQL>BEGIN  
FOR I IN 1..100000 LOOP  
INSERT INTO TAB_FOR_RECOVER VALUES(I);
```

```

        COMMIT;

    END LOOP;

END;

/

--插入数据的同时，另外一个会话备份数据库

SQL>BACKUP DATABASE FULL TO B1 BACKUPSET '/home/dm_bak/B1' DEVICE TYPE DISK

BACKUPINFO 'DAMENG FULL BACKUP ONLINE' MAXPIECESIZE 2048;

```

3) 继续向数据库插入数据一段时间后关掉数据库实例，模拟数据库故障。从备份后到数据库故障这段时间的生成的归档称为 A1。

4) 创建目标库 D2，即待还原的库，并配置归档（参考 3.1.2 [归档配置](#)），归档目录与数据库 D1 相同。

```
./dminit path=/opt/dmdbms/data db_name=DAMENG_FOR_RES
```

5) 启动 DMRMAN 工具，利用备份集 B1 和归档 A1 恢复数据库 D2 到 D1 故障前的状态。

```

RMAN>RESTORE DATABASE '/opt/dmdbms/data/DAMENG_FOR_RES/dm.ini' FROM BACKUPSET

'/home/dm_bak/B1';

RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RES/dm.ini' WITH ARCHIVEDIR

'/home/dm_arch/arch';

RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RES/dm.ini' UPDATE DB_MAGIC;

```

启动目标库 D2，向数据库中继续插入数据。

```

./dmserver /opt/dmdbms/data/DAMENG/dm.ini

SQL>BEGIN

    FOR I IN 1..100000 LOOP

        INSERT INTO TAB_FOR_RECOVER VALUES(I);

        COMMIT;

    END LOOP;

END;

/

```

6) 插入数据一段时间后，关掉数据库实例，模拟数据库第二次故障。数据库 D2 启动到第二次故障之间产生的归档为 A2。

7) 恢复目标库 D2 到第一次故障前的状态。恢复一次，目标库的 DB\_MAGIC 都会改变，

因此归档 A1、A2 的 DB\_MAGIC 不同,属于不同的数据库。第一次恢复只能先重做归档 A1,即恢复到第一次故障前的状态。

```
RMAN>RESTORE DATABASE '/opt/dmdbms/data/DAMENG_FOR_RES/dm.ini' FROM BACKUPSET
'/home/dm_bak/B1';

RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RES/dm.ini' WITH ARCHIVEDIR
'/home/dm_arch/arch';
```

### 8) 查看归档 A2 的 DB\_MAGIC。

```
./dmrchk arch_fil=/home/dm_arch/arch/ARCHIVE_LOCAL1_2019061208085947282.log
rchk V8.1.0.166-Build(2019.06.10-107924)ENT

/*****/

归档文件/home/dm_arch/arch/ARCHIVE_LOCAL1_2019061208085947282.log 明细.

Version      : 0x7005

Status       : INACTIVE

n_rpags      : 7369

pemnt_magic  : 1449476774

db_magic     : 1449535319

src_db_magic : 2032790579

arch_lsn     : 39891

arch_seq     : 0

clsn        : 76829

next_seq     : 7368

file len     : 3777024

file free    : 3777024

create time  : 2019-06-12 8:59:47

close time   : 2019-06-12 8:59:47

crc_check    : TRUE

/*****/

概要 (节点[0]) :

总计: 1 个文件

正确: 1 个文件
```

错误: 0 个文件

重复: 0 个文件

the rachk tool running cost 27.161 ms

9) 利用归档 A2 恢复数据库至第二次故障前的状态。从归档恢复时, 默认使用目标库的 DB\_MAGIC, 此时目标库 DB\_MAGIC 值被替换为 B1 备份集源库 DB\_MAGIC, 与归档 A2 DB\_MAGIC 值不一致, 无法利用该部分归档恢复到最新状态。此时可使用 USE DB\_MAGIC 语法, 指定 DB\_MAGIC 值为待恢复归档的 DB\_MAGIC 值即可。

```
RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RES/dm.ini' WITH ARCHIVEDIR  
'/home/dm_arch/arch' USE DB_MAGIC 1449535319 ;  
  
file dm.key not found, use default license!  
  
start redo arch file: /home/dm_arch/arch/ARCHIVE_LOCAL1_2019061208085947282.log,  
rpages: 7377, start lsn: 39891, end lsn: 76829  
  
total redo pages:7369  
  
end redo arch file: /home/dm_arch/arch/ARCHIVE_LOCAL1_2019061208085947282.log  
used time: 0s  
  
recover successfully!  
recover time used: 7578.080(ms)
```

10) 更新数据库。

```
RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RES/dm.ini' UPDATE DB_MAGIC;
```

### 3.3.6 表空间还原和恢复

#### 3.3.6.1 执行表空间还原和恢复

##### 3.3.6.1.1 表空间还原

使用 RESTORE 命令完成表空间的脱机还原, 还原的备份集可以是联机或脱机生成的库

备份集，也可以是联机生成的表空间备份集。脱机表空间还原仅涉及表空间数据文件的重建与数据页的拷贝。不需要事先置目标表空间为 OFFLINE 状态。

表空间还原后，表空间状态被置为 RES\_OFFLINE，并设置数据标记 FIL\_TS\_RECV\_STATE\_RESTORED，表示已经过还原但数据不完整。

语法如下：

---

```
RESTORE DATABASE <INI 路径> TABLESPACE <表空间名> [WITH CHECK]
[DATAFILE<<文件编号> {,<文件编号>} | '<文件路径>' {,'<文件路径>'}>]
FROM BACKUPSET '<备份集路径>' [DEVICE TYPE <介质类型> [PARMS '<介质参数>']]
[IDENTIFIED BY <加密码>] [ENCRYPT WITH <加密算法>]
[WITH BACKUPDIR '<备份目录>' {,'<备份目录>'}]
[MAPPED FILE '<映射文件>']
[TASK THREAD <线程数>];
```

---

**DATABASE:** 指定还原库目标的 dm.ini 文件路径。

**TABLESPACE:** 指定还原的表空间，除了 temp 表空间。

**WITH CHECK:** 指定还原前校验备份集数据完整性。缺省不校验。

**DATAFILE:** 还原指定的数据文件。可以指定数据文件编号或数据文件路径。文件编号，对应动态视图 V\$DATAFILE 中 ID 列的值；文件路径，对应动态视图 V\$DATAFILE 中 PATH 或者 MIRROR\_PATH 列的值，也可以仅指定数据文件名称（相对路径），与表空间中数据文件匹配时，会使用 SYSTEM 目录补齐。

**BACKUPSET:** 指定还原备份集的路径。若指定为相对路径，会在默认备份目录下搜索备份集。

**DEVICE TYPE:** 指存储备份集的介质类型，支持 DISK 和 TAPE，默认 DISK，详见 1.3 介质管理层。

**PARMS:** 介质参数，只对介质类型为 TAPE 时有效。

**IDENTIFIED BY:** 加密备份表空间时，用户设置的密码。密码长度为 9 到 48 个字节。

**ENCRYPT WITH:** 加密算法。缺省情况下，算法为 AES256\_CFB。具体包含哪些加密算法请参考表空间备份的参数说明。

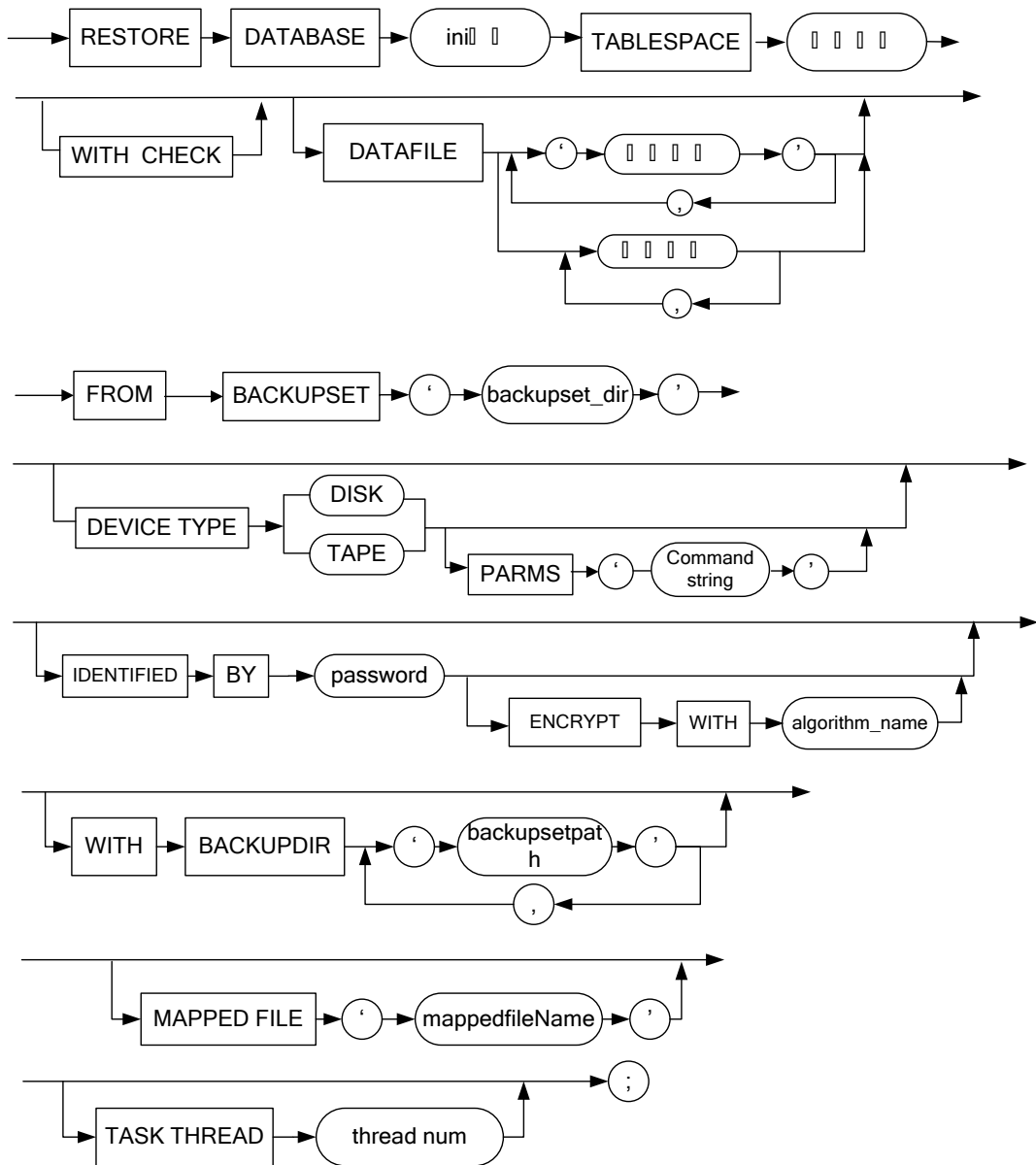
**WITH BACKUPDIR:** 指定备份搜索目录，最大长度为 256 个字节。使用完全备份还原中，若指定的备份集路径为相对路径可通过设置此参数搜索备份集；增量备份还原中设置该

参数除上述功能外还用于搜索基备份集。

**MAPPED FILE**：指定存放还原目标路径的文件，参考 3.3.5.2.1 [数据库还原](#)。

**TASK THREAD**：还原过程中数据处理过程线程的个数，取值范围 0~64，默认为 4，与备份过程中数据处理过程相对应。若指定为 0，则调整为 1；若指定超过当前系统主机核数，则调整为当前系统主机核数。

图例



使用说明：

1. 表空间还原不能是 TEMP 表空间，指定文件还原也不能为 TEMP 表空间中文件。
2. 表空间还原要求还原目标库与备份库为同一个库，且若还原目标库中 SYSTEM 表空间没有故障且还原目标表空间不是 SYSTEM 表空间的话，则要求目标库校验通过，比如



加密校验和日志校验等；否则将使用备份集中信息与当前的进行校验。

3. 还原目标库不能为已经执行过 `restore`，但是未指定过 `recover` 的库。

4. `SYSTEM` 表空间和 `ROLL` 表空间不允许指定 `UNTIL LSN` 或者 `UNTIL TIME` 还原；其他用户表空间可以。若未指定 `UNTIL TIME` 或者 `UNTIL LSN`，则均还原到最新状态。使用 `UNTIL TIME`（或者 `UNTIL LSN`）的同时不能指定 `DATAFILE` 还原，否则报语法错误。

5. 处于 `RES_OFFLINE` 或 `CORRUPT` 状态的表空间不允许指定表空间中数据文件还原。

6. 整个还原过程中不会修改数据库本身状态或者调整 `CKPT_LSN`。

7. 不管是 `DSC` 环境，还是单机环境，若异常退出，需手动指定各节点归档修复后，使用各节点完整的归档日志执行还原恢复；否则，将可能无法恢复到最新。

8. 异常退出库指定 `UNTIL TIME` 或者 `UNTIL LSN` 还原失效，因为库故障重启时会执行故障修复，`UNTIL` 失效，表空间仍会恢复到最新状态。

9. 若目标库中 `SYSTEM` 表空间故障，则必须优先还原 `SYSTEM` 表空间。

在 `DMDSC` 环境中进行表空间还原，需要先确保所有节点实例都已退出，此时在任一节点上使用该节点的备份集均可进行表空间还原操作，且只要在一个节点上执行目标表空间还原即可。

10. 如果 `SYSTEM` 表空间处于 `ONLINE/OFFLINE` 状态且文件丢失，则必须要通过库还原修复，不支持对其执行表空间还原。

以联机表空间备份集为例，展示 `DMRMAN` 如何完成表空间的还原：

1) 联机备份数据库，保证数据库运行在归档模式及 `OPEN` 状态。

```
SQL>BACKUP TABLESPACE MAIN BACKUPSET '/home/dm_bak/ts_full_bak_for_restore';
```

2) 校验备份，校验待还原备份集的合法性。校验备份有两种方式，联机和脱机，此处使用脱机校验。

```
RMAN>CHECK BACKUPSET '/home/dm_bak/ts_full_bak_for_restore';
```

3) 还原表空间。需要注意，表空间还原的目标库只能是备份集产生的源库，否则将报错。启动 `DMRMAN`，输入以下命令：

```
RMAN>RESTORE DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' TABLESPACE  
MAIN FROM BACKUPSET '/home/dm_bak/db_full_bak_for_restore';
```

### 3.3.6.1.2 表空间恢复

表空间恢复通过重做 `REDO` 日志，以将数据更新到一致状态。

恢复完成后，表空间状态置为 ONLINE，并设置数据标记为 FIL\_TS\_RECV\_STAT\_RECOVERED，表示数据已恢复到一致状态。

语法如下：

```
RECOVER DATABASE <INI 路径> TABLESPACE <表空间名> [WITH ARCHIVEDIR '归档目录'{'', '
归档目录'}][USE DB_MAGIC <db_magic>];
```

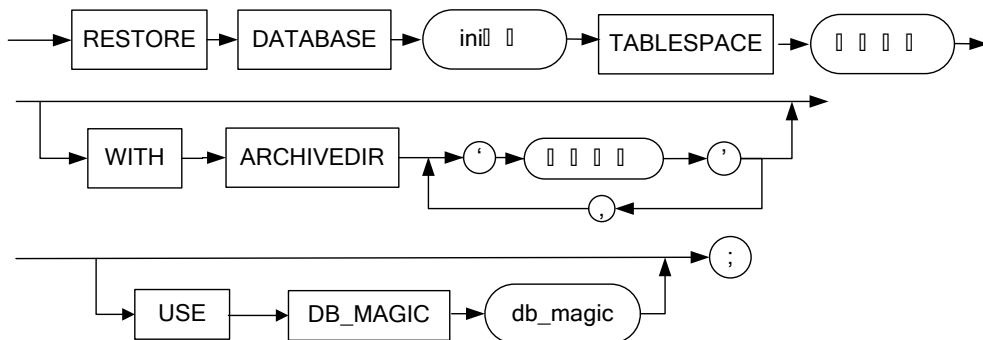
DATABASE：指定还原库目标的 dm.ini 文件路径。

TABLESPACE：指定还原的表空间，除了 temp 表空间。

WITH ARCHIVEDIR：归档文件搜索目录。缺省情况下在 dmarch.ini 中指定归档目录中搜索。如果归档日志不在配置文件 dmarch.ini 中指定的目录下，或者归档日志分散在多个目录下，需要使用该参数指定搜索归档日志。

USE DB\_MAGIC：指定本地归档日志对应数据库的 DB\_MAGIC，若不指定，则默认使用目标恢复数据库的 DB\_MAGIC。

图例



以联机表空间备份为例，展示 DMRMAN 如何完成表空间的恢复：

- 1) 联机备份数据库，保证数据库运行在归档模式及 OPEN 状态。

```
SQL>BACKUP TABLESPACE MAIN BACKUPSET '/home/dm_bak/ts_full_bak_for_recover';
```

- 2) 校验备份，校验待还原备份集的合法性，此处使用脱机校验。

```
RMAN>CHECK BACKUPSET '/home/dm_bak/ts_full_bak_for_recover';
```

- 3) 还原表空间。启动 DMRMAN，输入以下命令：

```
RMAN>RESTORE DATABASE '/opt/dmdbms/data/DAMENG_FOR_RECOVER/dm.ini' TABLESPACE
MAIN FROM BACKUPSET '/home/dm_bak/db_full_bak_for_recover';
```

- 4) 恢复表空间。启动 DMRMAN，输入以下命令：

```
RMAN>RECOVER DATABASE '/opt/dmdbms/data/DAMENG_FOR_RECOVER/dm.ini' TABLESPACE
```

```
MAIN;
```

### 3.3.6.2 执行表空间还原和恢复：高级场景

#### 3.3.6.2.1 表空间还原

##### 还原表空间中指定的数据文件

DM 不仅支持从数据库备份和表空间备份中还原表空间，还支持还原表空间中特定的数据文件。若表空间已经被破坏，则不允许执行特定数据文件的还原。使用数据库备份还原表空间或表空间中的数据文件与使用表空间备份操作类似，区别在于 RESTORE 语句中指定的备份集为数据库备份集。因此，本节仅以使用表空间备份为例说明还原数据文件的操作步骤。

1) 创建待备份的表空间 TS\_FOR\_RES\_01，并在库目录下创建 3 个数据文件。

```
SQL>CREATE TABLESPACE TS_FOR_RES_01 DATAFILE'ts_for_res_01_01.dbf' SIZE 128;
SQL>ALTER TABLESPACE TS_FOR_RES_01 ADD DATAFILE'ts_for_res_01_02.dbf' SIZE 128;
SQL> ALTER TABLESPACE TS_FOR_RES_01 ADD DATAFILE'ts_for_res_01_03.dbf' SIZE 128;
```

2) 备份表空间。

```
SQL>BACKUP TABLESPACE TS_FOR_RES_01 BACKUPSET '/home/dm_bak/ts_bak_for_dbf';
```

3) 查询数据文件的文件编号或路径。还原指定的数据文件需知道数据文件对应的文件编号或路径，相关信息可通过查询动态视图 V\$DATAFILE 获取。

```
SQL>SELECT ID, PATH FROM V$DATAFILE;
```

查询结果：

行号	ID	PATH
10		/home/xm/DAMENG/SYSTEM.DBF
20		/home/xm/DAMENG/ROLL.DBF
30		/home/xm/DAMENG/MAIN.DBF
40		/home/xm/DAMENG/TEMP.DBF
50		/home/xm/DAMENG/ts_for_res_01_01.dbf
61		/home/xm/DAMENG/ts_for_res_01_02.dbf
72		/home/xm/DAMENG/ts_for_res_01_03.dbf

如果想还原 ts\_for\_res\_01\_02.dbf 和 ts\_for\_res\_01\_03.dbf 两个数据文件，通过查询结果可知它们的文件编号（1 和 2）以及相应的路径。

4) 校验备份。此步骤为可选。

```
RMAN>CHECK BACKUPSET '/home/dm_bak/ts_bak_for_dbf';
```

5) 通过文件编号还原表空间 TS\_FOR\_RES\_01 中的数据文件 ts\_for\_res\_01\_02.dbf 和 ts\_for\_res\_01\_03.dbf。

```
RMAN>RESTORE DATABASE '/home/xm/DAMENG/dm.ini' TABLESPACE TS_FOR_RES_01 DATAFILE
1, 2 FROM BACKUPSET '/home/dm_bak/ts_bak_for_dbf';
```

6) 如果不想使用文件编号还原，使用指定数据文件路径还原数据文件的语句如下：

```
SQL>RESTORE                TABLESPACE                TS_FOR_RES_01                DATAFILE
'/home/xm/DAMENG/ts_for_res_01_02.dbf',
'/home/xm/DAMENG/ts_for_res_01_03.dbf'                FROM                BACKUPSET
'/home/dm_bak/ts_bak_for_dbf';
```

### 指定映射文件还原

映射文件用于指定存放还原目标路径，即备份集里面的数据文件路径。指定映射文件还原可以重新指定备份集中数据文件的路径。下面以 MAIN 表空间为例说明如何使用映射文件还原。

1) 备份 MAIN 表空间。

