第 9 章　备份与恢复

数据库的备份与恢复是保证数据库安全运行的一项重要内容，也是数据库管理员的重要职责。数据库运行过程中出现故障是不可避免的，如何最大限度地恢复数据库，减少数据的丢失，这就需要合理地制定数据库的备份与恢复策略。本章将主要介绍数据库备份与恢复的概念、类型以及手动备份与恢复数据库、逻辑备份与恢复数据库、利用 RMAN 备份与恢复数据库。

9.1　备份与恢复概述

9.1.1　备份与恢复的概念

1．备份与恢复的作用

在以数据库为数据管理中心的信息系统中，由于数据库故障而导致业务数据部分或全部丢失、系统运行失败的情况时有发生。因此，如何有效地预防数据库故障的发生以及在数据库发生故障后如何快速、有效地恢复数据库系统是数据库管理员的重要任务，其解决方法就是合理制定数据库的备份与恢复策略，执行有效的数据库备份与恢复操作。

备份与恢复是数据库的一对相反操作，备份保存数据库中数据的副本，而恢复是在数据库出现故障时使用备份的副本恢复数据库。

在 Oracle 数据库中，既可以由管理员手动进行备份与恢复操作，也可以利用 Oracle 恢复管理器（RMAN）自动进行备份与恢复操作。

2．备份的概念与类型

数据库备份就是对数据库中部分或全部数据进行复制，形成副本，存放到一个相对独立的设备上，如磁盘、磁带，以备将来数据库出现故障时使用。

根据数据备份方式的不同，数据库备份分为物理备份和逻辑备份两类。物理备份是将组成数据库的数据文件、重做日志文件、控制文件、初始化参数文件等操作系统文件进行复制，将形成的副本保存到与当前系统独立的磁盘或磁带上。逻辑备份是指利用 Oracle 提供的导出工具（如 Expdp，Export）将数据库中的数据抽取出来存放到一个二进制文件中。通常，数据库备份以物理备份为主，逻辑备份为辅。

根据数据库备份时是否关闭数据库服务器，物理备份分为冷备份和热备份两种情况。冷备份又称停机备份，是指在关闭数据库的情况下将所有的数据库文件复制到另一个磁盘或磁带上去。热备份又称联机备份，是指在数据库运行的情况下对数据库进行的备份。要进行热备份，数据库必须运行在归档日志模式下。

根据数据库备份的规模不同，物理备份还可以分为完全备份和部分备份。完全备份是指对整个数据库进行备份，包括所有的物理文件。部分备份是对部分数据文件、表空间、控制文件、归档重做日志文件等进行备份。

根据数据库是否运行在归档模式，物理备份还可以分为归档备份和非归档备份。

3．恢复的概念、类型与恢复机制

数据库恢复是指在数据库发生故障时，使用数据库备份还原数据库，使数据库恢复到无故障状态。

根据数据库恢复时使用的备份不同，恢复分为物理恢复和逻辑恢复两类。所谓的物理恢复就是，利用物理备份来恢复数据库，即利用物理备份文件恢复损毁文件，是在操作系统级别上进行的。逻辑恢复是指利用逻辑备份的二进制文件，使用 Oracle 提供的导入工具（如 Impdp，Import）将部分或全部信息重新导入数据库，恢复损毁或丢失的数据。

根据数据库恢复程度的不同，恢复可分为完全恢复和不完全恢复。数据库出现故障后，如果能够利用备份使数据库恢复到出现故障时的状态，称为完全恢复，否则称为不完全恢复。

数据库的恢复分 3 个步骤进行：首先使用一个完整备份将数据库恢复到备份时刻的状态；然后利用归档日志文件和联机重做日志文件中的日志信息，采用前滚技术（Roll Forward）重做备份以后已经完成并提交的事物；最后利用回滚技术（Roll Back）取消发生故障时已写入日志文件但没有提交的事物，将数据库恢复到故障时刻的状态。

例如，在图 9-1 所示案例中，在 T1 和 T3 时刻进行了两次数据库备份，在 T5 时刻数据库出现故障。如果使用 T1 时刻的备份 1 恢复数据库，则只能恢复到 T1 时刻的状态，即不完全恢复，因为缺少从 T1 时刻到 T2 时刻的归档日志；如果使用 T3 时刻的备份 2 恢复数据库，则可以恢复到 T3 时刻到 T5 时刻的任意状态，因为从 T3 时刻到 T5 时刻所有的日志文件是完整的（归档日志与联机日志）。

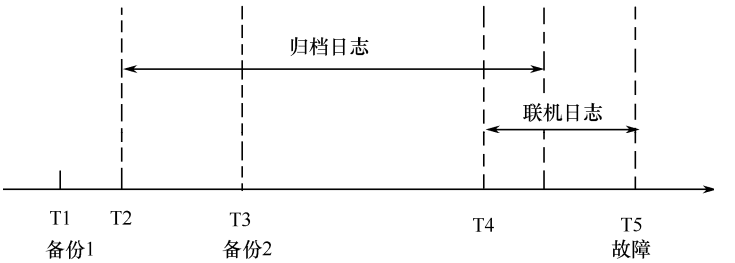

图 9-1　数据库恢复的过程

由图 9-1 的分析可以看出，如果数据库处于归档模式，且日志文件是完整的，则可以将数据库恢复到备份时刻后的任意状态，实现完全恢复或不完全恢复；如果数据库处于非归档模式，则只能将数据库恢复到备份时刻的状态，即实现不完全恢复。

9.1.2　Oracle 数据库故障类型及恢复措施

数据库在运行过程中可能出现多种类型的故障，不同类型的故障需要管理员采取不同的备份与恢复策略。

在 Oracle 数据库中常见的故障包括以下 6 种。

1．语句故障

语句故障是指执行 SQL 语句时发生的故障。例如，对不存在的表执行 SELECT 操作、向已无空间可用的表中执行 INSERT 操作等都会发生语句故障，Oracle 将返回给用户一个错误信息。语句故障通常不需要 DBA 干预，Oracle 会自动回滚产生错误的 SQL 语句操作。

2．进程故障

进程故障是指用户进程、服务器进程或数据库后台进程由于某种原因而意外终止，此时该

进程将无法使用，但不影响其他进程的运行。Oracle 的后台进程 PMON 能够自动监测并恢复故障进程。如果该进程无法恢复，则需要 DBA 关闭并重新启动数据库实例。