```
SQL>BACKUP TABLESPACE MAIN BACKUPSET '/home/dm_bak/ts_bak_for_map';
```

2) 创建映射文件

使用 DMRMAN 的 DUMP 命令生成映射文件 map\_file.txt，存至 /home/dm\_mappedfile 目录。文件中指定数据文件 MAIN.DBF 还原后的路径为 /home/dm\_dbf/MAIN.DBF。关于映射文件的详细内容参见下文的 [3.3.5.2.1 数据库还原](#)。

```
/**=====*/
/*[DAMENG_MAIN_FIL_0]*/
fil_id      = 0
ts_id       = 4
ts_name     = MAIN
data_path   = /home/dm_dbf/MAIN.DBF
/***** END *****/
```

### 3) 还原 MAIN 表空间。

```
RMAN>RESTORE DATABASE '/home/xm/DAMENG/dm.ini' TABLESPACE MAIN FROM BACKUPSET
'/home/dm_bak/ts_bak_for_map'MAPPED FILE'/home/dm_mappedfile/map_file.txt';
```

## 3.3.6.2.2 表空间恢复

### 指定归档目录还原

由于磁盘空间的影响，数据库归档可能出现分布在多个目录的情况。出现这种情况时就需要指定归档目录还原。还原时指定多个归档目录的操作步骤如下：

#### 1) 备份用户表空间 MAIN。

```
SQL>BACKUP TABLESPACE MAIN BACKUPSET '/home/dm_bak/ts_bak_for_arch';
```

#### 2) 校验备份。此步骤可选。

```
RMAN>CHECKBACKUPSET '/home/dm_bak/ts_bak_for_arch';
```

3) 还原用户表空间 MAIN。假设归档日志分布在目录为 /home/dm\_arch1 和 /home/dm\_arch2 两个目录下。

```
RMAN>RECOVER DATABASE '/home/xm/DAMENG/dm.ini' TABLESPACE MAIN FROM BACKUPSET
'/home/dm_bak/ts_bak_for_arch' WITH ARCHIVEDIR '/home/dm_arch1',
'/home/dm_arch2';
```

### 主备环境下指定 DB\_MAGIC 收集归档

主备环境下，如果当前节点的归档缺失，使用归档进行恢复时，会由于归档日志不连续而报错，无法将表空间恢复到最新状态。如果另一节点保存有该部分日志，则可以使用该节点日志进行恢复。默认情况下，恢复阶段收集归档时只收集与待恢复库 DB\_MAGIC、PERMANENT\_MAGIC 一致的归档文件，而在主备环境中 PERMANNET\_MAGIC 是一致的，此时可通过指定 DB\_MAGIC 跳过这一限制。

#### 1) 搭建主备环境并登录任意节点进行备份。

```
SQL>BACKUP TABLESPACE MAIN BACKUPSET '/home/dm_bak/ts_bak_for_arch';
```

#### 2) 登录主节点插入部分数据后退出，然后删除该节点下所有归档日志。

#### 3) 从备份集还原表空间 MAIN 到主节点

```
RMAN>RESTORE DATABASE '/home/xm/DAMENG/dm.ini' TABLESPACE MAIN FROM BACKUPSET
'/home/dm_bak/ts_bak_for_arch';
```

4) 指定 DB\_MAGIC, 使用备库归档进行恢复。由于主节点下所有归档已被删除, 无法恢复到最新状态, 此时需要借助备库存在的归档

```
RMAN>RECOVER DATABASE '/home/xm/DAMENG/dm.ini' TABLESPACE MAIN FROM BACKUPSET  
'/home/dm_bak/ts_bak_for_arch' WITH ARCHIVEDIR ' '/home/dm_arch2' USE DB_MAGIC  
18446520;
```

### 3.3.7 归档还原

使用 RESTORE 命令完成脱机还原归档操作, 在还原语句中指定归档备份集。备份集可以是脱机归档备份集, 也可以是联机归档备份集。

语法如下:

```
RESTORE <ARCHIVE LOG | ARCHIVELOG> [WITH CHECK] FROM BACKUPSET '<备份集目录>'  
[DEVICE TYPE DISK|TAPE[PARMS '<介质参数>']]  
[IDENTIFIED BY <密码> [ENCRYPT WITH <加密算法>]]  
[TASK THREAD <任务线程数>] [NOT PARALLEL]  
[ALL | [FROM LSN <lsn 值>] | [UNTIL LSN <lsn 值>] | [LSN BETWEEN < lsn 值> AND <  
lsn 值>] | [FROM TIME '时间串'] | [UNTIL TIME '时间串'] | [TIME BETWEEN '时间串' AND  
'时间串'] ]  
TO <还原目录>  
[OVERWRITE level];  
<还原目录>:: =  
ARCHIVEDIR '<归档目录>' |  
DATABASE '<INI 文件路径>'
```

**WITH CHECK:** 指定还原前校验备份集数据完整性。 缺省不校验。

**BACKUPSET:** 指定用于还原目标数据库的备份集目录。若指定为相对路径, 会在默认备份目录下搜索备份集。

**DEVICE TYPE:** 指存储备份集的介质类型, 支持 DISK 和 TAPE, 默认为 DISK, 详见 1.3 介质管理层。

**PARMS:** 介质参数, 供第三方存储介质 (TAPE 类型) 管理使用。

**IDENTIFIED BY:** 指定备份时使用的加密密码, 供还原过程解密使用。

**ENCRYPT WITH:** 指定备份时使用的加密算法，供还原过程解密使用，若未指定，则使用默认算法。

**TASK THREAD:** 指定还原过程中用于处理压缩和解密任务的线程个数。若未指定，则默认为 4；若指定为 0，调整为 1；若指定超过当前系统主机核数，则调整为当前核数。

**NOT PARALLEL:** 指定并行备份集使用非并行方式还原。对于非并行备份集，不论是否指定该关键字，均采用非并行还原。

**ALL:** 还原所有的归档。

**FROM LSN, FROM TIME:** 指定还原的开始 LSN 或者开始的时间点。真正的起始点为该 LSN 或该时间点所在的整个归档日志文件作为起始点。例如，指定 FROM 10001，而归档日志文件 X 的 LSN 为 9000-12000，那么就会将该归档日志文件 X 作为起始归档文件。

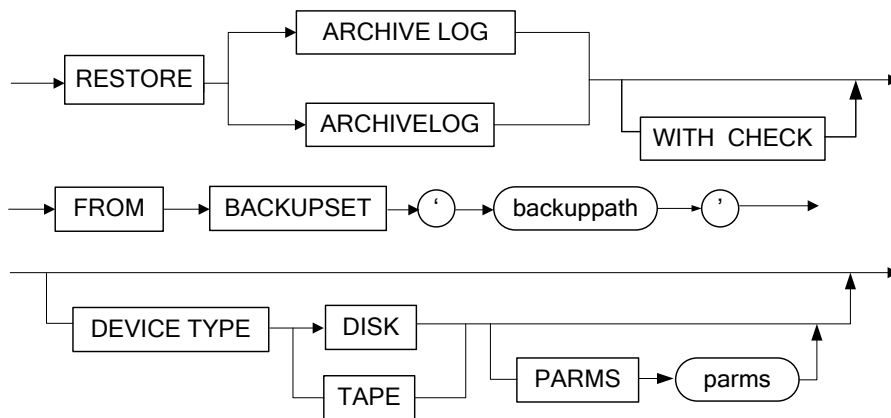
**UNTIL LSN, UNTIL TIME:** 指定还原的截止 LSN 或者截止的时间点。真正的截止点为该 LSN 或该时间点所在的整个归档日志文件作为结束归档文件。

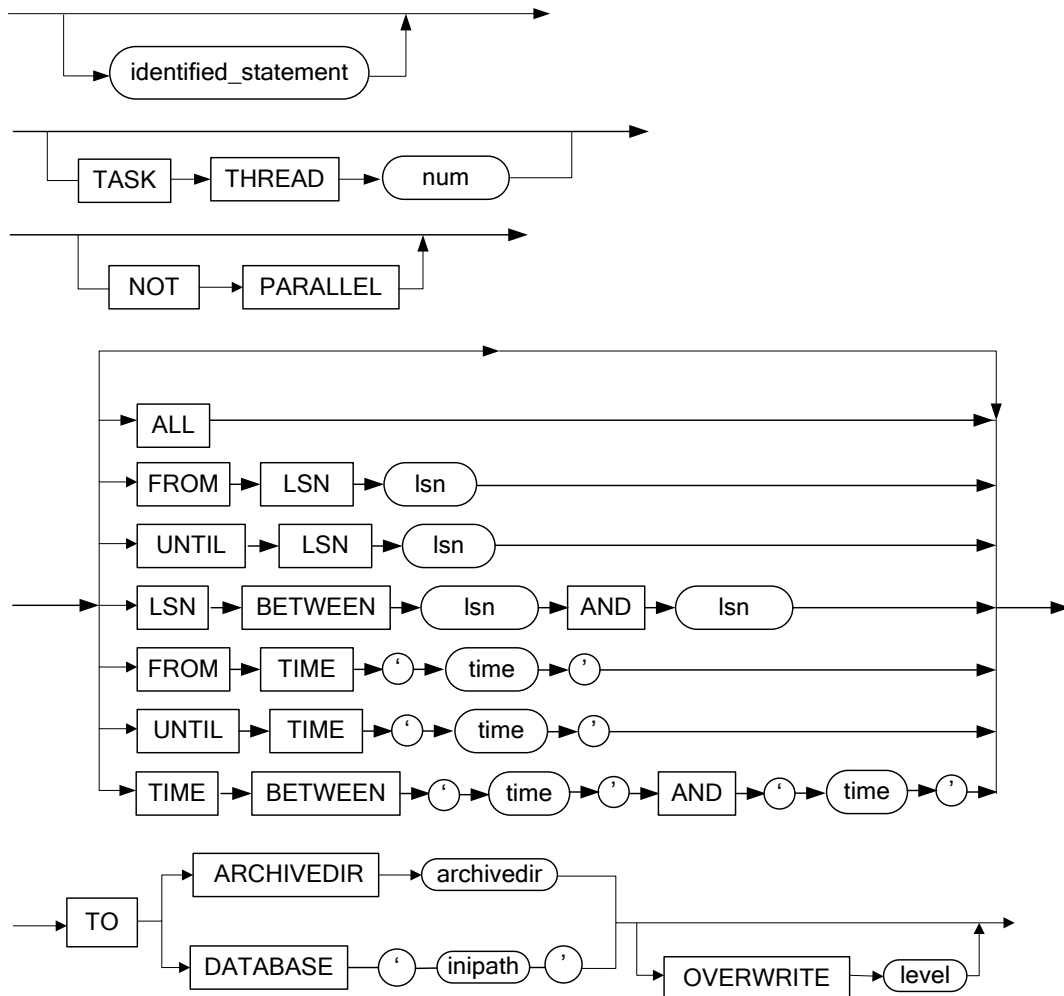
**BETWEEN ... AND ...:** 指定还原的区间。指落在还原区间内的所有归档文件。和该 LSN 范围有关的所有归档文件都会被选中。例如，如果指定还原区间为 BETWEEN 100 AND 200，而归档日志文件 1 的 LSN 范围为 1-150，归档文件 2 的 LSN 范围为 150-180，归档文，3 的 LSN 范围为 180-600，那么整个归档日志文件 1、2 和 3 都会被选中。

**<还原目录>:** 用户可以指定还原到指定的目录 ARCHIVEDIR，也可以指定还原指定的库的目录 DATABASE。

**OVERWRITE:** 还原归档时，如果遇到归档已经存在的处理，1: 跳过已存在的归档日志，继续其他日志的还原。跳过的信息会生成一条日志记录在安装目录 /log/dm\_BAKRES\_年月.log 里；2: 直接报错返回；3: 强制覆盖存在的归档日志。缺省为 1。

图例：





identified\_statement: 请参考3.2.5.1.1[概述](#)中的表空间还原图例

使用说明:

- 1) 联机备份归档，保证数据库运行在归档模式及 OPEN 状态。

```
SQL>BACKUP ARCHIVE LOG ALL BACKUPSET '/home/dm_bak/arch_all_for_restore';
```

- 2) 校验备份，校验待还原备份集的合法性。校验备份有两种方式，联机和脱机，此处使用脱机校验。

```
RMAN>CHECK BACKUPSET '/home/dm_bak/arch_all_for_restore';
```

校验成功:

```
RMAN> CHECK BACKUPSET '/home/dm_bak/arch_all_for_restore';
```

```
CHECK BACKUPSET '/home/dm_bak/arch_all_for_restore';
```

```
CMD END.CODE:[0]
```

```
check backup set successfully.
```

```
time used: 206.026(ms)
```



- 3) 还原归档。启动 DMRMAN, 设置 OVERWRITE 为 2, 如果归档文件已存在, 会报错。

```
RMAN>RESTORE ARCHIVE LOG FROM BACKUPSET '/home/dm_bak/arch_all_for_restore' TO
DATABASE '/opt/dmdbms/data/DAMENG_FOR_RESTORE/dm.ini' OVERWRITE 2;
```

或者

```
RMAN>RESTORE ARCHIVE LOG FROM BACKUPSET '/home/dm_bak/arch_all_for_restore' TO
ARCHIVEDIR '/opt/dmdbms/data/DAMENG_FOR_RESTORE/arch_dest' OVERWRITE 2;
```

### 3.3.8 归档修复

使用 REPAIR 命令完成指定数据库的归档修复, 归档修复会将目标库 dmarch.ini 中配置的所有本地归档日志目录执行修复。若目标库没有配置本地归档, 则不执行修复。执行修复时, 目标库一定不能处于运行状态。一般建议在数据库故障后, 应立即执行归档修复, 否则后续还原恢复导致联机日志中未刷入本地归档的 REDO 日志丢失, 届时再利用本地归档恢复将无法恢复到故障前的最新状态。

语法如下:

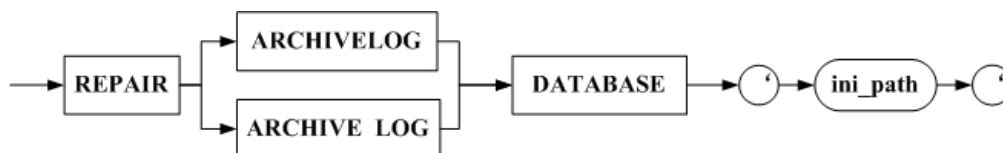
---

```
REPAIR <ARCHIVE LOG | ARCHIVELOG> DATABASE <'INI 文件路径'>;
```

---

INI 文件路径: 指定待修复归档的数据库对应的 dm.ini 路径。

图例:



使用说明:

- 1) 单机环境时, 确保目标库已经停止工作后, 执行归档修复。

```
RMAN>REPAIR ARCHIVELOG DATABASE '/opt/dmdbms/data/dm.ini';
```

- 2) DSC 环境时, 需要每个节点停止工作后, 且每个节点独立执行修复操作。对于两节点 DSC01、DSC02 执行修复如下。

```
RMAN> REPAIR ARCHIVELOG DATABASE '/opt/dmdbms/dsc/dm01.ini';
```

```
RMAN> REPAIR ARCHIVELOG DATABASE '/opt/dmdbms/dsc/dm02.ini';
```

- 3) DSC 环境下, REPAIR 操作也会利用其他节点的联机日志修复本地对应的 REMOTE 远程归档。如果脱机修复之后, 再进行备份还会报“指定或者默认目录中找不到完整归档日

志的错误”、“收集到的归档日志和起始 LSN 不连续”或“归档日志不完整”等，这种情况可能是因为节点之前的远程归档就已经不完整造成的，可以借助 dmrachk 工具手动拷贝远程节点的 LOCAL 归档到本节点对应的 REMOTE 归档目录中进行修复。

### 3.3.9 查看操作日志

本节主要介绍整个操作过程中可能涉及的日志或跟踪文件。默认情况下，这些文件都位于 DM 数据库的 log 目录中。文件名中的“xxx”为生成文件的年月，如 2018 年 5 月生成日志文件其文件名中的“xxx”部分就是“201805”，每个月生成一个日志文件，在该月中日志内容都是不断增加的。操作日志类型包括：

- 全局日志文件 dm\_DMSERVER\_xxx.log
- 备份还原日志文件 dm\_BAKRES\_\_xxx.log
- SBT 接口跟踪文件 dm\_SBTTRACE\_xxx.log

#### 3.3.9.1 全局日志文件

dm\_DMSERVER\_xxx.log 是全局日志文件，全局日志文件不可配置。dm\_DMSERVER\_xxx.log 是 DM 数据库的运行日志，数据库运行过程中的日志信息都包含在这个日志中，备份还原过程中备份还原操作的阶段性信息会记录在这个日志文件中，包括日志生成时间、日志级别、日志生成进程名称和进程 ID、日志生成线程和内容等。

日志格式：时间+日志类型（INFO/WARN/ERROR/FATAL）+进程(database)+进程 ID(Pxxxx)+线程(dm\_sql\_thd/main\_thread 等)+日志内容。

如执行备份时可能会生成如下日志信息：

```
2018-10-08 16:05:06 [INFO] database P0000012596 main_thread version info:
develop
2018-10-08 16:05:06 [INFO] database P0000012596 main_thread
nsrv_create_lsnr_sockets success, port_num=5236, sock_v4=868, sock_v6=864
2018-10-08 16:05:06 [INFO] database P0000012596 main_thread DM Database Server
x64 V8.0.0.90-Build(2018.08.20-96055)ENT startup...
2018-10-08 16:05:06 [INFO] database P0000012596 main_thread current instance new
pid is 12596, check whether it is running...
```

```
2018-10-08 16:05:13 [INFO] database P0000012596 main_thread fil_sys_init
2018-10-08 16:05:13 [INFO] database P0000012596 main_thread fil_ts_create
ts_id=0 ts_name=SYSTEM
2018-10-08 16:05:13 [INFO] database P0000012596 main_thread fil_ts_create
ts_id=1 ts_name=ROLL
2018-10-08 16:05:13 [INFO] database P0000012596 main_thread fil_ts_create
ts_id=2 ts_name=RLOG
2018-10-08 16:05:13 [INFO] database P0000012596 main_thread fil_ts_create
ts_id=4 ts_name=MAIN
2018-10-08 16:05:13 [INFO] database P0000012596 main_thread fil_ts_create
ts_id=5 ts_name=BOOKSHOP
2018-10-08 16:05:13 [WARNING] database P0000012596 main_thread License will
expire on 2019-08-20
.....
2018-10-08 16:05:15 [FATAL] database P0000012596 main_thread alter database open;
nsvr_check_version failed
2018-10-08 16:05:15 [FATAL] database P0000012596 main_thread code = -109,
dm_sys_halt now!!!
.....
2018-10-10 15:40:14 [INFO] database P0000004764 main_thread version info:
develop
2018-10-10 15:40:14 [INFO] database P0000004764 main_thread
nsvr_create_lsnr_sockets success, port_num=5236, sock_v4=836, sock_v6=832
2018-10-10 15:40:14 [INFO] database P0000004764 main_thread DM Database Server
x64 V8.0.0.98-Build(2018.09.28-97578)ENT startup...
2018-10-10 15:40:15 [INFO] database P0000004764 main_thread current instance new
pid is 4764, check whether it is running...
2018-10-10 15:40:21 [INFO] database P0000004764 main_thread fil_sys_init
2018-10-10 15:40:21 [INFO] database P0000004764 main_thread fil_ts_create
ts_id=0 ts_name=SYSTEM
```

```
2018-10-10 15:40:21 [INFO] database P0000004764 main_thread fil_ts_create
ts_id=1 ts_name=ROLL
.....
2018-10-10 15:40:24 [INFO] database P0000004764 main_thread
nsvr_process_before_open success.
2018-10-10 15:40:24 [INFO] database P0000004764 main_thread backup control file
d:\dmdbms\data\DAMENG\dm.ctl to file
d:\dmdbms\data\DAMENG\dm_20181010154024_230000.ctl
2018-10-10 15:40:24 [INFO] database P0000004764 main_thread backup control file
d:\dmdbms\data\DAMENG\dm.ctl to file
d:\dmdbms\data\DAMENG\ctl_bak\dm_20181010154024_249000.ctl succeed
2018-10-10 15:40:24 [INFO] database P0000004764 main_thread SYSTEM IS READY.
.....
2018-10-10 15:54:53 [INFO] database P0000004764 dm_sql_thd BACKUP DATABASE
BACKUPSET 'd:\dmdbms\data\bak\db_bak_for_encrypt' IDENTIFIED BY *****;
2018-10-10 15:54:53 [INFO] database P0000004764 dm_sql_thd BACKUP DATABASE
[DAMENG]
2018-10-10 15:54:53 [INFO] database P0000004764 dm_sql_thd CMD START.....
2018-10-10 15:54:53 [INFO] database P0000004764 dm_tskwrk_thd
rlog4_buf_apply_info_fill, fill apply lsn value 0!
2018-10-10 15:54:53 [INFO] database P0000004764 dm_redolog_thd
rlog4_write_to_file: file[d:\dmdbms\data\DAMENG\DAMENG01.log], handle[0x3a70],
offset[8946176], bytes_write[512]
2018-10-10 15:54:53 [INFO] database P0000004764 dm_redolog_thd
rlog4_write_to_file: lsn[46192, 46192], time[199439652456, 199440652059,
999603]
2018-10-10 15:54:54 [INFO] database P0000004764 dm_tskwrk_thd
rlog4_buf_apply_info_fill, fill apply lsn value 0!
2018-10-10 15:54:54 [INFO] database P0000004764 dm_redolog_thd
rlog4_write_to_file: file[d:\dmdbms\data\DAMENG\DAMENG01.log], handle[0x3a70],
```

```
offset[8946688], bytes_write[512]

2018-10-10 15:54:54 [INFO] database P0000004764 dm_redolog_thd
rlog4_write_to_file: lsn[46192, 46192], time[199440652648, 199441654404,
1001756]

2018-10-10 15:54:55 [INFO] database P0000004764 dm_tskwrk_thd
rlog4_buf_apply_info_fill, fill apply lsn value 0!