3．用户错误

用户错误是指用户在使用数据库时产生的错误。例如，用户意外删除某个表或表中的数据。用户错误无法由 Oracle 自动进行恢复，管理员可以使用逻辑备份来恢复。

4．实例失败

实例失败是指由于某种原因导致数据库实例无法正常工作。例如，突然断电导致数据库服务器立即关闭、数据库服务器硬件故障导致操作系统无法运行等。实例失败时，需要进行实例重新启动，在实例重新启动的过程中，数据库后台进程 SMON 会自动对实例进行恢复。

5．网络故障

网络故障是指由于通信软件或硬件故障，导致应用程序或用户与数据库服务器之间的通信中断。数据库的后台进程 PMON 将自动监测并处理意外中断的用户进程和服务器进程。

6．介质故障

介质故障是指由于各种原因引起的数据库数据文件、控制文件或重做日志文件的损坏，导致系统无法正常运行。例如，磁盘损坏导致文件系统被破坏。介质故障是数据库备份与恢复中主要关心的故障类型，需要管理员提前做好数据库的备份，否则将导致数据库无法恢复。

本章将主要针对介质故障的备份与恢复进行介绍。

9.2 物理备份与恢复

9.2.1 冷备份

如果数据库可以正常关闭，而且允许关闭足够长的时间，那么就可以采用冷备份（脱机备份），可以是归档冷备份，也可以是非归档冷备份。其方法是首先关闭数据库，然后备份所有的物理文件，包括数据文件、控制文件、联机重做日志文件等。

在 SQL*Plus 环境中进行数据库冷备份的步骤如下。

（1）启动 SQL*Plus，以 SYSDBA 身份登录数据库

（2）查询当前数据库所有数据文件、控制文件、联机重做日志文件的位置

```
SQL>SELECT file_name FROM dba_data_files;
SQL>SELECT member FROM v$logfile;
SQL>SELECT value FROM v$parameter WHERE name='control_files';
```

（3）关闭数据库

```
SQL>SHUTDOWN IMMEDIATE
```

（4）复制所有数据文件、联机重做日志文件以及控制文件到备份磁盘

可以直接在操作系统中使用复制、粘贴方式进行，也可以使用下面的操作系统命令完成：

```
SQL>HOST COPY 原文件名称  目标路径名称
```

（5）重新启动数据库

```
SQL>STARTUP
```

9.2.2 热备份

虽然冷备份简单、快捷，但是在很多情况下，例如数据库运行于 24×7 状态时（每天工作

24 小时，每周工作 7 天），没有足够的时间可以关闭数据库进行冷备份，这时只能采用热备份。

热备份是数据库在归档模式下进行的数据文件、控制文件、归档日志文件等的备份。

在 SQL*Plus 环境中进行数据库完全热备份的步骤如下。

（1）启动 SQL*Plus，以 SYSDBA 身份登录数据库

（2）将数据库设置为归档模式

由于热备份是数据库处于归档模式下的备份，因此在热备份之前需要保证数据库已经处于归档模式。可以执行 ARCHIVE LOG LIST 命令，查看当前数据库是否处于归档日志模式。如果没有处于归档日志模式，需要先将数据库转换为归档模式，并启动自动存档。关于数据库归档模式的设置详见 3.5 节的介绍。

（3）以表空间为单位，进行数据文件备份

① 查看当前数据库有哪些表空间，以及每个表空间中有哪些数据文件。

```
SQL>SELECT tablespace_name,file_name FROM dba_data_files
   ORDER BY tablespace_name;
```

② 分别对每个表空间中的数据文件进行备份，其方法为：

- 将需要备份的表空间（如 USERS）设置为备份状态

```
SQL>ALTER TABLESPACE USERS BEGIN BACKUP;
```

- 将表空间中所有的数据文件复制到备份磁盘

```
SQL>HOST COPY
   D:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\USERS01.DBF
   D:\ORACLE\BACKUP\USERS01.DBF
```

- 结束表空间的备份状态

```
SQL>ALTER TABLESPACE USERS END BACKUP;
```

对数据库中所有表空间分别采用该步骤进行备份。

（4）备份控制文件

通常应该在数据库物理结构做出修改之后，如添加、删除或重命名数据文件，添加、删除或修改表空间，添加或删除重做日志文件和重做日志文件组等，都需要重新备份控制文件。

① 将控制文件备份为二进制文件。

```
SQL>ALTER DATABASE BACKUP CONTROLFILE TO 'D:\ORACLE\BACKUP\CONTROL.BKP';
```

② 将控制文件备份为文本文件。

```
SQL>ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

（5）备份其他物理文件

① 归档当前的联机重做日志文件。

```
SQL>ALTER  SYSTEM ARCHIVE LOG CURRENT;
```

归档当前的联机重做日志文件，也可以通过日志切换完成。

```
SQL>ALTER  SYSTEM SWITCH LOGFILE;
```

② 备份归档重做日志文件，将所有的归档重做日志文件复制到备份磁盘中。

③ 备份初始化参数文件，将初始化参数文件复制到备份磁盘中。

9.2.3 非归档模式下数据库的恢复

非归档模式下数据库的恢复主要指利用非归档模式下的冷备份恢复数据库，其步骤如下。

① 关闭数据库。

```
SQL>SHUTDOWN IMMEDIATE
```

② 将备份的所有数据文件、控制文件、联机重做日志文件还原到原来所在的位置。

③ 重新启动数据库。

```
SQL>STARTUP
```

注意：非归档模式下的数据库恢复是不完全恢复，只能将数据库恢复到最近一次完全冷备份的状态。

9.2.4 归档模式下数据库的完全恢复

归档模式下数据库的完全恢复是指归档模式下一个或多个数据文件损坏，利用热备份的数据文件替换损坏的数据文件，再结合归档日志文件和联机重做日志文件，采用前滚技术重做自备份以来的所有改动，采用回滚技术回滚未提交的操作，以恢复到数据库故障时刻的状态。因此，数据库完全恢复的前提条件是归档日志文件、联机重做日志文件以及控制文件都没有损坏。

根据数据文件损坏程度的不同，数据库完全恢复可分为数据库级、表空间级、数据文件级 3 种类型。数据库级完全恢复主要应用于所有或多数数据文件损坏的恢复；表空间级完全恢复是对指定表空间中的数据文件进行恢复；数据文件级完全恢复是针对特定的数据文件进行恢复。

数据库级的完全恢复只能在数据库装载但没有打开的状态下进行，而表空间级完全恢复和数据文件级完全恢复可以在数据库处于装载状态或打开的状态下进行。

归档模式下数据库完全恢复的基本语法为：

```
RECOVER [AUTOMATIC] [FROM 'location']
[DATABASE|TABLESPACE tspname |DATAFILE dfname]
```

其中：

- AUTOMATIC：进行自动恢复，不需要 DBA 提供重做日志文件名称。
- location：制定归档重做日志文件的位置。默认为数据库默认的归档路径。

1．数据库级完全恢复

在 SQL*Plus 环境中进行数据库级完全恢复的步骤如下。

（1）如果数据库没有关闭，则强制关闭数据库

```
SQL>SHUTDOWN ABORT
```

（2）利用备份的数据文件还原所有损坏的数据文件

（3）将数据库启动到 MOUNT 状态

```
SQL>STARTUP MOUNT
```

（4）执行数据库恢复命令

```
SQL>RECOVER DATABASE
```

（5）打开数据库

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

2．表空间级完全恢复

下面以 EXAMPLE 表空间的数据文件 example01.dbf 损坏为例模拟表空间级的完全恢复。

（1）数据库处于装载状态下的恢复

① 如果数据库没有关闭，则强制关闭数据库。

```
SQL>SHUTDOWN ABORT
```

② 利用备份的数据文件 example01.dbf 还原损坏的数据文件 example01.dbf。

③ 将数据库启动到 MOUNT 状态。

```
SQL>STARTUP MOUNT
```

④ 执行表空间恢复命令。

```
SQL>RECOVER TABLESPACE EXAMPLE
```

⑤ 打开数据库。

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

（2）数据库处于打开状态下的恢复

① 如果数据库已经关闭，则将数据库启动到 MOUNT 状态。

```
SQL>STARTUP MOUNT
```

② 将损坏的数据文件设置为脱机状态。

```
SQL>ALTER DATABASE DATAFILE
   'D:\oracle\product\10.2.0\oradata\orcl\EXAMPLE01.DBF' OFFLINE;
```

③ 打开数据库。

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

④ 将损坏的数据文件所在的表空间脱机。

```
SQL>ALTER TABLESPACE EXAMPLE OFFLINE FOR RECOVER;
```

⑤ 利用备份的数据文件 example01.dbf 还原损坏的数据文件 example01.dbf。

⑥ 执行表空间恢复命令。

```
SQL>RECOVER TABLESPACE EXAMPLE;
```

⑦ 将表空间联机。

```
SQL>ALTER TABLESPACE EXAMPLE ONLINE;
```

如果数据文件损坏时数据库正处于打开状态，则可以直接执行步骤④～⑦。

3．数据文件级完全恢复

下面以数据文件 D:\oracle\product\10.2.0\oradata\orcl\example01.dbf 损坏为例模拟数据文件级的完全恢复。

（1）数据库处于装载状态下的恢复

① 如果数据库没有关闭，则强制关闭数据库。

```
SQL>SHUTDOWN ABORT
```

② 利用备份的数据文件 example01.dbf 还原损坏的数据文件 example01.dbf。

③ 将数据库启动到 MOUNT 状态。

```
SQL>STARTUP MOUNT
```

④ 执行数据文件恢复命令。

```
SQL>RECOVER DATAFILE
    'D:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\EXAMPLE01.DBF';
```

⑤ 将数据文件联机。

```
SQL>ALTER DATABASE DATAFILE
   'D:\oracle\product\10.2.0\oradata\orcl\EXAMPLE01.DBF' ONLINE
```

⑥ 打开数据库。

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

（2）数据库处于打开状态下的恢复

① 如果数据库已经关闭，则将数据库启动到 MOUNT 状态。

```
SQL>STARTUP MOUNT
```

② 将损坏的数据文件设置为脱机状态。

```
SQL>ALTER DATABASE DATAFILE
   'D:\oracle\product\10.2.0\oradata\orcl\EXAMPLE01.DBF' OFFLINE;
```

③ 打开数据库。

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

④ 利用备份的数据文件 example01.dbf 还原损坏的数据文件 example01.dbf。

⑤ 执行数据文件恢复命令。

```
SQL>RECOVER DATAFILE
   'D:\oracle\product\10.2.0\oradata\orcl\EXAMPLE01.DBF';
```

⑥ 将数据文件联机。

```
SQL>ALTER DATABASE DATAFILE
   'D:\oracle\product\10.2.0\oradata\orcl\EXAMPLE01.DBF' ONLINE;
```

如果数据文件损坏时数据库正处于打开状态，则可以直接执行步骤②、④～⑥。

4．数据库完全恢复示例

下面以 SYSTEM 表空间的数据文件 D:\oracle\product\10.2.0\oradata\orcl\system01.dbf 损坏为例演示归档模式下的完全恢复操作。

① 首先进行一次归档模式下的数据库完整备份。

② 以 SYSDBA 身份登录数据库进行下列操作。

```
SQL>CREATE TABLE test_rec(ID NUMBER PRIMARY KEY,NAME CHAR(20))
    TABLESPACE SYSTEM;
SQL>INSERT INTO test_rec VALUES(1,'ZHANGSAN');
SQL>COMMIT;
SQL>INSERT INTO test_rec VALUES(2,'LISI');
SQL>COMMIT;
SQL>ALTER SYSTEM SWITCH LOGFILE;
SQL>SELECT * FROM test_rec;
ID      NAME
-----------------
1       ZHANGSAN
2       LISI
SQL> SHUTDOWN ABORT;
```

③ 删除 SYSTEM 表空间的数据文件 D:\oracle\product\10.2.0\oradata\orcl\system01.dbf，以模拟数据文件损坏的情形。

④ 用备份的数据文件 D:\oracle\product\10.2.0\oradata\orcl\system01.dbf 还原损坏（本文为被删除）的数据文件 D:\oracle\product\10.2.0\oradata\orcl\system01.dbf。

⑤ 执行恢复操作。由于 SYSTEM 表空间不能在数据库打开后进行恢复，因此只能在数据库处于装载状态时进行恢复。