2018-10-10 15:54:55 [INFO] database P0000004764 dm_redolog_thd
rlog4_write_to_file: file[d:\dmdbms\data\DAMENG\DAMENG01.log], handle[0x3a70],
offset[8947200], bytes_write[512]

2018-10-10 15:54:55 [INFO] database P0000004764 dm_redolog_thd
rlog4_write_to_file: lsn[46192, 46192], time[199441654978, 199442725760,
1070782]

2018-10-10 15:54:56 [INFO] database P0000004764 dm_sql_thd
rlog4_buf_apply_info_fill, fill apply lsn value 0!

2018-10-10 15:54:56 [INFO] database P0000004764 dm_redolog_thd
rlog4_write_to_file: file[d:\dmdbms\data\DAMENG\DAMENG01.log], handle[0x3a70],
offset[8947712], bytes_write[512]

2018-10-10 15:54:56 [INFO] database P0000004764 dm_redolog_thd
rlog4_write_to_file: lsn[46192, 46192], time[199442725920, 199442995610,
269690]

2018-10-10 15:54:57 [ERROR] database P0000004764 dm_sql_thd  CMD
END.CODE:[-8216],DESC:[指定或者默认归档目录中找不到完整归档日志,脱机请执行归档修复
(repair archive log database 'xxxx/dm.ini')或者联机请刷检查点(select
checkpoint(xxx))之后再试(DSC 环境下各节点都要做)]

.....

2018-10-10 16:06:22 [INFO] database P0000004764 dm_sql_thd  BACKUP DATABASE
BACKUPSET 'd:\dmdbms\data\trace' TRACE LEVEL 2;

2018-10-10 16:06:22 [INFO] database P0000004764 dm_sql_thd  BACKUP DATABASE
[DAMENG]

2018-10-10 16:06:22 [INFO] database P0000004764 dm_sql_thd  CMD START.....
```

```
2018-10-10 16:06:25 [INFO] database P0000004764 dm_sql_thd  CMD END.CODE:[0]

2018-10-10 16:10:23 [INFO] database P0000004764 dm_chkpt_thd  ckpt info adjust:

ckpt_lsn(46192), ckpt_fil(0), ckpt_off(9140736), cur_lsn(46192),

next_seq(17845), cur_free(9140736)

.....
```

### 3.3.9.2 备份还原日志文件

dm\_BAKRS\_xxx.log 是备份还原日志文件，备份还原日志文件不可配置。dm\_BAKRS\_xxx.log 是 DM 备份还原在执行备份与还原时生成的日志，包括主进程（联机时为 dmserver 工作线程，脱机时为 dmrman）发送，以及 AP 进程处理过程信息。每条信息，包括生成时间、信息级别、生成日志的进程名称和进程 ID，对于 AP 进程还会生成对应主进程的 ID。另外，为了保证日志内容的完整性，还会记录由全局日志生成接口生成的日志记录，这部分日志格式参照全局日志文件说明。

备份还原日志格式为：

时 间 + 日 志 类 型 （ INFO/WARN/ERROR/FATAL/CMD/CMD\_PARSE ） + 进 程  
（ dmrman/database/dmbakres ） + 进程 ID（P 开头） + 主进程 ID（PP 开头，非 dmbakres  
显示 4294967295，非法值，否则为相应 dmrman 或者 database 的进程号） + 日志内容。

如下所示为一段 dm\_BAKRES\_xxx.log 日志信息示例：

```
.....

2018-10-10 16:06:04 [CMD] database P0000004764 PPID4294967295  BACKUP DATABASE
BACKUPSET 'd:\dmdbms\data\bak' TRACE LEVEL 2;

2018-10-10 16:06:04 [CMD] database P0000004764 PPID4294967295  BACKUP DATABASE
[DAMENG]

2018-10-10 16:06:04 [INFO] database P0000004764  PPID4294967295  CMD START....

2018-10-10 16:06:04 [INFO] database P0000004764  PPID4294967295  BACKUP DATABASE
[DAMENG], execute.....

2018-10-10 16:06:04 [ERROR] database P0000004764  PPID4294967295  CMD
END.CODE:[-8055],DESC:[备份目录冲突]

.....
```

## 备份与还原

```
2018-10-10 16:06:22 [CMD] database P0000004764 PPID4294967295 BACKUP DATABASE
BACKUPSET 'd:\dmdbms\data\trace' TRACE LEVEL 2;

2018-10-10 16:06:22 [CMD] database P0000004764 PPID4294967295 BACKUP DATABASE
[DAMENG]

2018-10-10 16:06:22 [INFO] database P0000004764 PPID4294967295 CMD START....

2018-10-10 16:06:22 [INFO] database P0000004764 PPID4294967295 BACKUP DATABASE
[DAMENG], execute.....

2018-10-10 16:06:22 [INFO] database P0000004764 PPID4294967295 connect to dmap
with channel[\\.\pipe\DM_PIPE_1384342949-4764_1], tsk_num: [4], code: [0].

2018-10-10 16:06:22 [INFO] dmap P0000012356 PPID4294967295 bakres start cmd
process with named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1].....

2018-10-10 16:06:22 [INFO] database P0000004764 PPID4294967295 send CMD
[BMML_INIT] by channel [\\.\pipe\DM_PIPE_1384342949-4764_1].

2018-10-10 16:06:22 [INFO] dmap P0000012356 PPID4294967295 bakres receive from
named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1] success, CMD [BMML_INIT].

2018-10-10 16:06:22 [INFO] dmap P0000012356 PPID0000004764 CMD [BMML_INIT] is
ready

2018-10-10 16:06:22 [INFO] database P0000004764 PPID4294967295 CMD CHECK
LSN .....

2018-10-10 16:06:22 [INFO] database P0000004764 PPID4294967295 BACKUP DATABASE
[DAMENG], collect dbf.....

2018-10-10 16:06:22 [INFO] database P0000004764 PPID4294967295 CMD CHECK .....

2018-10-10 16:06:23 [INFO] database P0000004764 PPID4294967295 send CMD
[BACKUP_START] by channel [\\.\pipe\DM_PIPE_1384342949-4764_1].

2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID4294967295 bakres receive from
named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1] success, CMD [BACKUP_START].

2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID0000004764 CMD [BACKUP_START]
is ready

2018-10-10 16:06:23 [INFO] database P0000004764 PPID4294967295 CMD EXEC:
BEGIN_LSN->46193, CUR_LSN->46192, MAXPIECESIZE->131072, BAK_MAGIC->335245487,
```

```
PARALLEL_NUM->1, INI_PATH->d:\dmdbms\data\dameng\dm.ini,
BAKSET_DIR->d:\dmdbms\data\trace, BKPNAME->trace,
BAKNAME->DB_FULL_DAMENG_20181010_160622_000485, BASE_DIR->none,
BASE_NAME->none, BASE_BEGIN_LSN->-1, BASE_END_LSN->-1
2018-10-10 16:06:23 [INFO] database P0000004764 PPID4294967295 DBF BACKUP
SUBS.....
2018-10-10 16:06:23 [INFO] database P0000004764 PPID4294967295 send CMD
[BACKUP_INIT] by channel [\\.\pipe\DM_PIPE_1384342949-4764_1], parallel_num:
[1], base_dir: [], base_name: [], base_begin_lsn: [-1], base_end_lsn: [-1],
bakset_dir: [d:\dmdbms\data\trace].
2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID4294967295 bakres receive from
named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1] success, CMD [BACKUP_INIT].
2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID0000004764 backup db DAMENG to
file ...
2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID0000004764 CMD [BACKUP_INIT]
is ready
2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID4294967295 bakres send EXECUTE
OVER status by named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1], CODE [0]
2018-10-10 16:06:23 [INFO] database P0000004764 PPID4294967295 send CMD
[BACKUP_DBF] by channel [\\.\pipe\DM_PIPE_1384342949-4764_1] for DBF
[d:\dmdbms\data\DAMENG\SYSTEM.DBF].
2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID4294967295 bakres receive from
named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1] success, CMD [BACKUP_DBF].
2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID0000004764 CMD [BACKUP_DBF] for
DBF [d:\dmdbms\data\DAMENG\SYSTEM.DBF] end, code:[0].....
2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID0000004764 CMD [BACKUP_DBF] for
DBF [d:\dmdbms\data\DAMENG\SYSTEM.DBF] end, code:[0].....
2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID4294967295 bakres send EXECUTE
OVER status by named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1], CODE [0]
2018-10-10 16:06:23 [INFO] database P0000004764 PPID4294967295 total 1 packages
```



```
processed...

2018-10-10 16:06:23 [INFO] database P0000004764 PPID4294967295 send CMD
[BACKUP_DBF] by channel [\\.\pipe\DM_PIPE_1384342949-4764_1] for DBF
[d:\dmdbms\data\DAMENG\ROLL.DBF].

2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID4294967295 bakres receive from
named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1] success, CMD [BACKUP_DBF].

2018-10-10 16:06:23 [INFO] dmap P0000012356 PPID0000004764 CMD [BACKUP_DBF] for
DBF [d:\dmdbms\data\DAMENG\ROLL.DBF] end, code:[0].....

2018-10-10 16:06:24 [INFO] dmap P0000012356 PPID0000004764 CMD [BACKUP_DBF] for
DBF [d:\dmdbms\data\DAMENG\ROLL.DBF] end, code:[0].....

2018-10-10 16:06:24 [INFO] dmap P0000012356 PPID4294967295 bakres send EXECUTE
OVER status by named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1], CODE [0]

2018-10-10 16:06:24 [INFO] database P0000004764 PPID4294967295 total 3 packages
processed...

2018-10-10 16:06:24 [INFO] database P0000004764 PPID4294967295 send CMD
[BACKUP_DBF] by channel [\\.\pipe\DM_PIPE_1384342949-4764_1] for DBF
[d:\dmdbms\data\DAMENG\MAIN.DBF].

2018-10-10 16:06:24 [INFO] dmap P0000012356 PPID4294967295 bakres receive from
named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1] success, CMD [BACKUP_DBF].

2018-10-10 16:06:24 [INFO] dmap P0000012356 PPID0000004764 CMD [BACKUP_DBF] for
DBF [d:\dmdbms\data\DAMENG\MAIN.DBF] end, code:[0].....

2018-10-10 16:06:24 [INFO] dmap P0000012356 PPID0000004764 CMD [BACKUP_DBF] for
DBF [d:\dmdbms\data\DAMENG\MAIN.DBF] end, code:[0].....

2018-10-10 16:06:24 [INFO] dmap P0000012356 PPID4294967295 bakres send EXECUTE
OVER status by named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1], CODE [0]

2018-10-10 16:06:24 [INFO] database P0000004764 PPID4294967295 total 4 packages
processed...

2018-10-10 16:06:24 [INFO] database P0000004764 PPID4294967295 DBF BACKUP
MAIN.....

2018-10-10 16:06:24 [INFO] database P0000004764 PPID4294967295 BACKUPSET
```

```
[d:\dmdbms\data\trace] END, CODE [0].....

2018-10-10 16:06:24 [INFO] database P0000004764 PPID4294967295

END_LSN->46192.

2018-10-10 16:06:24 [INFO] database P0000004764 PPID4294967295 META

GENERATING.....

2018-10-10 16:06:24 [INFO] database P0000004764 PPID4294967295 send CMD

[BACKUP_META] by channel [\\.\pipe\DM_PIPE_1384342949-4764_1].

2018-10-10 16:06:24 [INFO] dmap P0000012356 PPID4294967295 bakres receive from

named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1] success, CMD [BACKUP_META].

2018-10-10 16:06:24 [INFO] dmap P0000012356 PPID0000004764 end the process

thread.

2018-10-10 16:06:24 [INFO] dmap P0000012356 PPID0000004764 end the process

thread.

2018-10-10 16:06:24 [INFO] dmap P0000012356 PPID0000004764 end the process

thread.

2018-10-10 16:06:24 [INFO] dmap P0000012356 PPID0000004764 end the flush

thread.

2018-10-10 16:06:25 [INFO] dmap P0000012356 PPID0000004764 CMD [BACKUP_META]

is ready.

2018-10-10 16:06:25 [INFO] database P0000004764 PPID4294967295 total 8 packages

processed...

2018-10-10 16:06:25 [INFO] database P0000004764 PPID4294967295 BPSP notify

subprocess complete, pipe send successfully

2018-10-10 16:06:25 [INFO] dmap P0000012356 PPID4294967295 bakres receive from

named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1] success, CMD [COMPLETE].

2018-10-10 16:06:25 [INFO] database P0000004764 PPID4294967295 total 8 packages

processed...

2018-10-10 16:06:25 [INFO] database P0000004764 PPID4294967295 BPSP notify

subprocess end, pipe send successfully

2018-10-10 16:06:25 [INFO] dmap P0000012356 PPID4294967295 bakres receive from
```

```
named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1] success, CMD [END].

2018-10-10 16:06:25 [INFO] dmap P0000012356 PPID4294967295 code = 0, bakres CMD

[END] begin...

2018-10-10 16:06:25 [INFO] dmap P0000012356 PPID4294967295 bakres CMD [END] tsk

node get...

2018-10-10 16:06:25 [INFO] dmap P0000012356 PPID4294967295 bakres CMD [END]

end.

2018-10-10 16:06:25 [INFO] dmap P0000012356 PPID4294967295 bakres end cmd

process with named pipe [\\.\pipe\DM_PIPE_1384342949-4764_1].

2018-10-10 16:06:25 [INFO] database P0000004764 PPID4294967295 total 8 packages

processed!

2018-10-10 16:06:25 [INFO] database P0000004764 PPID4294967295 CMD

END.CODE:[0]

.....
```

### 3.3.9.3 SBT 接口跟踪文件

dm\_SBTTRACE\_xxx.log 是 SBT 接口跟踪文件，SBT 接口跟踪文件可配置，相关 TRACE 文件详见 [3.2.3 数据备份高级主题](#)。SBT 接口在打开 TRACE 跟踪(TRACE LEVEL > 1) 的情况下生成执行日志，默认文件名为 dm\_SBTTRACE\_xxx.log，用户可根据需要自行设置文件名。

如下所示为一段 dm\_SBTTRACE\_xxx.log 日志信息示例：

```
.....

2018-10-10 16:06:23 sbtrestore(filepath=d:\dmdbms\data\bak\bak.meta)

2018-10-10 16:06:23 sbtread(buf != NULL, buf_len = 4096)

2018-10-10 16:06:23 sbtread(buf != NULL, buf_len = 49152)

2018-10-10 16:06:23 sbtread(buf != NULL, buf_len = 25088)

2018-10-10 16:06:23 sbtclose()

2018-10-10 16:06:23 sbtend(del_flag=FALSE)

2018-10-10 16:06:23 sbtbackup(type=1, name=trace.bak)
```

```

2018-10-10 16:06:23 sbtwrite(buf != NULL, buf_len = 4096)

2018-10-10 16:06:23 sbtwrite(buf != NULL, buf_len = 5169664)

2018-10-10 16:06:24 sbtwrite(buf != NULL, buf_len = 33554944)

2018-10-10 16:06:24 sbtwrite(buf != NULL, buf_len = 2753024)

2018-10-10 16:06:24 sbtwrite(buf != NULL, buf_len = 33280)

2018-10-10 16:06:24 sbtclose()

2018-10-10 16:06:24 sbtinfo(!NULL)

file [trace.bak] finished, name:[trace.bak],created time:[2018-10-10
16:06:23],expired time:[2105-12-31 23:59:59],label:[],comment:[]

2018-10-10 16:06:24 sbtbackup(type=3, name=trace.meta)

2018-10-10 16:06:24 sbtwrite(buf != NULL, buf_len = 53248)

2018-10-10 16:06:24 sbtwrite(buf != NULL, buf_len = 4096)

2018-10-10 16:06:25 sbtwrite(buf != NULL, buf_len = 4096)

2018-10-10 16:06:25 sbtwrite(buf != NULL, buf_len = 4096)

2018-10-10 16:06:25 sbtwrite(buf != NULL, buf_len = 4096)

2018-10-10 16:06:25 sbtwrite(buf != NULL, buf_len = 512)

2018-10-10 16:06:25 sbtclose()