```
SQL>STARTUP MOUNT
SQL>RECOVER DATABASE;
SQL>ALTER DATABASE OPEN;
SQL>SELECT * FROM test_rec;
ID     NAME
-------------------
1      ZHANGSAN
2      LISI
```

9.2.5 归档模式下数据库的不完全恢复

1. 数据库不完全恢复概述

在归档模式下，数据库的不完全恢复主要是指归档模式下数据文件损坏后，没有将数据库恢复到故障时刻的状态。

在进行数据库不完全恢复之前，首先确保对数据库进行了完全备份；在不完全恢复后，需要使用 RESETLOGS 选项打开数据库，原来的重做日志文件被清空，新的重做日志文件序列号重新从 1 开始，因此原来的归档日志文件都不再起作用了，应该移走或删除；打开数据库后，应该及时备份数据库，因为原来的备份都已经无效了。

由于归档模式下，对数据文件只能执行前滚操作，而无法将已经提交的操作回滚，因此只能通过应用归档重做日志文件和联机重做日志文件将备份时刻的数据库向前恢复到某个时刻，而不能将数据库向后恢复到某个时刻。所以，在进行数据文件损坏的不完全恢复时必须先使用完整的数据文件备份将数据库恢复到备份时刻的状态。

如果数据库的所有多路镜像的控制文件都损坏了，那么可以使用备份的控制文件进行恢复。

不完全恢复分为以下 3 种类型。

- 基于时间的不完全恢复：将数据库恢复到备份与故障时刻之间的某个特定时刻。
- 基于撤销的不完全恢复：数据库的恢复随用户输入 CANCEL 命令而中止。
- 基于 SCN 的不完全恢复：将数据库恢复到指定的 SCN 值时的状态。

不完全恢复的语法为：

```
RECOVER [AUTOMATIC] [FROM 'location'][DATABASE]
[UNTIL TIME time|CANCEL|CHANGE scn]
[USING BACKUP CONTROLFILE]
```

2. 数据文件损坏的数据库不完全恢复的步骤

数据文件损坏的数据库不完全恢复的基本步骤如下。

① 如果数据库没有关闭，则强制关闭数据库。

```
SQL>SHUTDOWN ABORT
```

② 用备份的所有数据文件还原当前数据库的所有数据文件，即将数据库的所有数据文件恢复到备份时刻的状态。

③ 将数据库启动到 MOUNT 状态。

```
SQL>STARTUP MOUNT
```

④ 执行数据文件的不完全恢复命令。

```
SQL>RECOVER DATABASE UNTIL TIME time;(基于时间恢复)
SQL>RECOVER DATABASE UNTIL CANCEL;（基于撤销恢复）
SQL>RECOVER DATABASE UNTIL CHANGE scn;（基于 SCN 恢复）
```

可以通过查询数据字典视图 V$LOG_HISTORY 获得时间和 SCN 的信息。

⑤ 不完全恢复完成后，使用 RESETLOGS 选项启动数据库。

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

3. 数据库不完全恢复的示例

数据库不完全恢复的 3 类方法操作过程基本相似，下面以基于时间的不完全恢复来演示操作的过程。

注意：为了避免由于不完全恢复操作失败导致数据库无法恢复，切记先对数据库进行一次归档模式下的完全备份。

```
SQL>CREATE TABLE scott.test(ID NUMBER PRIMARY KEY, NAME CHAR(10));
SQL>SET TIME ON
09:39:36 SQL>INSERT INTO scott.test VALUES(1,'WANG');
09:40:04 SQL>COMMIT;
09:40:07 SQL>ALTER SYSTEM SWITCH LOGFILE;
09:40:16 SQL>INSERT INTO scott.test VALUES(2,'ZHANG');
09:40:33 SQL>COMMIT;
09:40:35 SQL>ALTER SYSTEM SWITCH LOGFILE;
09:40:48 SQL>INSERT INTO scott.test VALUES(3,'LI');
09:41:03 SQL>COMMIT;
09:41:05 SQL>ALTER SYSTEM SWITCH LOGFILE;
09:41:17 SQL>DELETE FROM scott.test WHERE id=2;
09:41:49 SQL>COMMIT;
09:41:54 SQL>ALTER SYSTEM SWITCH LOGFILE;
09:42:00 SQL>SELECT * FROM scott.test;
ID      NAME
----------------
1       WANG
3       LI
```

执行完上述操作后，通过查询数据字典视图 V$LOG_HISTORY 获取上述操作的日志信息。

```
09:42:42 SQL>ALTER SESSION SET NLS_DATE_FORMAT='YYYY-MM-DD HH24:MI:SS';
09:44:58 SQL>SELECT RECID,STAMP,SEQUENCE#,FIRST_CHANGE#, FIRST_TIME,
           NEXT_CHANGE# FROM V$LOG_HISTORY;
RECID STAMP  SEQUENCE#  FIRST_CHANGE# FIRST_TIME       NEXT_CHANGE#
-------------------------------------------------------------------------
  1    681210952   1     534907        2016-03-11 09:13:47   567860
  2    681210966   2     567860        2016-03-11 09:15:52   573783
  3    681212416   3     573783        2016-03-11 09:16:06   578274
  4    681212448   4     578274        2016-03-11 09:40:15   578286
  5    681212477   5     578286        2016-03-11 09:40:48   578303
  6    681212520   6     578303        2016-03-11 09:41:17   578319
```

如果此时用户发现删除数据的操作是错误的，或数据文件发生了损坏，需要恢复到删除操作之前的状态，此时就需要采用不完全恢复。通过前面的操作可以知道，删除操作对应的日志序列号为 6，第一个事务的 SCN 为 578303，起始时间为 2016-03-11 09:41:17。

```
10:32:21 SQL>SHUTDOWN ABORT
```

用所有的数据文件备份还原当前数据库的所有数据文件。

```
10:32:38 SQL>STARTUP MOUNT
10:35:42 SQL>RECOVER DATABASE UNTIL TIME '2016-03-11 09:41:17'
ORA-00279: 更改 577981 (在 03/11/2016 09:35:21 生成) 对于线程 1 是必需的
ORA-00289: 建议:
D:\ORACLE\PRODUCT\10.2.0\FLASH_RECOVERY_AREA\ORCL\ARCHIVELOG\2016_03_1
          1\ O1_MF_1_3_%U_.ARC
ORA-00280: 更改 577981 (用于线程 1) 在序列 #3 中
10:36:26 指定日志: {<RET>=suggested | filename | AUTO | CANCEL}
ORA-00279: 更改 578274 (在 03/11/2016 09:40:15 生成) 对于线程 1 是必需的
ORA-00289: 建议:
D:\ORACLE\PRODUCT\10.2.0\FLASH_RECOVERY_AREA\ORCL\ARCHIVELOG\2016_03_1
          1\ O1_MF_1_4_%U_.ARC
```