.....

```

### 3.4使用图形化客户端工具进行备份还原

DM 数据库提供几种类型的图形化客户端管理工具。在 WINDOWS 的 DM 数据库系统，可以通过选择“开始”-“程序”-“达梦数据库”-“客户端”，然后选择具体的某个客户端工具。比如，通过 MANAGER 管理工具可以与数据库进行交互，操作数据库对象和从数据库获取信息；通过控制台工具 CONSOLE 可以完成数据库的配置与监控等。下文分别介绍使用 MANAGER 管理工具联机执行备份还原和使用 CONSOLE 控制台工具脱机执行备份还原。

#### 3.4.1 使用 MANAGER 工具进行联机备份还原

下面将介绍如何使用 DM 的 MANAGER 管理工具来执行联机的备份与还原的操作。MANAGER 管理工具主界面如图 3.1 所示：

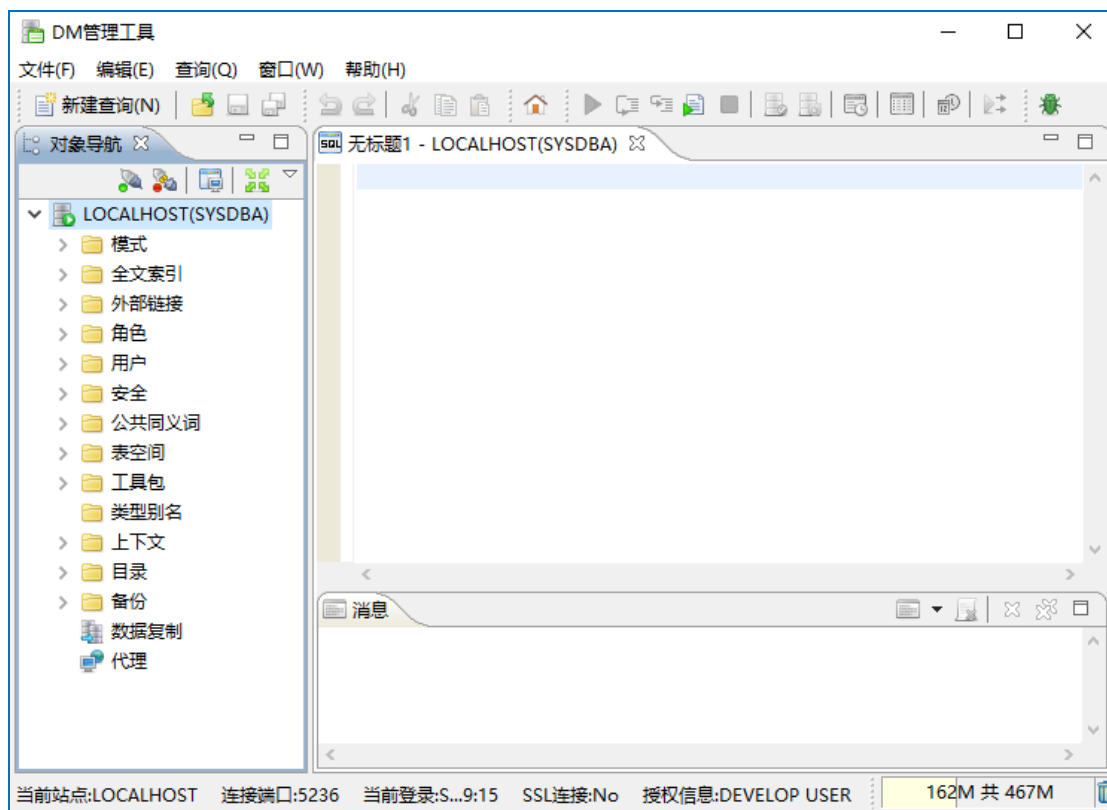


图 3.1 MANAGER 工具主页面

MANAGER 工具的对象导航树中的备份节点提供了联机备份还原相关的各项操作。导航树备份节点下面包含三个子节点：库备份、表备份、表空间备份和归档备份。各备份节点的右键菜单中提供了所有可操作的选项，包括新建备份、指定工作目录、备份恢复、备份校验、属性等。备份文件夹节点右键菜单如下：

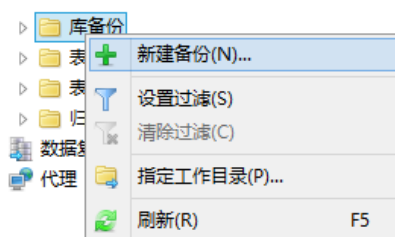


图 3.2 备份节点操作选项

备份节点右键菜单如下：

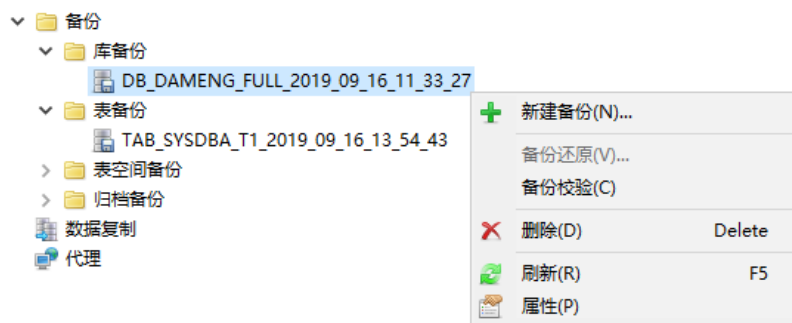


图 3.3 备份集节点操作选项

## 3.4.1.1 数据备份

数据备份包括库备份、表备份、表空间备份和归档备份，分别对应备份文件夹节点中库备份、表备份、表空间备份和归档备份的子节点。点开各子节点，显示当前已经存在的相应备份集名称。四者的备份流程基本一致，只是具体可配置的参数有些不同，下面以库备份为例介绍。

库备份节点右键菜单->新建备份，可以打开新建库备份对话框，首页显示的是常规选项页面，如图 3.4 所示：

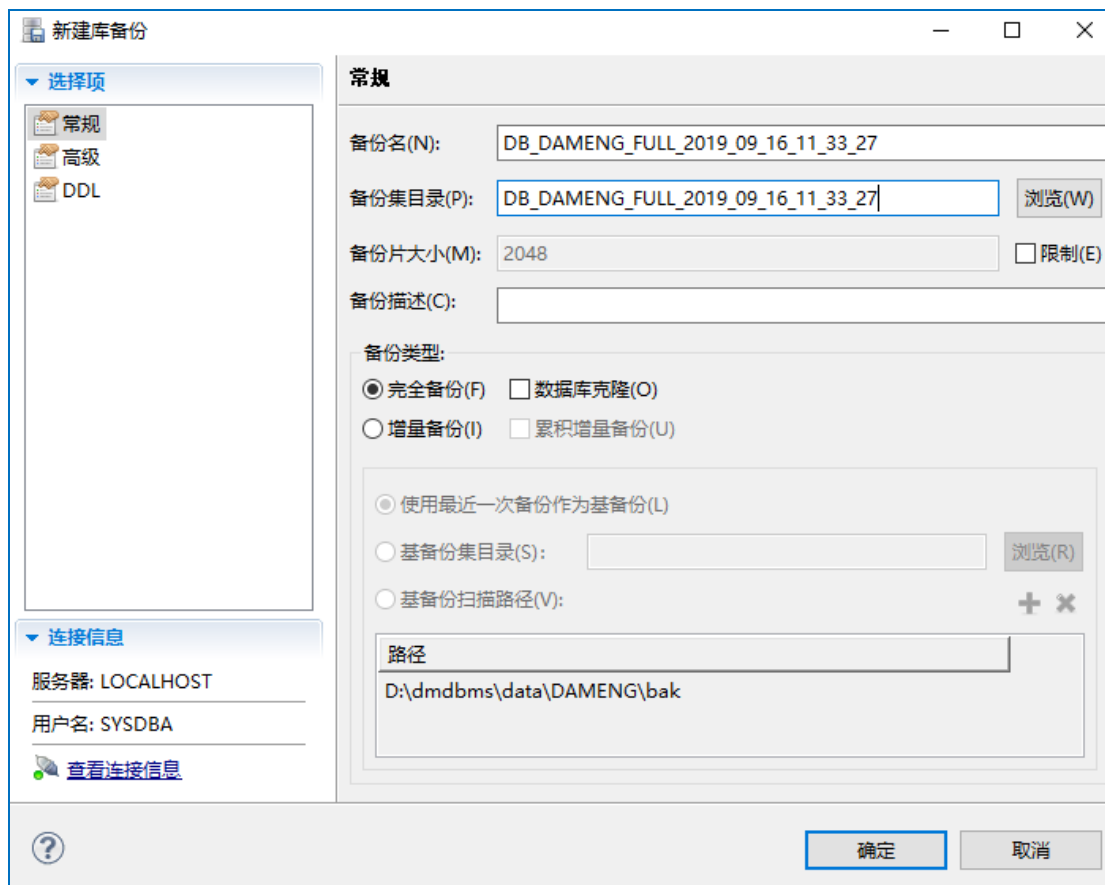


图 3.4 数据库备份常规选项页面

常规页面可配置库备份的常规参数信息，具体如下：

备份名：备份的名称。

备份集目录：备份集的目录，默认为系统默认备份集目录，即 DM 数据库安装目录下数据文件目录加备份集名称。

备份片最大大小：备份片的最大大小，如果备份文件超过该大小，则系统自动将备份文件分为多个文件。以兆为单位。不限制时，系统默认备份文件的最大大小为 2048M。限制时，最小 128M，32 位系统最大 2G，64 位系统最大 128G。

备份描述：对备份文件的描述，用户可以根据需要填写。

备份类型：

- ✓ 完全备份：对数据库的完全备份，不依赖于其他备份，可以独立地进行数据库恢复。  
数据库克隆选项，表示进行数据库备份时，仅拷贝所有的元数据不拷贝数据。如对于数据库中的表来说，只备份表的定义不备份表中数据。
- ✓ 增量备份：在完全备份或增量备份的基础上进行的备份。进行增量备份时，允许用户自己指定基备份来进行备份，默认使用最近一次备份作为基备份，在该数据

库不存在任何备份时，备份类型不能为增量备份。累积增量备份选项的备份类型只能是基备份，缺省为差异增量备份。

库备份和表空间备份都可指定备份类型为增量备份，如果选择增量备份，可以为增量备份配置以下参数：

**基备份目录：**可以指定基备份集目录，默认系统会自己在指定的基备份扫描路径下查找最近一次备份作为基备份。

**基备份扫描路径：**指定基备份的扫描路径，可以同时指定多个，增量备份时必须保证依赖的所有基备份都可以在指定的基备份扫描路径中找到。

高级选项页面中可以配置一些其他的高级选项，如下图所示：

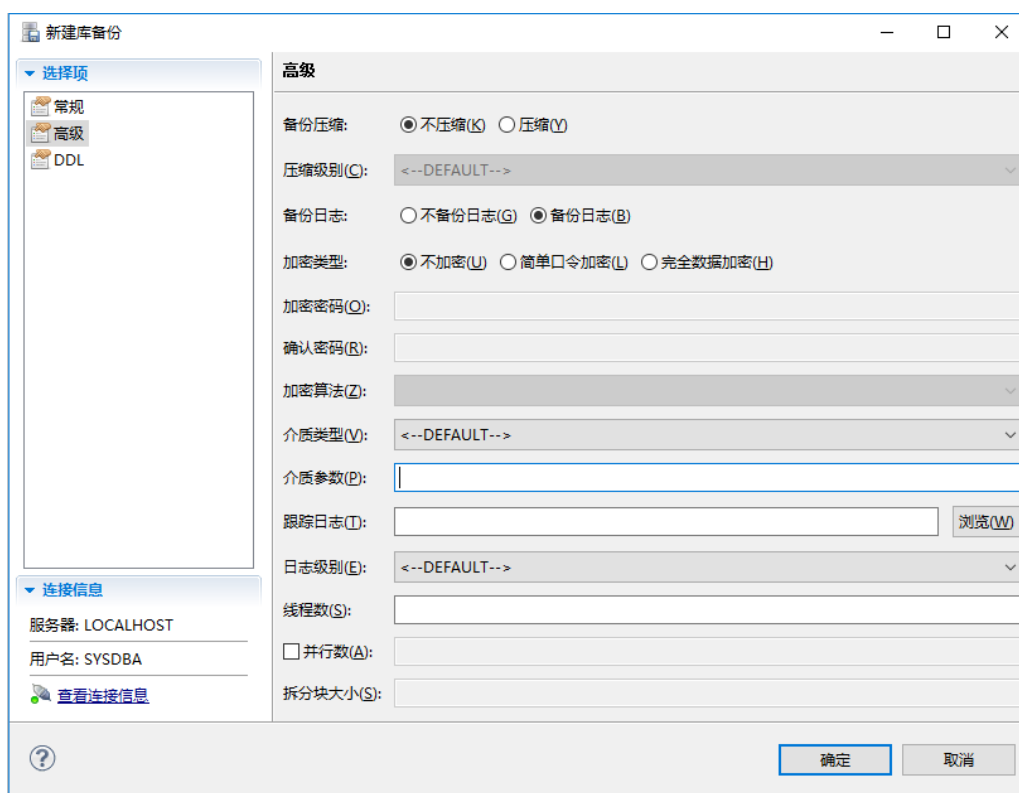


图 3.5 数据库备份高级选项页面

可为备份配置以下高级参数：

**备份压缩：**可以选择压缩或不压缩，通过压缩可以减少备份文件所需要的磁盘空间。默认为不压缩。

**压缩级别：**压缩的级别。可以设置压缩级别 0~9，1 表示 1 级压缩，9 表示 9 级压缩。压缩级别越高，压缩越慢，但压缩比越高。

**备份日志：**是否需要备份归档日志中的内容。默认备份日志。

**加密类型：**加密的类型，默认为不加密。



- ✓ 不加密：不对备份文件进行加密处理。
- ✓ 简单口令加密：对备份文件设置口令，但文件内容仍以明文存储。
- ✓ 完全数据加密：对备份文件进行完全的加密，备份文件以密文方式存储。

加密密码：当选择加密时，需要输入加密密码。

确认密码：当选择加密时，需要确认输入的加密密码。

加密算法：加密时使用的算法。在选择简单口令加密和完全数据加密的情况下适用。支持的加密算法参考数据库备份章节的参数说明。

介质类型：指定存储备份集的设备类型，默认选择磁盘，目前暂支持磁盘和磁带。

介质参数：使用第三方介质类型（TAPE 类型）时使用的参数。

跟踪日志：备份过程的跟踪日志路径。系统有默认路径，当日志级别为记录跟踪日志时启用。

日志级别：是否记录跟踪日志。默认不记录。

线程数：备份过程中数据处理过程线程的个数，取值范围 0~64，默认为 4。若指定为 0，则调整为 1；若指定超过当前系统主机核数，则调整为主机核数。线程数（TASK THREAD）\*并行数（PARALLEL）不得超过 512。

并行数：指定是否执行并行备份，以及执行并行备份的并行数。并行数没有上限，但实际可使用的最大并行数就是目标备份数据文件的个数。若勾选但没指定并行数，则默认并行数为 4。若未勾选，则认为非并行备份。

拆分块大小用户指定对于大数据量数据文件并行备份时拆分块的大小，默认为 1GB，最小不能低于 512MB，当指定的<拆分块大小>小于 512MB 时，系统会自动调整为 512MB。若指定并行备份，但未指定<拆分块大小>，则直接使用默认拆分块大小进行拆分。当数据文件的大小小于<拆分块大小>时，不执行拆分；当数据文件的大小大于<拆分块大小>时，执行拆分。并行数不能大于拆分之后的总块数。

DDL 页面可以查看新建库备份的 SQL 语句，右下角的保存按钮也可以保存 SQL 脚本到文件中，如下图所示：

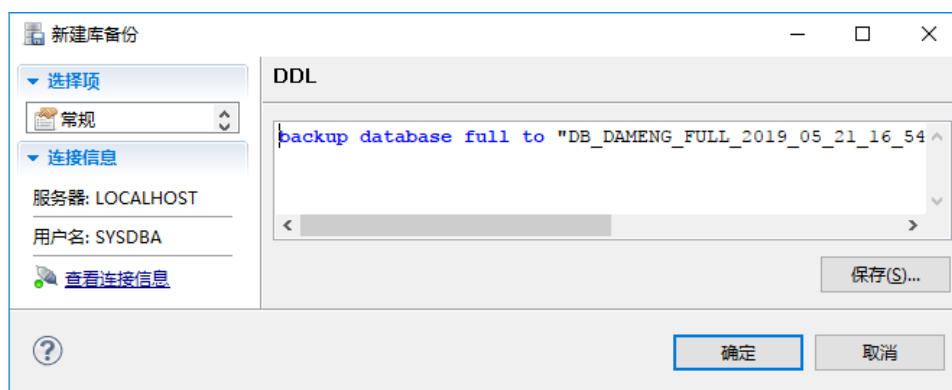


图 3.6 数据库备份查看 DDL 页面

### 3.4.1.2 备份管理

备份管理包括备份集信息、备份校验、备份删除和指定工作目录，其中备份删除是备份管理的主要功能。下面将分别对这些功能的使用进行介绍。

#### 备份集查看

点击某个备份节点右键菜单->属性，可以打开备份属性查看对话框，属性列表中可以查看该备份集的信息，包括备份集的元数据信息集数据库信息，分别如图 3.7、图 3.8、图 3.9 所示：

## 备份与还原

备份属性 - "DB\_DAMENG\_FULL\_2019\_09\_16\_11\_33\_27"

选择项

- 常规
- 文件信息
- 数据库信息
- DDL

连接信息

服务器: LOCALHOST

用户名: SYSDBA

[查看连接信息](#)

常规

属性名	属性值
服务器	LOCALHOST
连接用户名	SYSDBA
备份集ID	1846676583
备份名称	DB_DAMENG_FULL_2019_09_16_11_33_27
备份集目录	D:\dmdbms\data\DAMENG\bak\DB_DAMENG
备份时间	2019-09-16 11:33:31.000030
备份模式	联机
备份级别	库级
备份类型	完全备份
基准备份	
备份对象	DAMENG
是否累积增量备份	No
是否数据库克隆	No
是否备份日志	Yes
加密类型	不加密
压缩级别	0
介质类型	DISK
备份开始LSN	80830
备份结束LSN	80833

确定 取消

图 3.7 备份集常规信息显示页面

备份属性 - "DB\_DAMENG\_FULL\_2019\_09\_16\_11\_33\_27"

选择项

- 常规
- 文件信息
- 数据库信息
- DDL

连接信息

服务器: LOCALHOST

用户名: SYSDBA

[查看连接信息](#)

文件信息

备份集目录(G): D:\dmdbms\data\DAMENG\bak\DB\_DAMENG\_FULL\_2019\_09\_16\_11\_33\_27

备份片数(P): 2

备份片编号	备份片大小	备份片文件名
0	41932800	DB_DAMENG_FULL_2019_09_16_11_33_27.bak
1	5632	DB_DAMENG_FULL_201...16_11_33_27_1.bak

备份片(O):

数据文件数(E): 3

序号	文件ID	文件路径	文件大小(字节)	表空间ID	表空间名
1	0	D:\dmdbms\data\DAMEI	24117248	0	SYSTEM
2	0	D:\dmdbms\data\DAMEI	134217728	1	ROLL
3	0	D:\dmdbms\data\DAMEI	134217728	4	MAIN

数据文件(H):

确定 取消

图 3.8 备份集文件信息界面

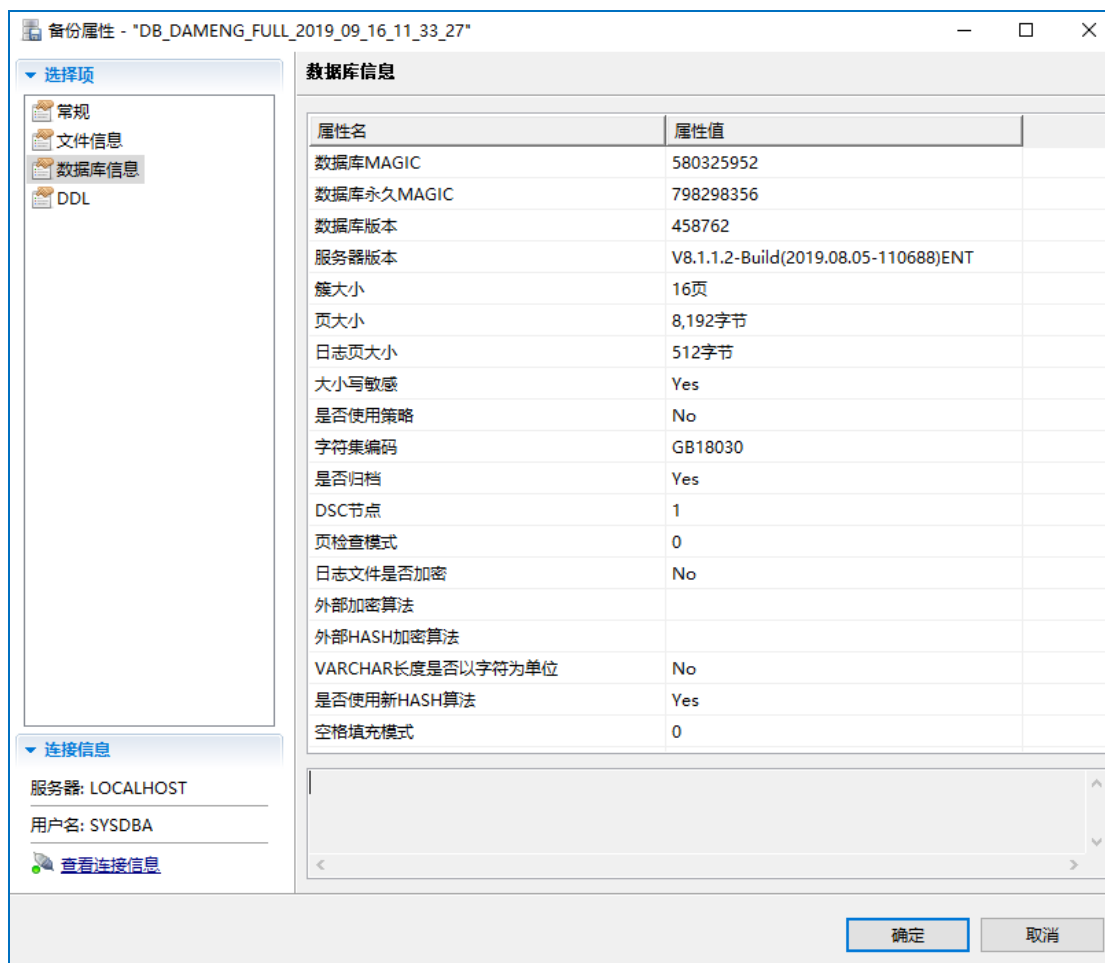


图 3.9 备份集数据库信息显示页面

### 备份校验

点击某备份节点右键菜单->备份校验，即可校验备份集的合法性。

### 备份删除

点击某备份节点右键菜单->删除，可以删除该备份集。也可以同时选中多个备份节点进行批量删除。

### 指定工作目录

点击备份文件夹节点右键菜单->指定工作目录，可以指定备份的工作目录，允许同时指定多个。如果设置了备份工作目录，备份文件夹下会显示所有工作目录下的所有备份。指定工作目录的对话框如下：

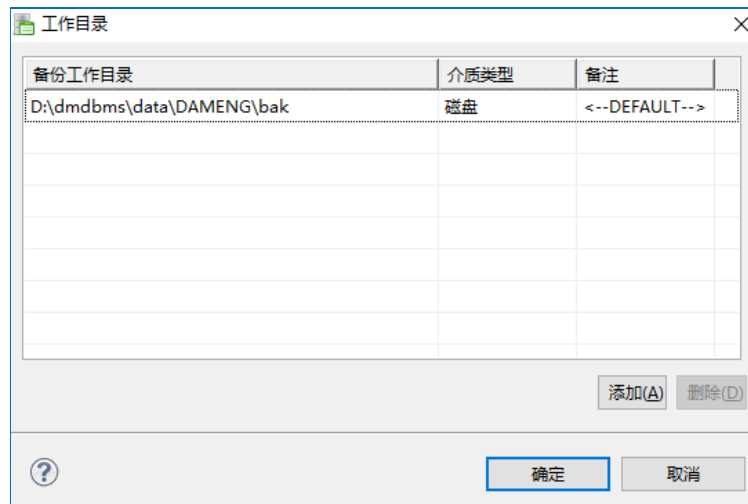


图 3.10 指定工作目录页面

点击[添加]按钮可以添加一个工作目录，选中一个工作目录点击[删除]按钮可以删除一个工作目录。系统有一个默认工作目录，该目录会一直保留不被删除。

### 3.4.1.3 数据还原

库备份和表空间不支持联机还原，只有表备份支持联机还原。表还原过程中表空间中其他的表还可以正常操作。

表备份节点右键菜单->备份还原，可以打开表空间备份还原对话框的常规页面如下图所示：

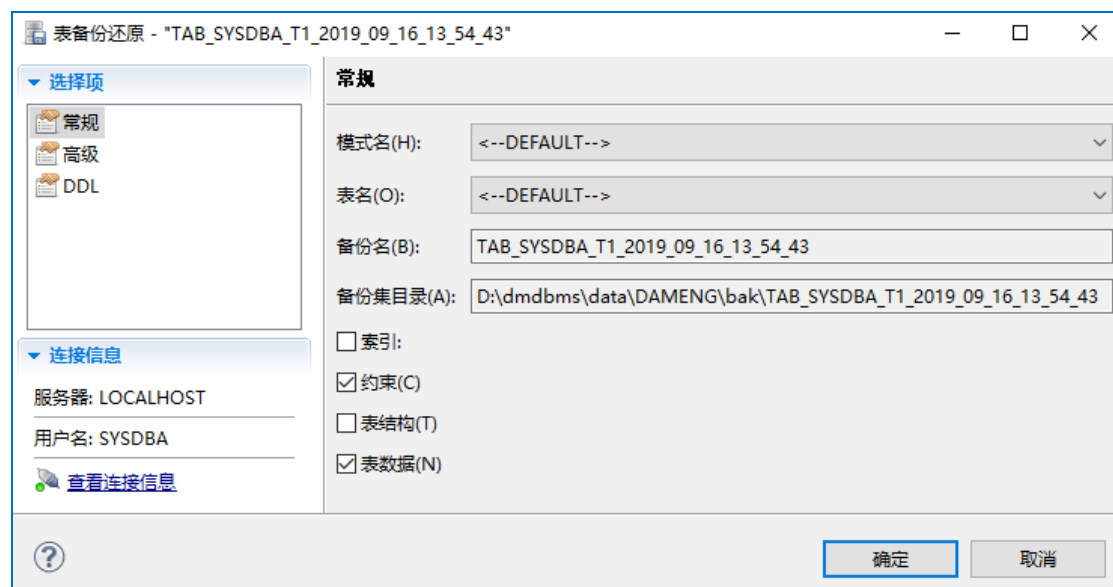


图 3.51 表还原常规选项页面

可为表还原配置以下常规参数：

模式名：模式的名称。

表名：表的名称。

备份名：备份的名称，无需填写，系统自动获取。

备份集目录：备份集所在的目录，无需填写，系统自动获取。

WITH INDEX/WITHOUT INDEX：指定还原数据后是否重建二级索引，默认重建。

WITH CONSTRAINT/WITHOUT CONSTRAINT：指定还原数据后是否重建约束，默认重建。

索引：指定还原数据后是否重建二级索引。

约束：指定还原数据后是否重建约束。

表结构：指定只执行表结构还原，不还原数据。若未指定，则必须勾选表数据。

表数据：指定是否只执行表中数据还原；表数据还原要求还原目标表结构与备份集中完全一致，否则报错，所以在表数据还原时，建议勾选表结构还原执行，减少报错。若未指定表数据，则必须勾选表结构。表结构和表数据二者，至少要选一个。

高级选项页面如下：

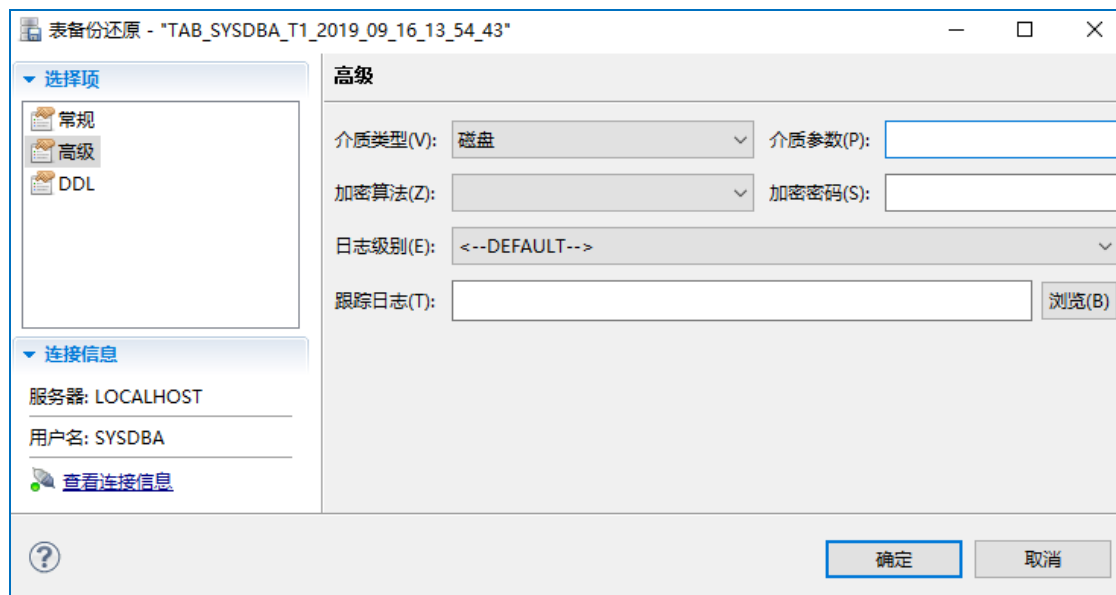


图 3.62 表还原常规选项页面

可配置以下参数：

介质类型：指存储备份集的介质类型，暂支持 DISK 和 TAPE，默认 DISK。

介质参数：使用第三方介质类型（TAPE 类型）时使用的参数。只对介质类型为 TAPE 时有效。

加密算法/加密密码：如果备份时使用了加密，则还原时必须输入加密算法和加密密码。

日志级别：是否记录跟踪日志。有效值 1、2，默认为 1 表示不启用 trace，为 2 表示启用 trace。

跟踪日志：指定生成的 trace 文件名。

## 3.4.2 使用 CONSOLE 工具进行脱机备份还原

下面将介绍如何使用 DM 控制台工具 CONSOLE 来执行库级脱机备份与还原的操作。控制台主界面如下图所示。

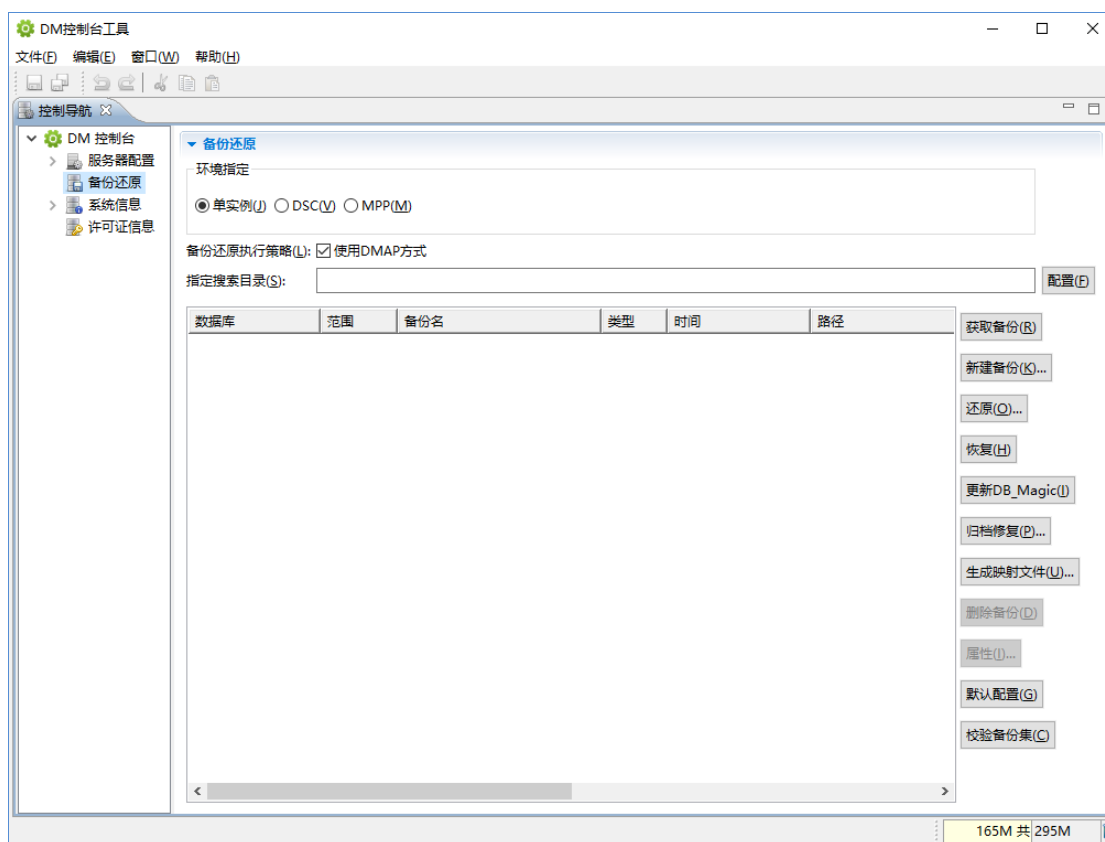


图 3.73 CONSOLE 工具备份还原主页面

### 3.4.2.1 数据备份

CONSOLE 工具的控制导航树中提供备份还原节点，单击备份还原节点可以打开备份还原管理界面。在备份还原管理界面中，点击新建备份按钮。打开新建备份对话框，如下图所示。

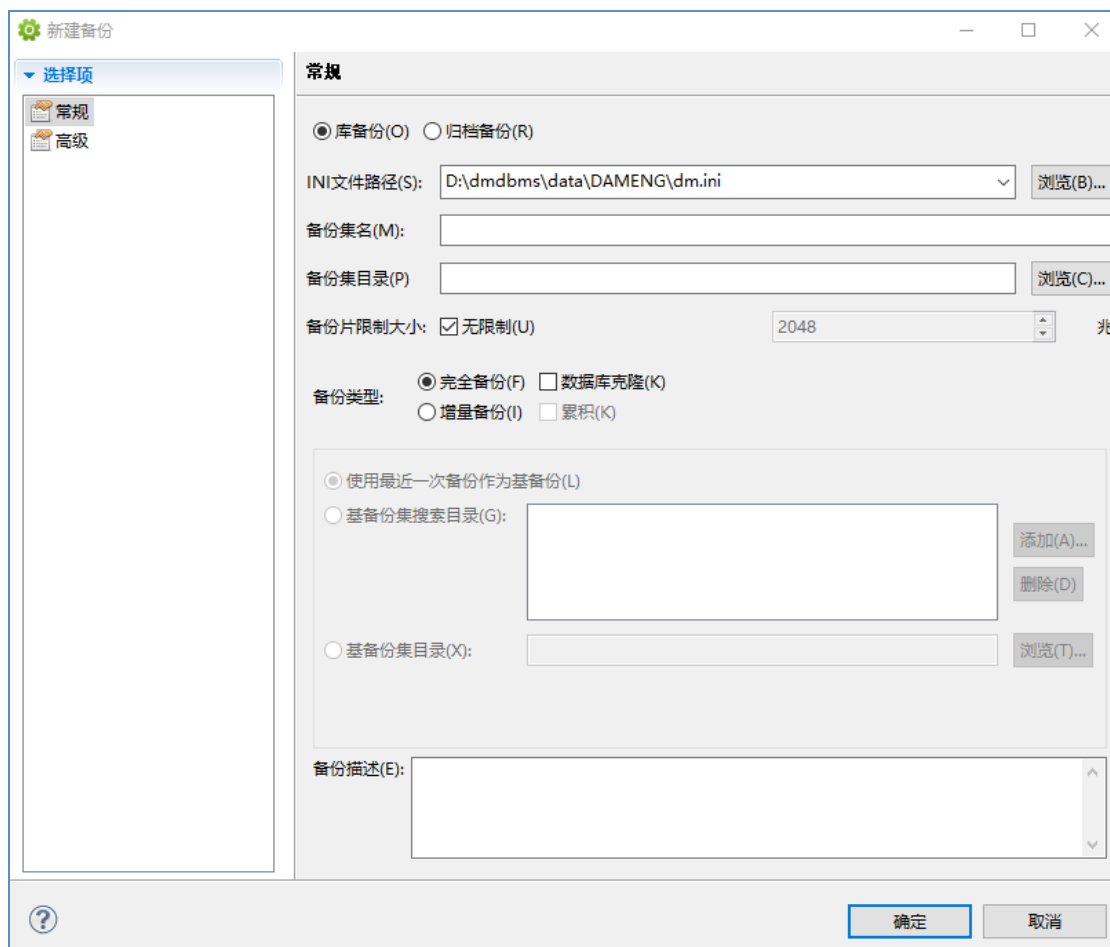


图 3.84 数据库备份常规选项页面-库备份

需要为脱机库备份配置以下参数：

INI 文件路径：必填，待备份目标数据库 dm.ini 文件路径。

备份集名：选填，指定生成备份集名称，若未指定，则使用约定方法随机生成。

备份集目录：选填，指定当前备份集生成目录，若未指定，则在默认备份目录中生成备份集目录。

备份片限制大小：默认无限制，取消选中无限制，则可以设置备份片的限制大小，以兆为单位，最小 128M，32 位系统最大 2G，64 位系统最大 128G。

备份类型：完全备份和增量备份。数据库克隆是完全备份的一个选项，表示进行数据库备份时，仅拷贝所有的元数据不拷贝数据。默认选完全备份，在已经存在完全备份的情况下，可以选择增量备份。

基备份集搜索目录：选填，并且仅适用于增量备份。

基备份集目录：选填，并且仅适用于增量备份。

备份描述：选填，根据实际情况对备份集进行描述。



根据实际需要配置完上述参数后，点击**确定**按钮就开始执行脱机备份操作了。备份操作可能需要花费一些时间，备份结束后，如果备份成功则弹出备份成功的对话框；如果备份失败，则弹出对话框提示失败原因。

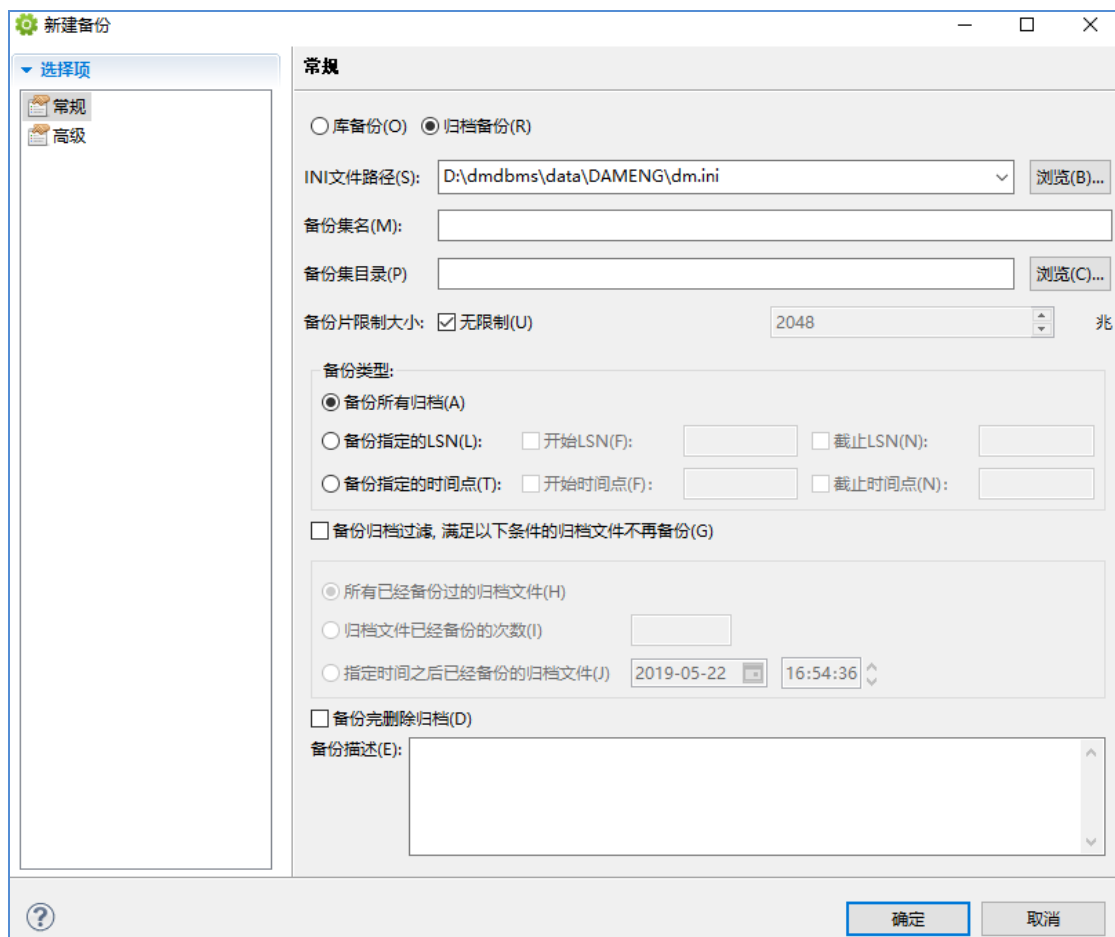


图 3.95 数据库备份常规选项页面-归档备份

需要为脱机归档备份配置以下参数:

备份类型:

- 备份所有归档。缺省情况下，为此项。
- 备份指定的 LSN
- 备份指定的时间点

备份归档过滤，满足的条件：

- 所有已经备份过得归档文件。缺省情况下，为此项。
- 归档文件已经备份的次数
- 指定之间之后已经备份的归档文件

备份完删除归档：可选项。归档备份完毕，是否删除归档文件。

备份描述：选填，根据实际情况对备份集进行描述。

高级选项页面如下：

The screenshot shows the 'New Backup' (新建备份) window with the 'Advanced' (高级) tab selected. The configuration options are as follows:

- 备份压缩 (Backup Compression):** Radio buttons for '不压缩(K)' (No Compression) and '压缩等级(Y)' (Compression Level). A dropdown menu shows level '1'.
- 备份日志 (Backup Log):** Radio buttons for '不备份日志(G)' (Do not backup log) and '备份日志(U)' (Backup log).
- 加密类型 (Encryption Type):** Radio buttons for '不加密(N)' (No encryption), '简单口令加密(L)' (Simple password encryption), and '完全数据加密(H)' (Full data encryption).
- 加密密码(O):** Text input field.
- 确认密码(C):** Text input field.
- 加密算法(R):** Dropdown menu.
- 介质类型(J):** Dropdown menu showing '磁盘' (Disk).
- 介质参数(W):** Text input field.
- 任务线程数(X):** Text input field.
- 并行备份(P):** Radio buttons for '不并行(A)' (Not parallel) and '并行(B)' (Parallel). A dropdown menu shows '4'.
- 拆分块大小(S):** Text input field.

At the bottom right are '确定' (OK) and '取消' (Cancel) buttons.

图 3.106 数据库备份高级选项页面

可为脱机备份配置以下参数：

**备份压缩：**默认不压缩。选择压缩等级，则可以设置压缩级别 0~9，1 表示 1 级压缩，9 表示 9 级压缩。压缩级别越高，压缩越慢，但压缩比越高。

**加密类型：**默认不加密。

**加密密码：**在选择简单口令加密和完全数据加密的情况下，需输入加密密码。

**加密算法：**在选择简单口令加密和完全数据加密的情况下适用。支持使用第三方加密。  
**CONSOLE 工具第三方加密算法的使用方法是：**1. 把自定义的加密算法生成 DLL 动态库；2. 把动态库放入 external\_crypto\_libs 文件夹；3. 再把文件夹放入 CONSOLE 工具所在的 DM 安装目录 tool 下。

**介质类型：**指定存储备份集的设备类型，默认选择磁盘，目前暂支持磁盘和磁带。

**介质参数：**使用第三方介质类型（TAPE 类型）时使用的参数。

**任务线程数：**备份过程中数据处理过程线程的个数，取值范围 0~64，默认为 4。若指定为 0，则调整为 1；若指定超过当前系统主机核数，则调整为主机核数。

**并行备份：**指定是否执行并行备份，以及执行并行备份的并行数。并行数没有上限，但

实际可使用的最大并行数就是目标备份数据文件的个数。若勾选但没指定并行数，则默认并行数为 4。若未勾选，则认为非并行备份。

## 3.4.2.2 备份管理

### 查看备份集

在备份还原管理页面中，指定搜索目录，点击**获取备份**按钮，即可获取备份集列表。操作步骤如图 3.16、图 3.17、图 3.18 所示。

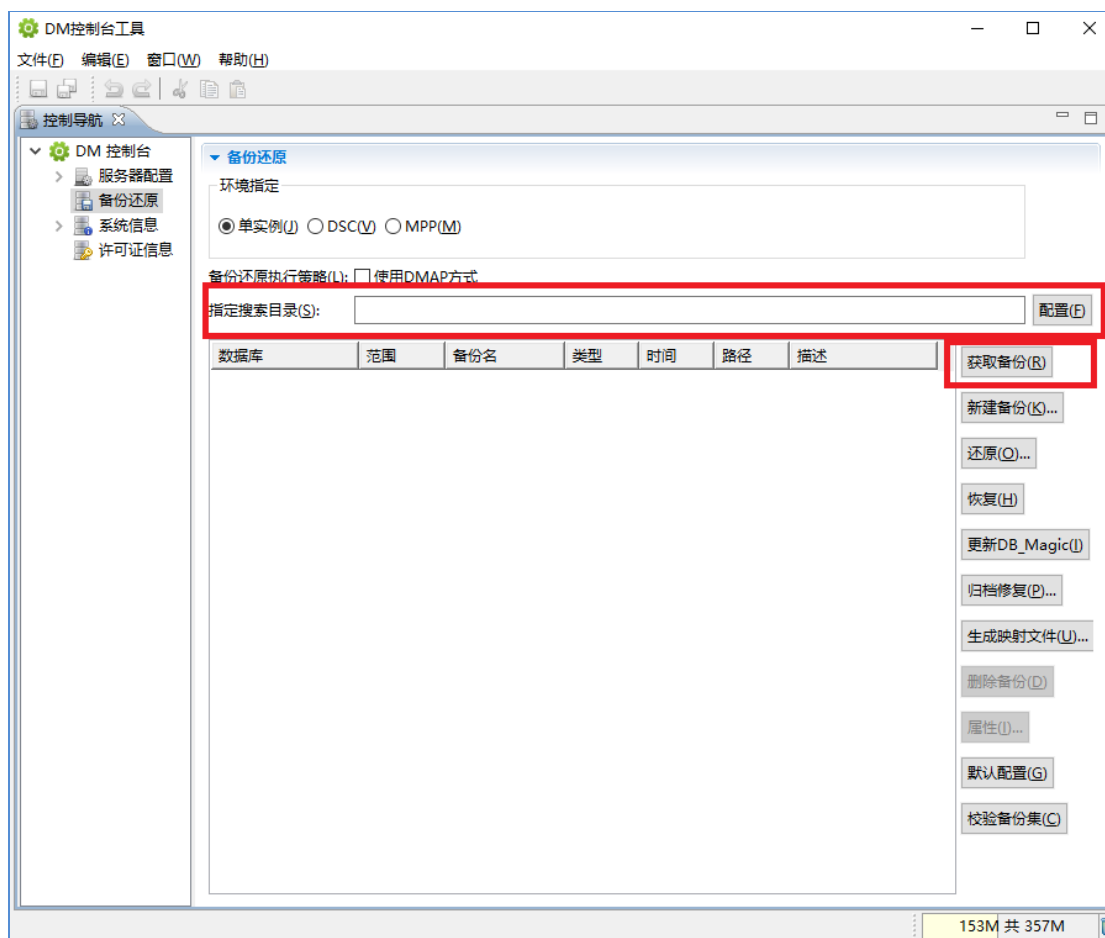


图 3.117 指定搜索目录并获取备份

可以获取到指定搜索目录下的所有备份集，显示在备份列表中。选择要查看的备份集，点击**属性**按钮。

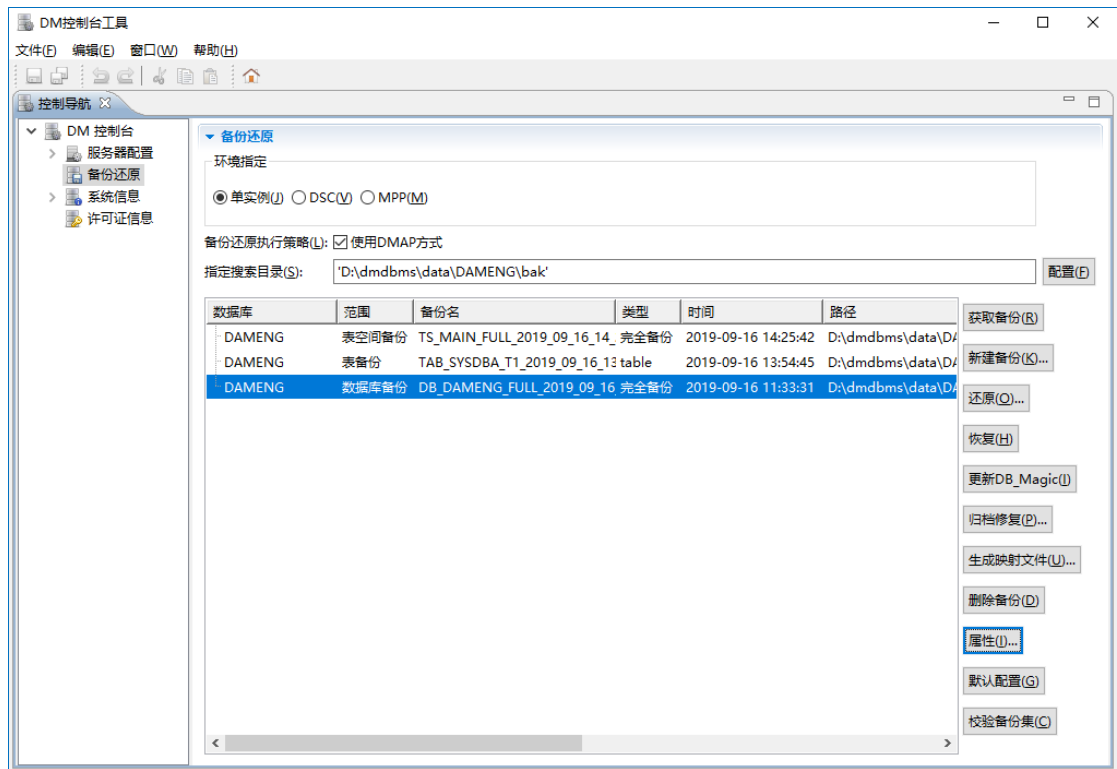


图 3.128 备份集列表显示页面

打开备份属性对话框，可查看备份属性。备份属性分为常规、元信息、DSC 信息、文件信息、数据库信息几类。其中 DSC 信息只有 DSC 环境的备份集才有。常规属性主要显示备份集相关的属性，如图 3.18 所示。

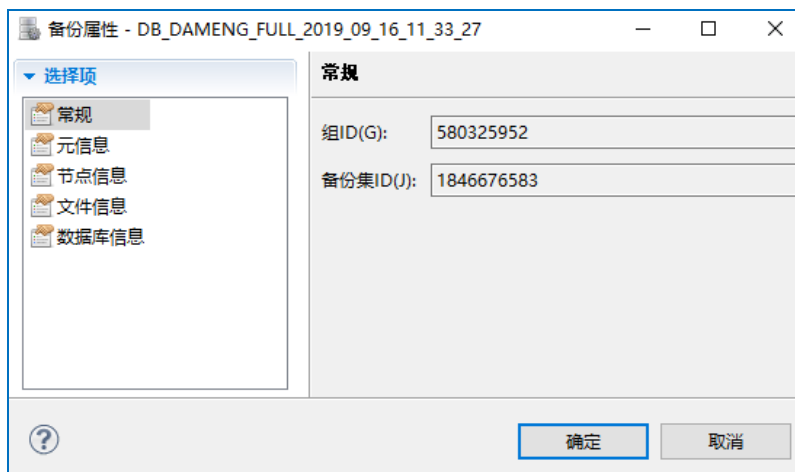


图 3.139 备份集-常规信息

元信息主要显示备份集本身相关的信息如备份是否联机备份、备份的范围、备份的加密信息，以及备份的压缩信息等，如下图所示。

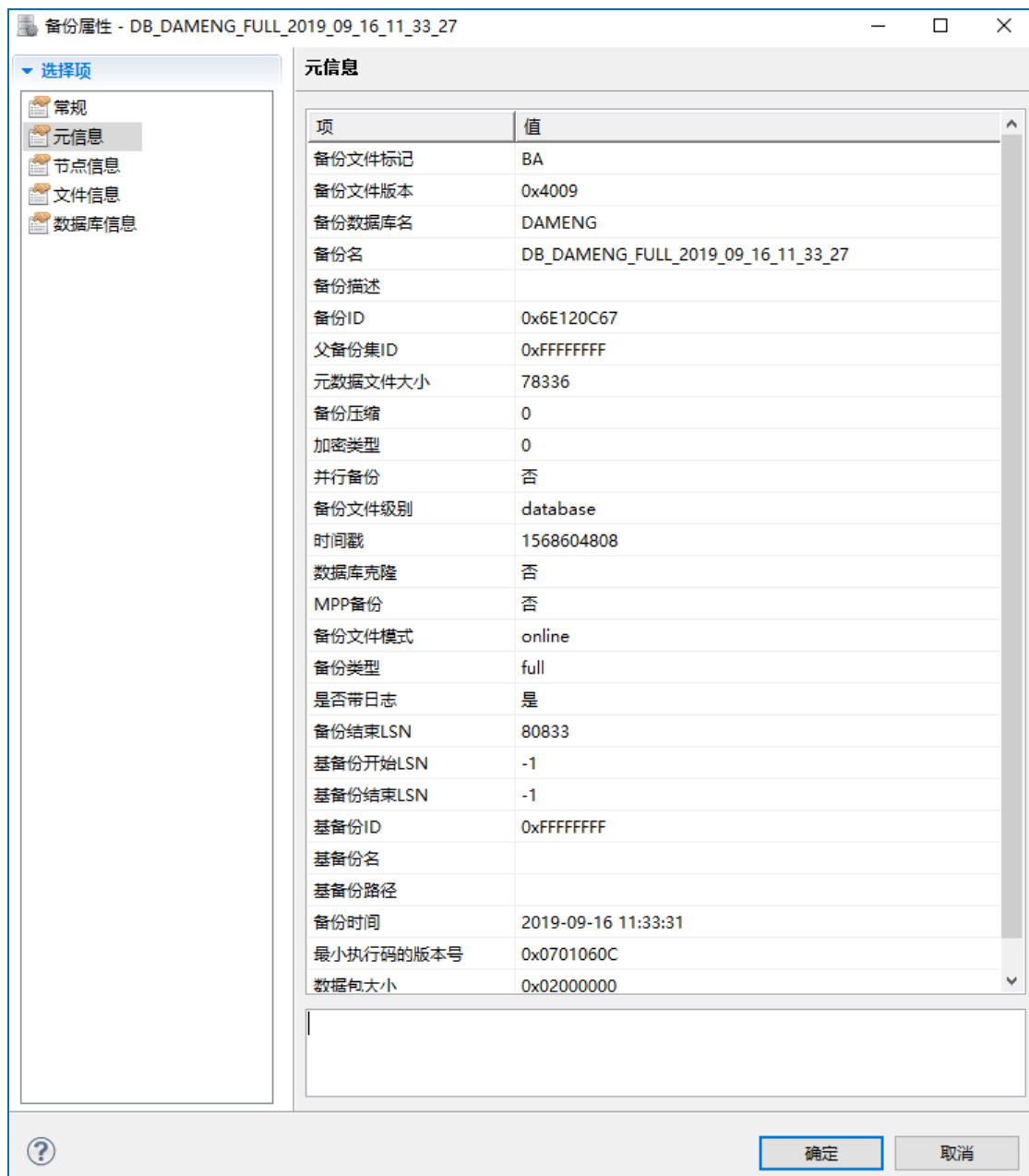
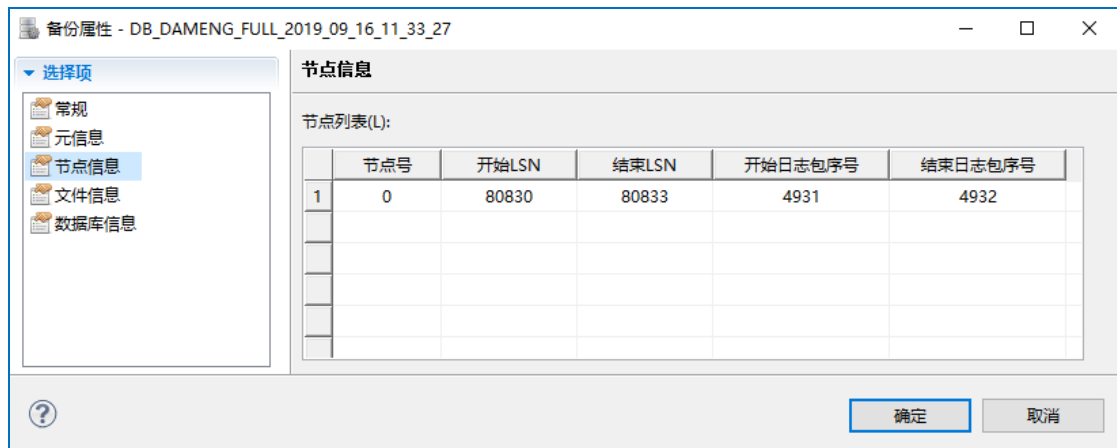


图 3.20 备份集-元数据信息

节点信息显示系统中各节点 END\_LSN/END\_SEQ 信息，开始/结束日志包序号。单库只有一个节点，集群系统中会有多个节点。如下图所示：



3.141 备份集-节点信息

文件信息显示备份集的所有数据文件和备份片信息。如下图所示：



图 3.152 备份集-文件信息

数据库信息显示备份库的固有信息。如下图所示：

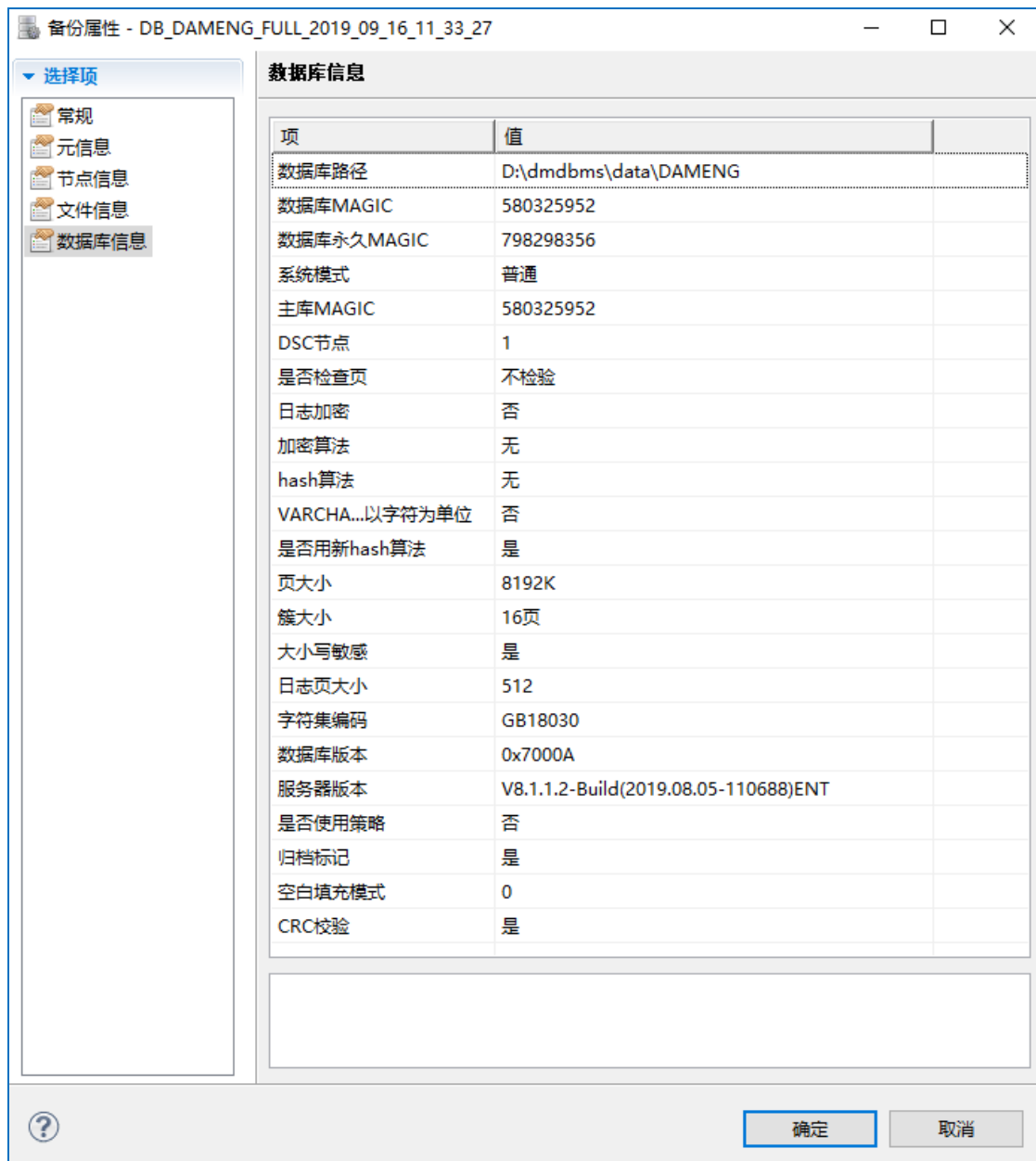


图 3.163 备份集-数据库信息

### 3.4.2.3 数据还原

#### 数据库还原

在备份还原管理主界面中，点击还原按钮，打开备份还原对话框。还原有三种类型：库还原（图 3.23）、归档还原（图 3.24）、表空间还原（图 3.25）和直接还原（图 3.26）。

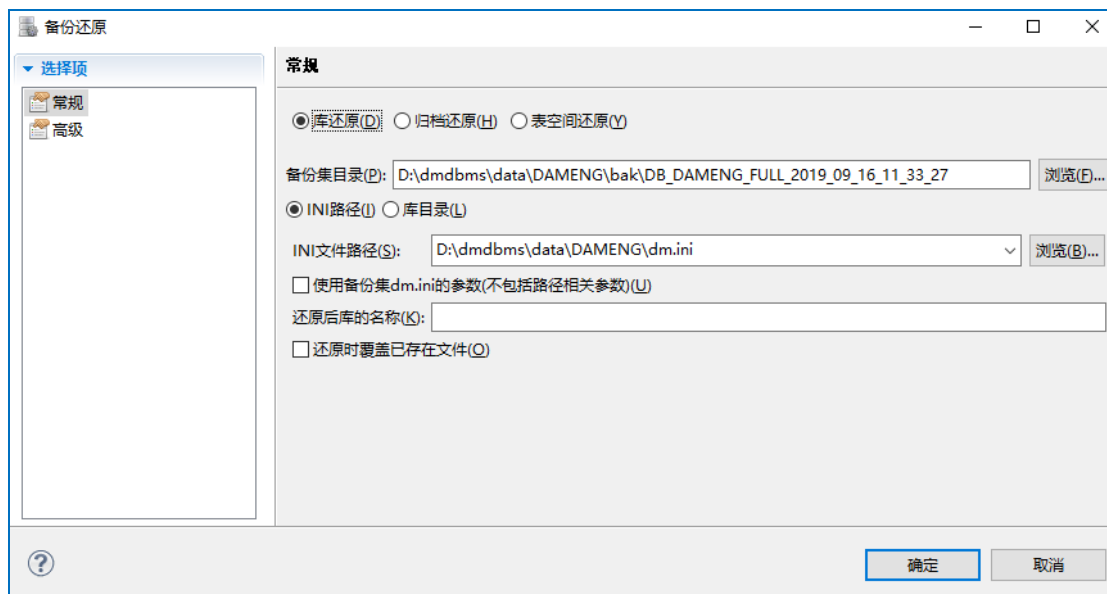


图 3.174 备份集还原常规页面--库还原

需要配置以下参数：

**备份集目录：**指定用于还原目标数据库的备份集目录。若指定为相对路径，会在基备份集搜索目录下搜索备份集。

**INI 文件路径：**指定还原库目标的 dm.ini 文件路径。此时，还可以使用备份集 dm.ini 的参数，表示会将备份集中备份的 dm.ini 参数（除路径相关的 INI 参数外），拷贝到当前 dm.ini 上，取而代之。

**库目录：**待还原的库所在目录，该路径必须已存在。此时，还可以选用还原时覆盖已存在的文件，表示如果待还原的文件已存在，直接覆盖，否则报错。

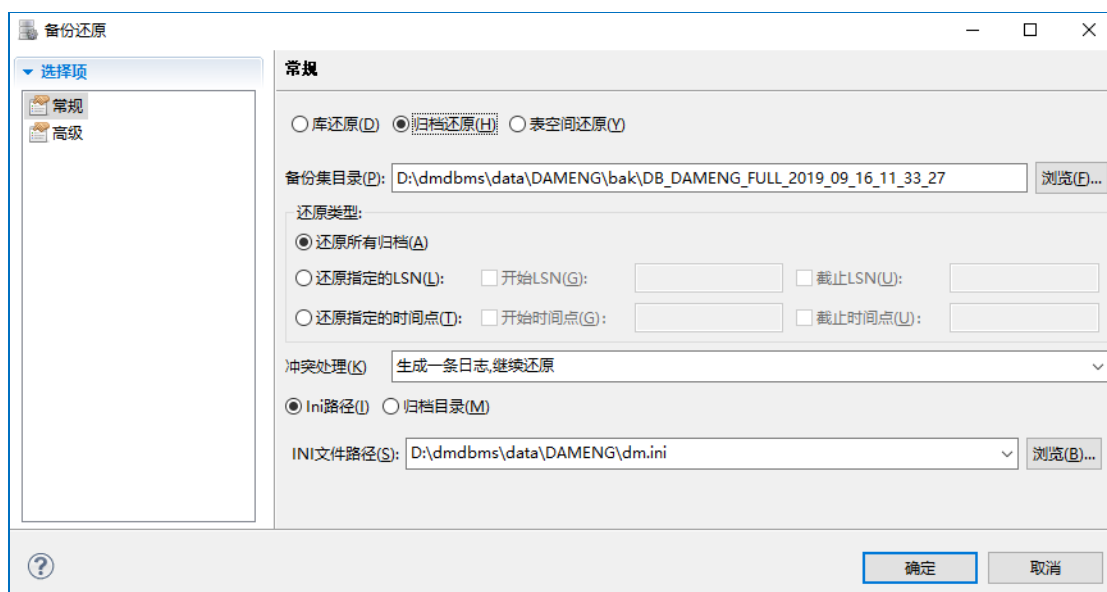


图 3.185 备份集还原常规页面--归档还原



需要配置以下参数：

**备份集目录：**指定用于还原目标数据库的备份集目录。若指定为相对路径，会在基备份集搜索目录下搜索备份集。

**还原所有归档：**还原所有的归档。

**还原指定的 LSN：**指定还原的开始 LSN 和结束 LSN。

**还原指定的时间点：**指定备份的开始时间点和结束时间点。

**冲突处理：**生成一条日志，继续还原；直接报错返回；强制覆盖存在的归档日志。

**INI 路径：**还原到 ini 对应库的本地归档配置目录中。即 dmarch.ini 指定的第一个本地归档目录。

**归档目录：**还原到指定的目录。

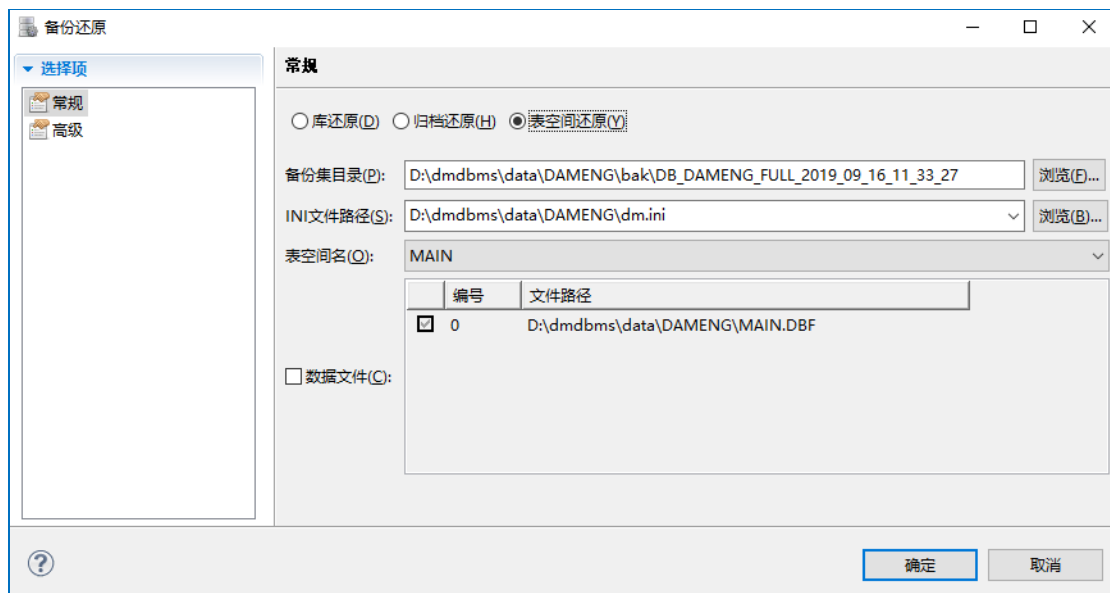