```
ORA-00280: 更改 578274 (用于线程 1) 在序列 #4 中
ORA-00278: 此恢复不再需要日志文件
'D:\ORACLE\PRODUCT\10.2.0\FLASH_RECOVERY_AREA\ORCL\ARCHIVELOG\2016_03_
          11\O1_MF_1_3_4VG5N07T_.ARC'
10:36:43 指定日志: {<RET>=suggested | filename | AUTO | CANCEL}
ORA-00279: 更改 578286 (在 03/11/2016 09:40:48 生成) 对于线程 1 是必需的
ORA-00289: 建议:
D:\ORACLE\PRODUCT\10.2.0\FLASH_RECOVERY_AREA\ORCL\ARCHIVELOG\2016_03_1
          1\ O1_MF_1_5_%U_.ARC
ORA-00280: 更改 578286 (用于线程 1) 在序列 #5 中
ORA-00278: 此恢复不再需要日志文件
'D:\ORACLE\PRODUCT\10.2.0\FLASH_RECOVERY_AREA\ORCL\ARCHIVELOG\2016_03_
          11\O1_MF_1_4_4VG5O05V_.ARC'
10:36:45 指定日志: {<RET>=suggested | filename | AUTO | CANCEL}
已应用的日志。
完成介质恢复。
10:36:50 SQL>ALTER DATABASE OPEN RESETLOGS;
10:37:42 SQL>SELECT * FROM scott.test;
ID     NAME
----------------
1      WANG
2      ZHANG
3      LI
```

此时，数据库恢复到了用户删除操作之前的状态。

在此例中，如果要使用基于 SCN 的不完全恢复，则应该将恢复命令修改为：

```
SQL> RECOVER DATABASE UNTIL CHANGE 578303;
```

4．控制文件损坏的数据库不完全恢复

① 如果数据库没有关闭，则强制关闭数据库。

```
SQL>SHUTDOWN ABORT
```

② 用备份的所有数据文件和控制文件还原当前数据库的所有数据文件、控制文件，即将数据库的所有数据文件、控制文件恢复到备份时刻的状态。

③ 将数据库启动到 MOUNT 状态。

```
SQL>STARTUP MOUNT
```

④ 执行不完全恢复命令。

```
SQL>RECOVER DATABASE UNTIL TIME time USING BACKUP CONTROLFILE;
SQL>RECOVER DATABASE UNTIL CANCEL USING BACKUP CONTROLFILE;
SQL>RECOVER DATABASE UNTIL CHANGE scn USING BACKUP CONTROLFILE;
```

⑤ 不完全恢复完成后，使用 RESETLOGS 选项启动数据库。

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

9.3 逻辑备份与恢复

9.3.1 逻辑备份与恢复概述

1．逻辑备份与恢复的特点

逻辑备份是指利用 Oracle 提供的导出工具，将数据库中选定的记录集或数据字典的逻辑副本以二进制文件的形式存储到操作系统中。这个逻辑备份的二进制文件称为转储文件，以

dmp 格式存储。逻辑恢复是指利用 Oracle 提供的导入工具将逻辑备份形成的转储文件导入数据库内部，进行数据库的逻辑恢复。

与物理备份与恢复不同，逻辑备份与恢复必须在数据库运行的状态下进行，因此当数据库发生介质损坏而无法启动时，不能利用逻辑备份恢复数据库。因此，数据库备份与恢复是以物理备份与恢复为主，逻辑备份与恢复为辅的。

逻辑备份与恢复有以下特点及用途：

① 可以在不同版本的数据库间进行数据移植，可以从 Oracle 数据库的低版本移植到高版本；

② 可以在不同操作系统上运行的数据库间进行数据移植，例如可以从 Windows 系统迁移到 UNIX 系统等；

③ 可以在数据库模式之间传递数据，即先将一个模式中的对象进行备份，然后再将该备份导入到数据库其他模式中；

④ 数据的导出与导入与数据库物理结构没有关系，是以对象为单位进行的，这些对象在物理上可能存储于不同的文件中；

⑤ 对数据库进行一次逻辑备份与恢复操作能重新组织数据，消除数据库中的链接及磁盘碎片，从而使数据库的性能有较大的提高；

⑥ 除了进行数据的备份与恢复外，还可以进行数据库对象定义、约束、权限等的备份与恢复。

2．数据泵技术

在 Oracle 9i 及其之前的数据库版本中，Oracle 数据库提供了 Export 和 Import 实用程序，用于实现数据库逻辑备份与恢复。在 Oracle 10g 数据库中又推出了数据泵技术，即 Data Pump Export（Expdp）和 Data Pump Import（Impdp）实用程序，实现数据库的逻辑备份与恢复。需要注意的是，这两类逻辑备份与恢复实用程序虽然在使用上非常相似，但之间并不兼容。使用 Export 备份的转储文件，不能使用 Impdp 进行导入；同样，使用 Expdp 备份的转储文件，也不能使用 Import 工具进行导入。

数据泵技术是 Oracle 10g 的新技术，与 Export，Import 是客户端实用程序不同，Expdp 和 Impdp 是服务器端实用程序，即 Export，Import 可以在服务器端使用，也可以在客户端使用，而 Expdp，Impdp 只能在数据库服务器端使用。因此，利用数据泵技术可以在服务器端多线程并行地执行大量数据的导出与导入操作。数据泵技术具有重新启动作业的能力，即当发生数据泵作业故障时，DBA 或用户进行干预修正后，可以发出数据泵重新启动命令，使作业从发生故障的位置继续进行。

由于 Expdp 和 Impdp 实用程序是服务器端程序，因此其转储文件只能存放在由 DIRECTORY 对象指定的特定数据库服务器操作系统目录，而不能使用直接指定的操作系统目录。所以，在使用 Expdp，Impdp 程序之前需要创建 DIRECTORY 对象，并将该对象的 READ，WRITE 权限授予用户。例如：