图 3.196 备份集还原常规页面--表空间还原

需要配置以下参数：

**备份集目录：**指定用于还原目标数据库的备份集目录。若指定为相对路径，会在基备份集搜索目录下搜索备份集。

**INI 文件路径：**指定还原库目标的 dm.ini 文件路径。

**表空间名：**指定还原的表空间，可以是用户表空间，也可以是 SYSTEM 系统表空间和 ROLL 回滚表空间。

**数据文件：**还原指定的数据文件。可以指定数据文件编号或数据文件路径。文件编号，对应动态视图 V\$DATAFILE 中 ID 列的值；文件路径，对应动态视图 V\$DATAFILE 中 PATH 或者 MIRROR\_PATH 列的值，也可以仅指定数据文件名称（相对路径），与表空间中数据文

件匹配时，会使用 SYSTEM 目录补齐。

**归档日志目录：**本地归档日志搜索目录，若未指定，则仅使用目标库配置的第一个本地归档。

**截止时间点：**恢复数据库到指定的时间点。

**截止 LSN：**恢复数据库到指定的 LSN。

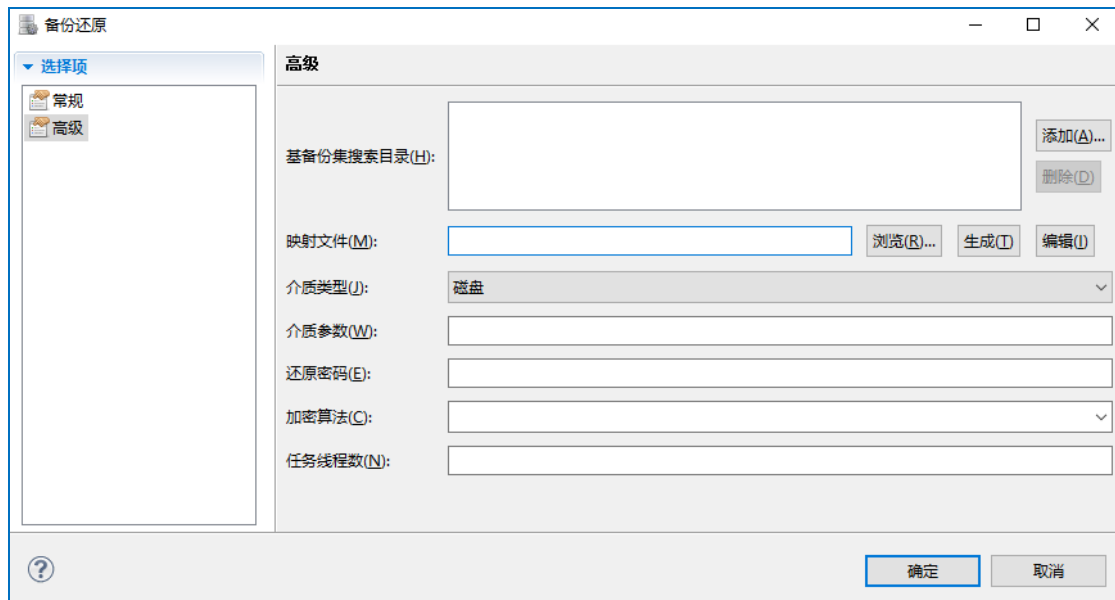


图 3.207 备份集还原常规页面--高级

需要配置以下参数：

**基备份集搜索目录：**指定基备份集搜索目录。若指定的备份集目录为相对路径，会在此目录下搜索备份集。

**映射文件：**用于指定存放还原目标路径，即备份集里面的数据文件的路径。也可以生成或编辑映射文件。当参数备份集目录和映射文件指定的路径不一致时，以映射文件中指定的路径为主。

**介质类型：**指存储备份集的介质类型，支持磁盘和磁带，默认为磁盘。磁盘表示备份集存储介质为磁盘，磁带表示存储介质为磁带。

**介质参数：**供第三方存储介质（TAPE 类型）管理使用。

**还原密码：**指定备份时使用的加密密码，供还原过程解密使用。

**加密算法：**指定备份时使用的加密算法，供还原过程解密使用，若未指定，则使用默认算法。可以使用第三方加密算法。

**任务线程数：**指定还原过程中用于处理压缩和解密任务的线程个数，取值范围 1~64。若未指定，则默认为 4。

非并行：指定并行备份集使用非并行方式还原。对于非并行备份集，不论是否指定该关键字，均采用非并行还原。

## 生成映射文件

数据库还原时可选择映射文件进行还原，映射文件用来指定还原后数据库的数据文件路径（具体参见 [3.3.5.2.1 数据库还原](#)）。CONSOLE 工具中生成映射文件有两种方式，一种是在备份还原页面的映射文件选项中点击生成按钮，另外一种是在备份还原主页面中，点击生成映射文件按钮。这里介绍后面一种映射文件生成方式。

打开生成映射文件对话框，显示如图 3.28 所示。

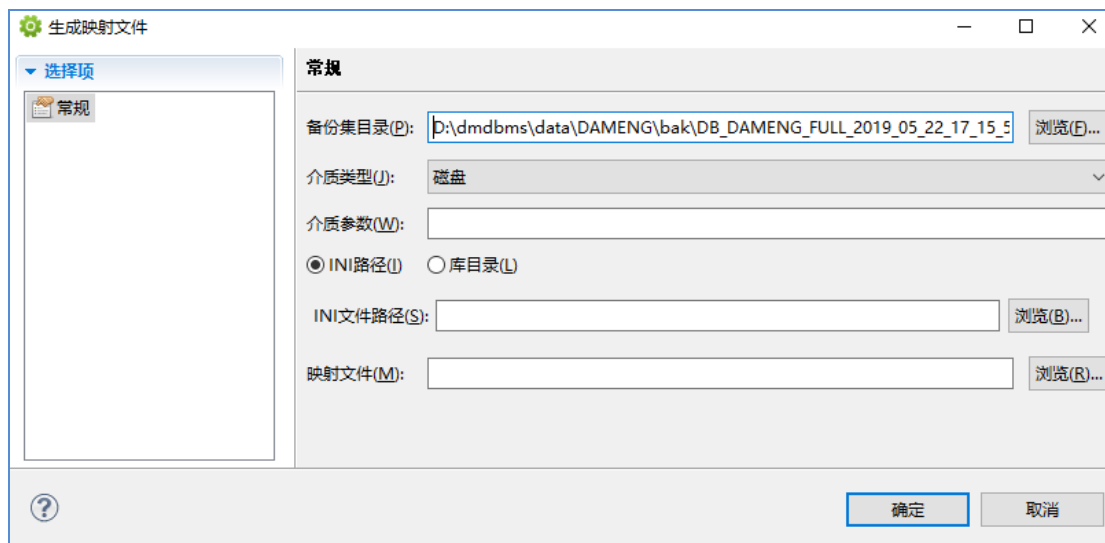


图 3.218 生成映射文件对话框

需要配置以下参数：

备份集目录：必填，待生成映射文件的备份集目录。

介质类型：指存储备份集的介质类型，暂支持磁盘和磁带，默认磁盘。

INI 文件路径：选填，目标还原库的 dm.ini 文件路径。

映射文件：必填，生成映射文件路径。

根据实际需要配置好上述参数后，点击确定按钮，执行生成映射文件操作。如果生成映射文件成功，则弹出生成映射文件成功的对话框；如果生成映射文件失败，则弹出对话框提示失败原因。

## 3.4.2.4 数据恢复

在备份还原管理界面中，点击恢复按钮，打开备份恢复对话框。有两种级别的恢复：

库恢复和表空间恢复。

### 3.4.2.4.1 库恢复

数据库恢复有两种类型。

- 从备份集恢复
- 指定归档恢复

以下分别进行介绍。

#### 从备份集恢复

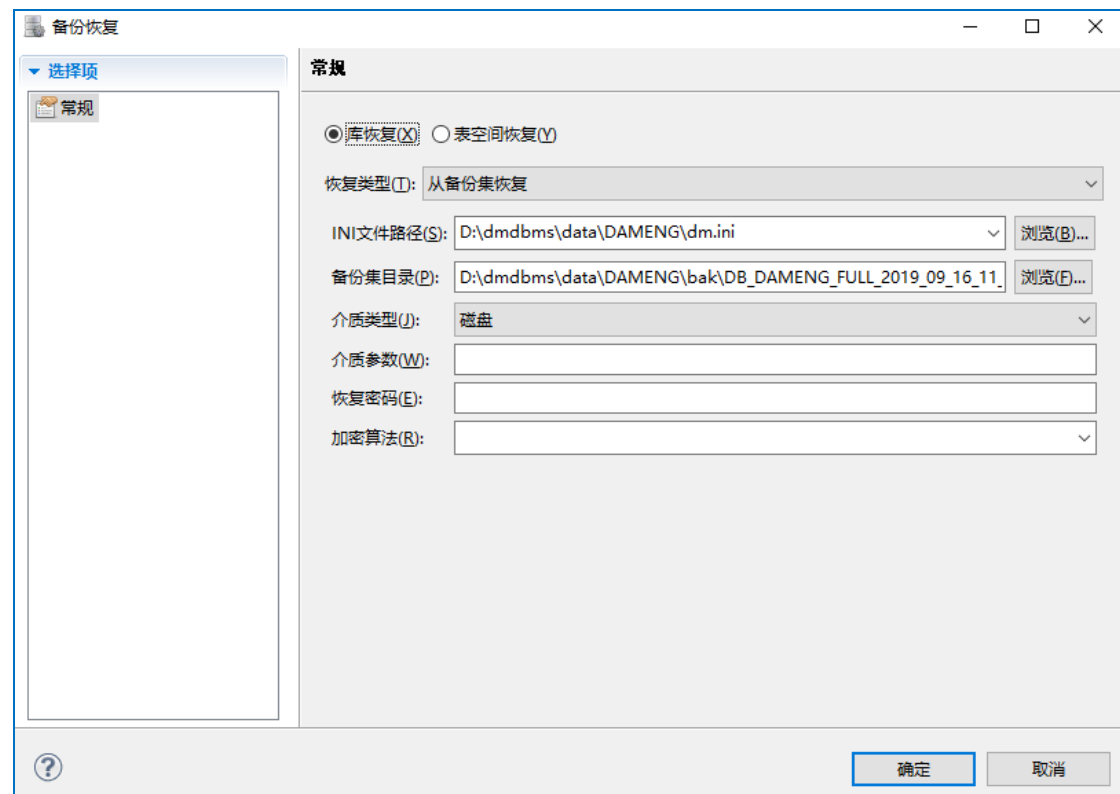


图 3.29 从备份集恢复页面

如果选择从备份集恢复，则需要配置以下参数：

INI 文件路径：必填，待恢复目标数据库 dm.ini 文件路径。

备份集目录：必填，指定待恢复备份集目录。

介质类型：指存储备份集的介质类型，暂支持 DISK 和 TAPE，默认 DISK。

介质参数：介质类型为 TAPE 时，第三方介质管理实现所需的参数字符串。

恢复密码：待恢复备份集，如果备份时使用了加密密码，则恢复时必须指定与备份时一

致的密码，供恢复过程解密使用。

加密算法：待恢复备份集，如果恢复时使用了加密算法，在恢复时必须指定与备份时一致的加密算法供恢复过程解密使用。支持使用第三方加密算法。

## 指定归档恢复

指定归档恢复，主要有两种类型：

1) 没有经过还原操作，直接在目标库上执行恢复操作。这种恢复主要针对一些已经滞后的库，用最新的归档把目标库恢复到某个时间点，或者恢复到最新的状态。

2) 经过了还原操作，这时需要借助归档把目标恢复到一个指定的时间点。可能还原的备份集是 WITHOUT LOG 的备份集，没有办法执行从备份集恢复；或者希望恢复到更新的状态，而从备份集恢复又不能够满足要求，所以需要借助归档恢复到某个时间点。

在未指定恢复时间或者恢复 LSN 号时，指定归档恢复默认恢复到最新状态，有多少归档就会重做多少。

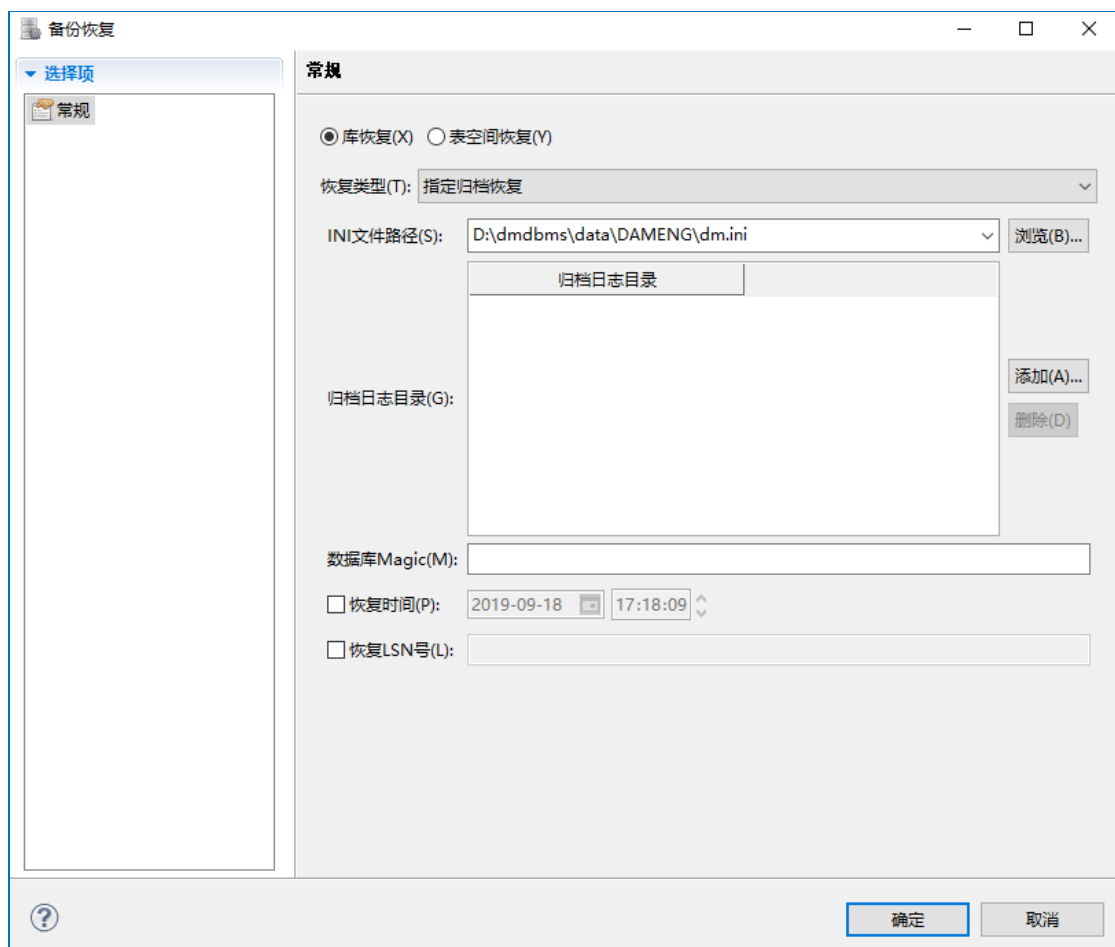


图 3.30 指定归档恢复页面

如果选择指定归档恢复，则需要配置以下参数：

INI 文件路径：必填，待恢复目标数据库 dm.ini 文件路径。

归档日志目录：必填，通过[添加]、[删除]按钮配置归档日志目录。

数据库 Magic：指定本地归档日志对应数据库的 Magic，若不指定，则默认使用目标恢复数据库的 Magic。

恢复时间：指定重做本地归档结束时间。

恢复 LSN 号：指定重做本地归档的结束 LSN 值。

根据实际需要选择对应的恢复类型，配置对应的参数后，点击[确定]按钮即开始执行对应恢复操作。如果恢复成功，则弹出恢复成功的对话框；如果恢复失败，则弹出对话框提示失败原因。

## 3.4.2.4.2 表空间恢复

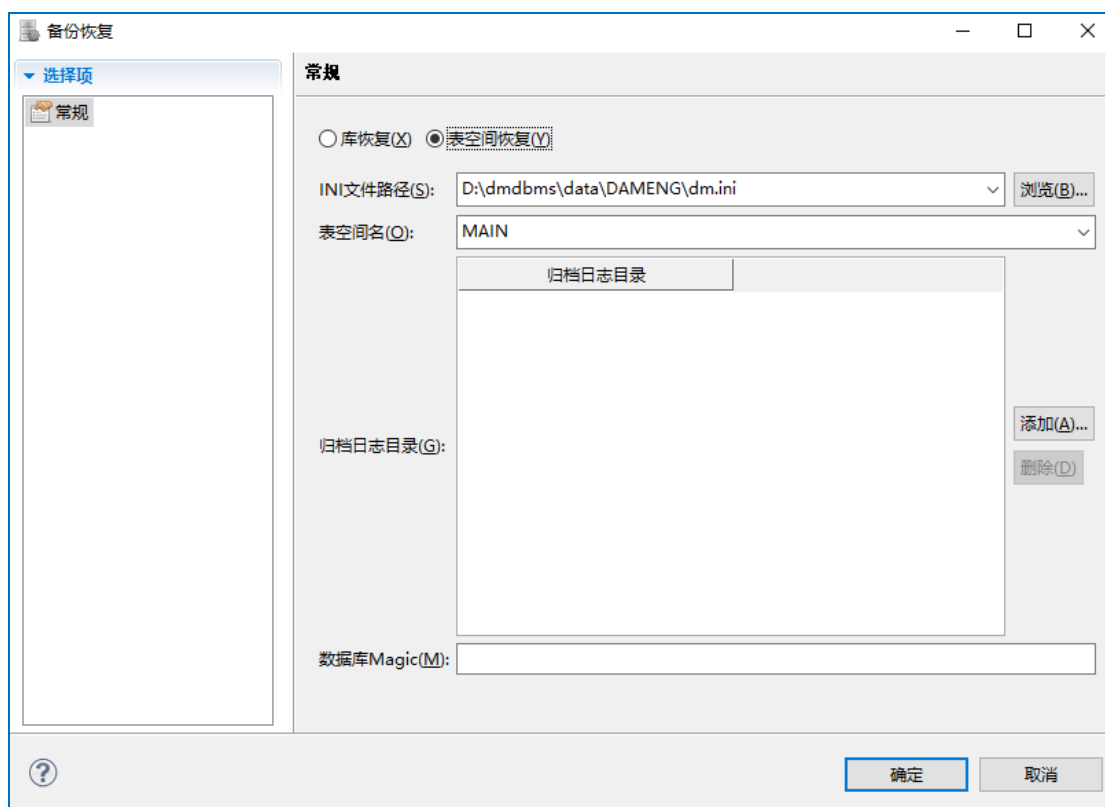


图 3.31 表空间恢复页面

如果选择表空间恢复，则需要配置以下参数：

INI 文件路径：必填，待恢复目标表空间所在数据库 dm.ini 文件路径。

归档日志目录：必填，通过[添加]、[删除]按钮配置归档日志目录。

数据库 Magic：指定本地归档日志对应数据库的 Magic，若不指定，则默认使用目标

恢复数据库的 Magic。

配置对应的参数后，点击**确定**按钮即开始执行对应恢复操作。如果恢复成功，则弹出恢复成功的对话框；如果恢复失败，则弹出对话框提示失败原因。

### 3.4.2.5 数据库更新

更新 DB\_MAGIC 是数据库还原之后，必须执行的一步。表空间等其他类型的还原不需要执行这一步。

对于联机备份的备份集，对目标库执行完还原操作后，如果仅希望把目标库恢复到备份的时间点，可以借助更新 DB\_MAGIC 把目标库恢复到备份结束的时间点。这种情况的恢复要求联机备份集是使用 WITHOUT LOG 类型的备份，且要注意只能恢复到备份结束的时间点。

对于脱机备份的备份集，或者联机备份的备份集中不需要重做联机日志的情况，在执行完还原操作后允许执行更新 DB\_MAGIC 操作，仅仅把目标库恢复到备份结束的时间点。



图 3.32 更新 DB\_MAGIC 页面

如果选择更新 DB\_MAGIC 恢复，则需要配置以下参数：

INI 文件路径：必填，待恢复目标数据库 dm.ini 文件路径。

### 3.4.2.6 归档修复

归档修复会将目标库 dmarch.ini 中配置的所有本地归档日志目录执行修复。

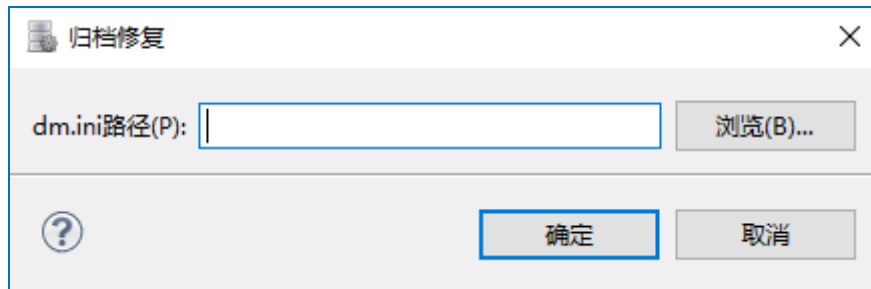


图 3.33 归档修复页面

如果选择归档恢复，则需要配置以下参数：

INI 文件路径：指定待修复归档的数据库对应的 dm.ini 路径。

### 3.4.2.7 默认配置

使用默认配置，配置 CONSOLE 还原时默认的存储介质类型、备份集搜集目录、归档日志搜集目录、跟踪日志文件。

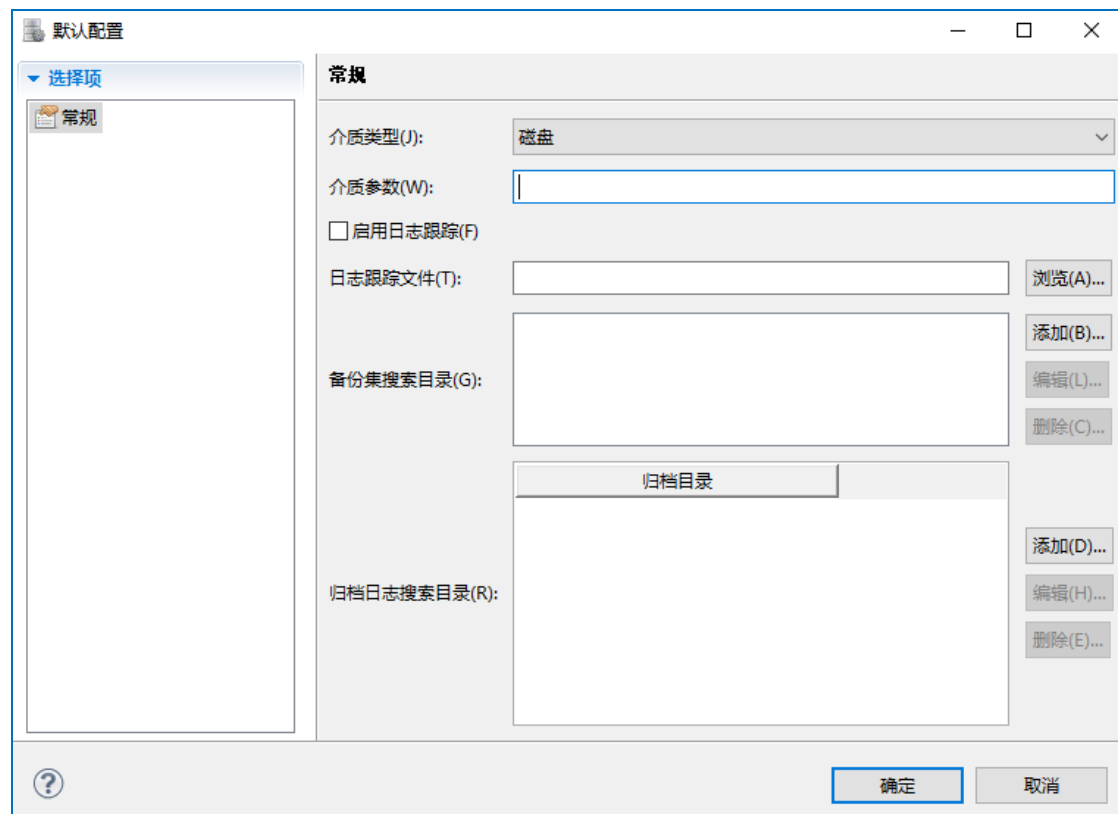


图 3.34 默认配置页面

如果选择默认配置，则需要配置以下参数：

介质类型：备份集存储的介质类型，磁盘（DISK）或者磁带（TAPE）。

介质参数：供第三方存储介质（TAPE 类型）管理使用。



跟踪日志文件：介质存储过程中使用的跟踪日志文件路径。

备份集搜集目录：备份集所在的目录。

归档日志搜集目录：归档日志所在的目录。

### 3.4.2.8 校验备份集

校验备份集是否合法。

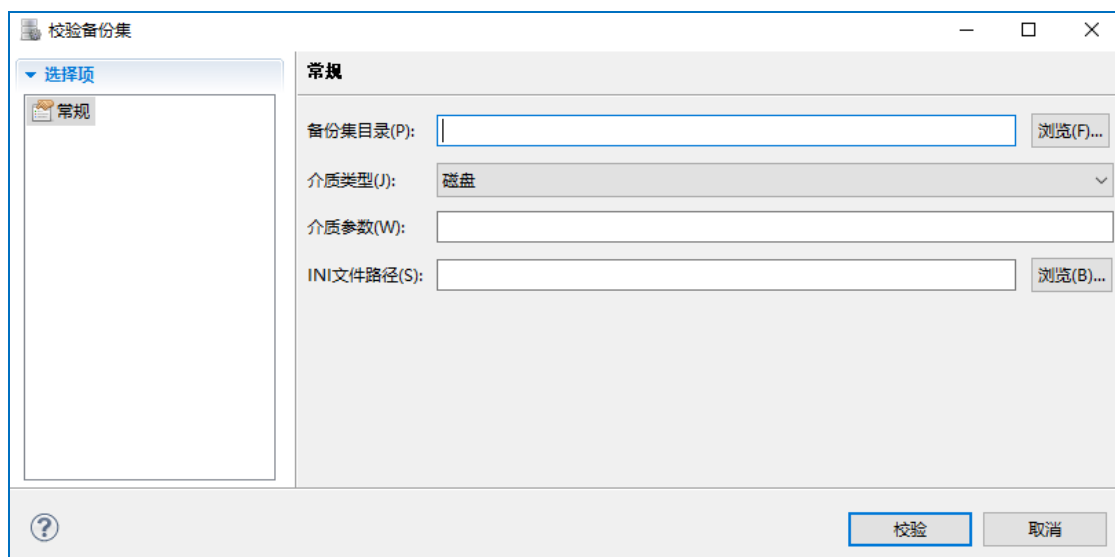


图 3.35 校验备份集页面

如果选择校验备份集，则需要配置以下参数：

备份集目录：指定目标校验备份集目录。

介质类型：指存储备份集的介质类型，支持磁盘或磁带。

介质参数：介质类型为磁带时，第三方介质（tape 类型）管理实现所需的参数字符串。

INI 文件路径：数据库 dm.ini 文件路径，若指定，则该数据库的默认备份目录作为备份集搜索目录之一。

## 4 联机拷贝数据库还原恢复

本章节介绍使用联机拷贝数据库进行还原恢复的方式，主要包括：

- 概述
- 拷贝文件
- 修改文件路径参数
- 还原恢复

### 4.1 概述

使用备份集进行还原恢复，在还原阶段，通过重建数据文件、联机日志文件，更新控制文件等操作重建了目标库的文件结构，并在恢复阶段重做日志将目标库恢复到最新状态。理论上，拷贝源库得到的一个副本，相当于目标从备份集还原后的状态，利用归档日志应该能将该副本恢复到最新状态，以下章节将对该还原恢复方式进行介绍。

### 4.2 约束与限制

1. 调用SP\_BACKUP\_COPY\_BEGIN前需设置当前连接AUTO\_COMMIT为FALSE，关闭事务自动提交。
2. SP\_BACKUP\_COPY\_BEGIN和SP\_BACKUP\_COPY\_END必须在同一连接中执行。
3. 每次调用SP\_BACKUP\_COPY\_END前必须先调用SP\_BACKUP\_COPY\_BEGIN，否则将报错。
4. 调用SP\_BACKUP\_COPY\_BEGIN和SP\_BACKUP\_COPY\_END之间不能执行CREATE TABLESPACE操作。

### 4.3 拷贝文件

联机拷贝数据库，主要包含三个步骤：

1. 拷贝开始前，记录系统各节点BEGIN\_LSN/BEGIN\_SEQ信息，以确定恢复阶段日志重做的起点。

开始拷贝前，调用系统过程SP\_BACKUP\_COPY记录当前系统BEGIN\_LSN/BEGIN\_SEQ信息。

```
SP_BACKUP_COPY_BEGIN()
```

2. 开始拷贝数据库，使用常规方式拷贝数据库文件到新的目录。

拷贝数据库文件到备份目录，拷贝数据库文件时，必须包含配置文件（dm.ini），控制文件（dm.ctl），密钥文件（dm\_service.prikey），联机日志文件（DAMENG01.log和DAMENG02.log），以及数据文件（\*.DBF）。

```
cp /home/dm/DB_FOR_COPY/DAMENG/dm.ini /home/dm/DB_COPY/
cp /home/dm/DB_FOR_COPY/DAMENG/dm.ctl /home/dm/DB_COPY/
cp /home/dm/DB_FOR_COPY/DAMENG/dm_service.prikey /home/dm/DB_COPY/
cp /home/dm/DB_FOR_COPY/DAMENG/DAMENG01.log /home/dm/DB_COPY/
cp /home/dm/DB_FOR_COPY/DAMENG/DAMENG02.log /home/dm/DB_COPY/
cp /home/dm/DB_FOR_COPY/DAMENG/*.DBF /home/dm/DB_COPY/
```

3. 拷贝结束后，调用系统过程SP\_BACKUP\_COPY\_END，记录系统各节点END\_LSN/END\_SEQ信息，以在恢复结束后校验数据完整性。该过程同时生成一个仅包含meta文件的空备份集，并将记录的信息填充到该文件。

SP\_BACKUP\_COPY\_END(path varchar)中path参数指定备份集绝对路径，当参数为相对路径时，将在默认备份路径下生成该备份集。