```
SQL>CREATE OR REPLACE DIRECTORY dumpdir AS 'D:\ORACLE\BACKUP';
SQL>GRANT READ,WRITE ON DIRECTORY dumpdir TO SCOTT;
```

此外，如果用户要导出或导入非同名模式的对象，还需要具有 EXP_FULL_DATABASE 和 IMP_FULL_DATABASE 权限。例如：

```
SQL>GRANT EXP_FULL_DATABASE,IMP_FULL_DATABASE TO SCOTT;
```

9.3.2 使用 Expdp 导出数据

1．Expdp 调用接口

Expdp 实用程序提供了 3 种应用接口供用户调用。

① 命令行接口（Command-Line Interface）：在命令行中直接指定参数设置。

② 参数文件接口（Parameter File Interface）：将需要的参数设置放到一个文件中，在命令行中用 PARFILE 参数指定参数文件。

③ 交互式命令接口（Interactive-Command Interface）：用户可以通过交互命令进行导出操作管理。

2．Expdp 导出模式

导出模式决定了所要导出的内容范围。Expdp 提供了 5 种导出模式，在命令行中通过参数设置来指定。

① 全库导出模式（Full Export Mode）：通过参数 FULL 指定，导出整个数据库。

② 模式导出模式（Schema Mode）：通过参数 SCHEMAS 指定，是默认的导出模式，导出指定模式中的所有对象。

③ 表导出模式（Table Mode）：通过参数 TABLES 指定，导出指定模式中指定的所有表、分区及其依赖对象。

④ 表空间导出模式（Tablespace Mode）：通过参数 TABLESPACES 指定，导出指定表空间中所有表及其依赖对象的定义和数据。

⑤ 传输表空间导出模式（Transportable Tablespace）：通过参数 TRANSPORT_TABLESPACES 指定，导出指定表空间中所有表及其依赖对象的定义。通过该导出模式以及相应导入模式，可以实现将一个数据库表空间的数据文件复制到另一个数据库中。

3．Expdp 帮助及参数

（1）获取 Expdp 帮助信息

在操作系统的命令提示符窗口中输入 expdp HELP=Y 命令，可以查看 Expdp 程序的使用、关键字（参数）、交互命令等介绍。例如：

```
C:\>expdp HELP=Y
Export: Release 10.2.0.1.0 - Production on 星期六, 14 3月, 2016 10:00:21
Copyright (c) 2003, 2005, Oracle.  All rights reserved.
    数据泵导出实用程序提供了一种用于在 Oracle 数据库之间传输数据对象的机制。该实用程序可以使用以下命令进行调用：
        示例: expdp scott/tiger DIRECTORY=dmpdir DUMPFILE=scott.dmp
    您可以控制导出的运行方式。具体方法是：在 'expdp' 命令后输入各种参数。要指定各参数，请使用关键字：
        格式:  expdp KEYWORD=value 或 KEYWORD=(value1,value2,...,valueN)
        示例: expdp scott/tiger DUMPFILE=scott.dmp DIRECTORY=dmpdir SCHEMAS=scott
                  或 TABLES=(T1:P1,T1:P2), 如果 T1 是分区表
    USERID 必须是命令行中的第一个参数。
    ……
```

（2）Expdp 参数详解

Expdp 命令的参数及其说明见表 9-1。

表 9-1　Expdp 命令参数及其说明

参 数 名 称	说　　明
ATTACH	把导出结果附加在一个已存在的导出作业中，默认为当前模式唯一的导出作业。语法为： ATTACH [=[schema_name.]job_name]
CONTENT	指定要导出的内容。语法为：CONTENT=[ALL | DATA_ONLY | METADATA_ONLY]。ALL 表示导出对象的定义及其数据；DATA_ONLY 表示只导出对象的数据；METADATA_ONLY 表示只导出对象的定义。默认为 ALL
DIRECTORY	指定转储文件和日志文件所在位置的目录对象，该对象由 DBA 预先创建。语法为： DIRECTORY=directory_object
DUMPFILE	指定转储文件名称列表，可以包含目录对象名，默认值为 expdat.dmp。语法为： DUMPFILE=[directory_object:]file_name [, ...]
ESTIMATE	指定用于估计导出作业中每个表中的数据占用磁盘空间大小的方法。语法为： ESTIMATE=[BLOCKS | STATISTICS]。默认值为 BLOCKS
ESTIMATE_ONLY	指定是否估计导出作业占用磁盘空间大小。语法为：ESTIMATE_ONLY=[YES | NO]。默认值为 NO
EXCLUDE	指定导出操作中要排除的对象类型和对象定义。语法为：EXCLUDE= object_type[:name_clause] [, ...]
FILESIZE	指定转储文件的最大尺寸。语法为：FILESIZE=integer[B | K | M | G]。默认值为 0，表示不受限制
FLASHBACK_SCN	指定导出操作时，允许数据库闪回到特定的 SCN。语法为：FLASHBACK_SCN=scn_value
FLASHBACK_TIME	指定导出操作时，用于获取最接近指定时间的 SCN，然后进行闪回。语法为： FLASHBACK_TIME="TO_TIMESTAMP(time-value)"
FULL	指定是否进行全数据库导出，包括所有数据及定义。语法为：FULL=[YES | NO]。默认值为 NO
HELP	指定是否显示 Export 命令的在线帮助。语法为：HELP = [YES| NO]。默认值为 NO
INCLUDE	指定导出操作中要导出的对象类型和对象定义。语法为：INCLUDE= object_type[:name_clause] [, ...]
JOB_NAME	指定导出作业的名称。语法为 JOB_NAME=jobname_string。默认值为系统自动为作业生成一个名称
LOGFILE	指定导出日志文件的名称。语法为：LOGFILE=[directory_object:]file_name。默认值为 export.log
NETWORK_LINK	指定网络导出时的数据库链接名称。语法为：NETWORK_LINK=source_database_link
NOLOGFILE	指定是否生成导出日志文件。语法为：NOLOGFILE=[YES|NO]。默认值为 NO
PARALLEL	指定执行导出作业时的并行进程最大个数。语法为：PARALLEL=integer。默认值为 1
PARFILE	指定导出参数文件的名称。语法为：PARFILE=[directory_path]file_name
QUERY	指定导出操作中 SELECT 语句中的数据导出条件。语法为： QUERY = [schema.][table_name:] query_clause
SCHEMAS	指定进行模式导出及模式列表。语法为：SCHEMAS=schema_name [, ...]。默认为当前模式
STATUS	指定显示导出作业状态的时间间隔。语法为：STATUS=[integer]。默认值为 0，表示操作结束时显示
TABLES	指定进行表模式导出及表列表。语法为：TABLES=[schema_name.]table_name[:partition_name] [, ...]
TABLESPACES	指定进行表空间模式导出及表空间列表。语法为：TABLESPACES=tablespace_name [, ...]
TRANSPORT_ FULL_CHECK	用于在传输表空间导出模式中指定是否进行导出表空间中的对象与非导出表空间对象间依赖关系的检查。语法为：TRANSPORT_FULL_CHECK=[YES|NO]。默认值为 NO
TRANSPORT_ TABLESPACES	指定进行传输表空间模式导出及表空间列表。语法为： TRANSPORT_TABLESPACES=tablespace_name [, ...]
VERSION	指定被导出的数据库对象的版本。语法为： VERSION=[COMPATIBLE | LATEST | version_string]。默认值为 COMPATIBLE

4．Expdp 应用实例

（1）命令行方式导出

① 表导出模式。表导出模式将一个或多个表的结构及其数据导出到转储文件中。导出表

时，每次只能导出一个模式中的表。

例如，导出 scott 模式下的 emp 表和 dept 表，转储文件名称为 emp_dept.dmp，日志文件命名为 emp_dept.log，作业命名为 emp_dept_job，导出操作启动 3 个进程。执行过程为：

```
C:\>expdp scott/tiger DIRECTORY=dumpdir DUMPFILE=emp_dept.dmp
   LOGFILE=emp_dept.log TABLES=emp,dept JOB_NAME=emp_dept_job PARALLEL=3
Export: Release 10.2.0.1.0 - Production on 星期五, 13 3月, 2016 22:00:18
Copyright (c) 2003, 2005, Oracle.  All rights reserved.
连接到: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
启动"SCOTT"."EMP_DEPT_JOB":scott/******DIRECTORY=dumpdir
DUMPFILE=emp_dept
.dmp LOGFILE=emp_dept.log TABLES=emp,dept JOB_NAME=emp_dept_job PARALLEL=3
正在使用 BLOCKS 方法进行估计...
处理对象类型 TABLE_EXPORT/TABLE/TABLE_DATA
使用 BLOCKS 方法的总估计: 128 KB
处理对象类型 TABLE_EXPORT/TABLE/TABLE
. . 导出了 "SCOTT"."DEPT"                              5.656 KB      4 行
处理对象类型 TABLE_EXPORT/TABLE/GRANT/OWNER_GRANT/OBJECT_GRANT
. . 导出了 "SCOTT"."EMP"                               7.843 KB     16 行
处理对象类型 TABLE_EXPORT/TABLE/INDEX/INDEX
处理对象类型 TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
处理对象类型 TABLE_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
处理对象类型 TABLE_EXPORT/TABLE/AUDIT_OBJ
处理对象类型 TABLE_EXPORT/TABLE/FGA_POLICY
处理对象类型 TABLE_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
处理对象类型 TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
已成功加载/卸载了主表 "SCOTT"."EMP_DEPT_JOB"
******************************************************************
SCOTT.EMP_DEPT_JOB 的转储文件集为:
D:\ORACLE\BACKUP\EMP_DEPT.DMP
作业 "SCOTT"."EMP_DEPT_JOB" 已于 22:02:18 成功完成
```

② 模式导出模式。模式导出模式是将一个或多个模式中的对象结构及其数据导出到转储文件中。

例如，导出 scott 模式下的所有对象及其数据。命令为：

```
C:\>expdp scott/tiger DIRECTORY=dumpdir DUMPFILE=scott.dmp
    LOGFILE=scott.log SCHEMAS=scott JOB_NAME=exp_scott_schema
```

③ 表空间导出模式。表空间导出模式是将一个或多个表空间中的所有对象结构及其数据导出到转储文件中。

例如，导出 EXAMPLE，USERS 表空间中的所有对象及其数据。命令为：

```
C:\>expdp scott/tiger DIRECTORY=dumpdir DUMPFILE=tsp.dmp
    TABLESPACES=example,users
```

④ 传输表空间导出模式。传输表空间导出模式是将一个或多个表空间中对象的定义信息导出到转储文件中。

例如，导出 EXAMPLE，USERS 表空间中数据对象的定义信息。命令为：

```
C:\>expdp scott/tiger DIRECTORY=dumpdir DUMPFILE=tts.dmp
    TRANSPORT_TABLESPACES=example,usersTRANSPORT_FULL_CHECK=Y
```

```
LOGFILE=tts.log
```

注意：当前用户不能使用传输表空间导出模式导出自己的默认表空间。

⑤ 数据库导出模式。数据库导出模式将数据库中的所有信息导出到转储文件中。

例如，将当前数据全部导出，不写日志文件。命令为：

```
C:\>expdp scott/tiger DIRECTORY=dumpdir DUMPFILE=expfull.dmp FULL=YES
    NOLOGFILE=YES
```

⑥ 按条件查询导出。按条件查询导出主要指在表模式导出中使用 QUERY 参数设置导出条件。

例如，导出 scott.emp 表中部门号大于 10，且工资大于 2000 的员工信息。命令为：

```
C:\>expdp scott/tiger DIRECTORY=dumpdir DUMPFILE=exp2.dmp TABLES=emp
    QUERY='emp:"WHERE deptno=10 AND sal>2000"' NOLOGFILE=YES
```

（2）参数文件方式导出

参数文件方式导出是指将导出参数的设置放入一个文件中，在命令行中通过 PARFILE 参数指定该参数文件。

例如，首先创建一个名为 scott.txt 的参数文件，并存放到 d:\backup 目录下，其内容为：

```
SCHEMAS=scott
DUMPFILE=filter.dmp
DIRECTORY=dumpdir
LOGFILE=filter.log
INCLUDE=TABLE:"IN ('EMP', 'DEPT')"
INCLUDE=INDEX:"LIKE 'EMP%'"
INCLUDE=PROCEDURE
```

然后在命令行中执行下列命令就可以执行数据的导出操作了。