```
SP_BACKUP_COPY_END('/home/dm/bak/BAKSET_COPY')
```



联机拷贝数据库前必须已经配置了归档，否则调用 SP\_BACKUP\_COPY\_END

**注意：**将报错；该过程产生的空备份集只能用于源库副本的还原，当利用该备份集还原其他库或从该备份集进行恢复时将报错。

## 4.4修改文件路径参数

当源库的一个副本拷贝到不同的路径时，在还原恢复前需要修改文件中全部的路径参数。

例如，修改dm.ini文件中参数CTL\_PATH的路径。

```
CTL_PATH = /home/dm/DB_FOR_COPY/DAMENG/dm.ctl #ctl file path
```

改为

```
CTL_PATH = /home/dm/DB_COPY/dm.ctl #ctl file path
```

其他路径参数的修改方法类似。控制文件中路径参数修改方法比较特殊，控制文件修改之前需要从ctl文件转换为txt文件，修改之后，再转为ctl文件。

例如，修改dm.ctl文件中的路径参数步骤如下：

首先，将dm.ctl文件转为文本文件。

```
dmctlcvt TYPE=1 SRC=/home/dm/DB_COPY/dm.ctl DEST=/home/dm/DB_COPY/dmctl.txt
```

其次，修改dmctl.txt中的路径参数。将原始路径修改为拷贝副本所在的路径。

```
# file path
fil_path=/home/dm/DB_FOR_COPY/DAMENG/SYSTEM.DBF
```

改为

```
# file path
fil_path=/home/dm/DB_COPY/SYSTEM.DBF
```

最后，将修改完成后的dmctl.txt转为二进制文件dm.ctl，并替换原始dm.ctl文件。

```
dmctlcvt TYPE=2 SRC=/home/dm/DB_COPY/dmctl.txt DEST=/home/dm/DB_COPY/dm.ctl
```

## 4.5 还原恢复

从源库的副本进行还原恢复，语法与脱机库还原恢复完全一致，语法参考3.3.5 [数据库还原和恢复](#)。

利用空备份集还原源库的副本。

```
RESTORE DATABASE '/home/dm/DB_COPY/dm.ini' FROM BACKUPSET
'/home/dm/bak/BAKSET_COPY'
```

利用源库的归档日志将副本恢复到最新状态。

```
RECOVER DATABASE '/home/dm/DB_COPY/dm.ini' WITH ARCHIVEDIR
'/home/dm/DB_FOR_COPY/DAMENG/arch'
```

恢复成功后更新副本DB\_MAGIC。

```
RECOVER DATABASE '/home/dm/DB_COPY/dm.ini' UPDATE DB_MAGIC
```

## 4.6 高级主题

DSC环境下，联机拷贝进行还原恢复的操作流程与单机环境基本相同，只在某些操作略

有差别，下面以使用模拟共享磁盘搭建的两节点DSC为例进行介绍。

### 拷贝文件

开始拷贝前，调用系统过程SP\_BAKCUP\_COPY\_BEGIN，与单机环境相同，可连接任意节点执行。

```
SP_BACKUP_COPY_BEGIN()
```

拷贝文件时，由于DSC环境数据文件通过DMASM管理并存储在共享磁盘上，需拷贝所有ASM磁盘文件 (\*.asm)，DMASM配置文件 (dmasvrml.ini, dmdcr\_cfg.ini 以及 dmdcr.ini)，以及各节点配置文件 (dm.ini) 和MAL配置文件 (dmmal.ini)。

```
cp /home/dm/DSC_FOR_COPY/asmdisks/*.asm /home/dm/DSC_COPY/asmdisks/
cp /home/dm/DSC_FOR_COPY/dmdcr_cfg.ini /home/dm/DSC_COPY/
cp /home/dm/DSC_FOR_COPY/dmasvrml.ini /home/dm/DSC_COPY/
cp /home/dm/DSC_FOR_COPY/dsc0/dmdcr.ini /home/dm/DSC_COPY/dsc0/
cp /home/dm/DSC_FOR_COPY/dsc1/dmdcr.ini /home/dm/DSC_COPY/dsc1/
cp /home/dm/DSC_FOR_COPY/dsc0/dm.ini /home/dm/DSC_COPY/dsc0/
cp /home/dm/DSC_FOR_COPY/dsc1/dm.ini /home/dm/DSC_COPY/dsc1/
cp /home/dm/DSC_FOR_COPY/dsc0/dmmal.ini /home/dm/DSC_COPY/dsc0/
cp /home/dm/DSC_FOR_COPY/dsc1/dmmal.ini /home/dm/DSC_COPY/dsc1/
```

拷贝结束后，调用系统过程SP\_BACKUP\_COPY\_END生成空备份集，这一步与单节点相同，可在任意节点执行。

```
SP_BACKUP_COPY_END('/home/dm/bak/BAKSET_DSC_COPY')
```

### 修改文件路径参数

当共享磁盘文件拷贝到不同的目录时，需修改相关路径参数。

一，修改dmdcr\_cfg.ini种的路径参数DCR\_VTD\_PATH和DCR\_EP\_ASM\_LOAD\_APTH。

以DCR\_VTD\_PATH为例进行说明。

拷贝前：

```
DCR_VTD_PATH = /home/dm/DSC_FOR_COPY/asmdisks/votedisk.asm
```

拷贝后，修改为：

```
DCR_VTD_PATH = /home/dm/DSC_COPY/asmdisks/votedisk.asm
```

dmdcr\_cfg.ini修改后，需重新初始化ASM磁盘，其中dcrdisk.asm为拷贝前被初始化为DCR磁盘的裸设备文件，IDENTIFIED BY后跟指定的密码。

```
ASM>INIT          DCRDISK          '/home/dm/DSC_COPY/asmdisks/dcrdisk.asm'          FROM  
'/home/dm/DSC_COPY/dmdcr_cfg.ini' IDENTIFIED BY 'password'
```

二，修改 dmdcr.ini 的路径参数 DMDCR\_PATH、DMDCR\_MAL\_PATH、DMDCR\_ASM\_STARTUP\_CMD和DMDCR\_DB\_STARTUP\_CMD。

以0号节点为例，修改DMDCR\_DB\_STARTUP\_CMD参数。

拷贝前：

```
DMDCR_DB_STARTUP_CMD = /dm/bin/dmserver path=/home/dm/DSC_FOR_COPY/dsc0/dm.ini  
dcr_ini=/home/dm/DSC_FOR_COPY/dsc0/dmdcr.ini
```

拷贝后，修改为：

```
DMDCR_DB_STARTUP_CMD = /dm/bin/dmserver path=/home/dm/DSC_COPY/dsc0/dm.ini  
dcr_ini=/home/dm/DSC_COPY/dsc0/dmdcr.ini
```

三，修改dm.ini中的CONFIG\_PATH。dm.ini中ASM路径参数不需要修改。

以0号节点为例，修改CONFIG\_PATH参数。

拷贝前：

```
CONFIG_PATH = /home/dm/DSC_FOR_COPY/dsc0      #config path
```

拷贝后，改为：

```
CONFIG_PATH = /home/dm/DSC_COPY/dsc0      #config path
```

四，修改 dmdcr\_cfg.ini，dmasvrml.ini，dmmal.ini 中的 IP 参数。以 dmdcr\_cfg.ini 中的DCR\_EP\_HOST参数为例。

拷贝前：

```
DCR_EP_HOST          = 192.168.0.111
```

拷贝后，修改为：

```
DCR_EP_HOST          = 192.168.0.222
```

### 还原恢复

DSC环境下的还原恢复操作与单机环境完全相同，请参考前述章节。

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

---