```
C:\>expdp scott/tiger PARFILE=d:\scott.txt
```

（3）交互命令方式导出

交互命令方式导出是指在导出作业进行的过程中，用户可以通过交互式命令对当前运行的导出作业进行控制和管理。

可以通过两种方式实现对运行的导出作业进行控制与管理：

① 在当前运行作业的终端中按 Ctrl+C 组合键，进入交互式命令状态；

② 在另一个非运行导出作业的终端中，通过导出作业名称来进行导出作业的管理。

用于对导出作业进行管理的命令见表 9-2。

表 9-2　Expdp 交互命令及其功能

命 令 名 称	功　　能
ADD_FILE	向转储文件集中添加转储文件。语法为：ADD_FILE=[directory_object]file_name [,...]
CONTINUE_CLIENT	终端返回到记录模式。如果处于空闲状态，将重新启动作业
EXIT_CLIENT	退出客户机会话并使作业处于运行状态
FILESIZE	后续 ADD_FILE 命令的默认文件大小（字节）
HELP	总结交互命令
KILL_JOB	分离和删除作业
PARALLEL	更改当前作业的活动进程数目。语法为：PARALLEL=integer
START_JOB	启动/恢复当前作业
STATUS	显示作业的累积状态，以及对操作的描述。语法为：STATUS=integer
STOP_JOB	顺序关闭执行的作业并退出客户机。STOP_JOB=IMMEDIATE 将立即关闭数据泵作业

在执行下列作业的过程中，按 Ctrl+C 组合键，进入交互模式，可以输入相应命令进行作业管理。例如：

① 执行一个作业。

```
C:\>expdp scott/tiger FULL=YES DIRECTORY=dumpdir
   DUMPFILE=fulldb1.dmp,fulldb2.dmp FILESIZE=2G PARALLEL=3
   LOGFILE=expfull.log JOB_NAME=expfull
```

② 作业开始执行后，按 Ctrl+C 组合键。

③ 在交互模式中输入导出作业的管理命令，根据提示进行操作。

```
Export>STOP_JOB=IMMEDIATE
Are you sure you wish to stop this job ([Y]/N): Y
```

9.3.3 使用 Impdp 导入数据

1．Impdp 调用接口

与 Expdp 类似，Impdp 也提供了 3 种应用接口：

- 命令行接口（Command-Line Interface）；
- 参数文件接口（Parameter File Interface）；
- 交互式命令接口（Interactive-Command Interface）。

2．Impdp 导入模式

与 Expdp 导出模式相对应，Impdp 导入模式也分为 5 种：

- 全库导入模式（Full Import Mode）；
- 模式导入模式（Schema Mode）；
- 表导入模式（Table Mode）；
- 表空间导入模式（Tablespace Mode）；
- 传输表空间导入模式（Transportable Tablespace）。

3．Impdp 帮助及参数

在操作系统的命令提示符窗口中输入 impdp HELP=Y 命令，可以查看 Impdp 程序的使用、关键字（参数）、交互命令等介绍。

在使用 Impdp 向数据库中导入数据时，需要通过参数进行数据导入的设置。Impdp 常用的命令参数及其说明见表 9-3。

表 9-3　Impdp 命令参数及其说明

参 数 名 称	说　　明
ATTACH	把导入结果附加在一个已存在的导入作业中，默认为当前模式的唯一的导入作业。语法为： ATTACH [=[schema_name.]job_name]
CONTENT	指定要导入的内容。语法为：CONTENT=[ALL | DATA_ONLY | METADATA_ONLY]。ALL 表示导入对象的定义及其数据；DATA_ONLY 表示只导入对象的数据；METADATA_ONLY 表示只导入对象的定义。默认为 ALL
DIRECTORY	指定转储文件和日志文件所在位置的目录对象，该对象由 DBA 预先创建。语法为： DIRECTORY=directory_object
DUMPFILE	指定转储文件名称列表，可以包含目录对象名，默认值为 expdat.dmp。语法为： DUMPFILE=[directory_object:]file_name [, ...]
ESTIMATE	用于指定估计网络导入操作时生成数据量多少的方法。语法为： ESTIMATE=[BLOCKS | STATISTICS]。默认值为 BLOCKS

续表

参 数 名 称	说　明
EXCLUDE	指定导入操作中要排除的对象类型和对象定义。语法为： EXCLUDE= object_type[:name_clause] [, ...]
FLASHBACK_SCN	指定导入操作时，允许数据库闪回到特定的 SCN。语法为 FLASHBACK_SCN=scn_value
FLASHBACK_TIME	指定导入操作时，用于获取最接近指定时间的 SCN，然后进行闪回。语法为： FLASHBACK_TIME="TO_TIMESTAMP(time-value)"
FULL	指定是否进行全数据库导入，包括所有数据及定义。语法为：FULL=[YES | NO]。默认值为 NO
HELP	指定是否显示 Impdp 命令的在线帮助。语法为：HELP = [YES| NO]。默认值为 NO
INCLUDE	指定导入操作中要导入的对象类型和对象定义。语法为：INCLUDE= object_type[:name_clause] [, ...]
JOB_NAME	指定导入作业的名称。语法为 JOB_NAME=jobname_string。默认值为系统自动为作业生成一个名称
LOGFILE	指定导入日志文件的名称。语法为：LOGFILE=[directory_object:]file_name。默认值为 import.log
NETWORK_LINK	指定网络导入时的数据库链接名称。语法为：NETWORK_LINK=source_database_link
NOLOGFILE	指定是否生成导入日志文件。语法为：NOLOGFILE=[YES|NO]。默认值为 NO
PARALLEL	指定执行导入作业时的并行进程最大个数。语法为：PARALLEL=integer。默认值为 1
PARFILE	指定导入参数文件的名称。语法为：PARFILE=[directory_path]file_name
QUERY	指定导入操作中 SELECT 语句中的数据导入条件。语法为： QUERY = [schema.][table_name:] query_clause
REMAP_DATAFILE	将源数据文件名转换为目标数据文件名。语法为：REMAP_DATAFILE=source_datafile:target_datafile
REMAP_SCHEMA	将源模式中的所有对象导入目标模式中。语法为：REMAP_SCHEMA=source_schema:target_schema
REMAP_ TABLESPACE	将源表空间所有对象导入目标表空间中。语法为： REMAP_TABLESPACE=source_tablespace:target_tablespace
REUSE_DATAFILES	指定是否使用创建表空间时已经存在的数据文件。语法为：REUSE_DATAFILES=[Y | N]
SCHEMAS	指定进行模式导入及模式列表。语法为：SCHEMAS=schema_name [, ...]。默认为当前模式
SKIP_UNUSABLE_ INDEXES	指定导入操作时是否跳过不可使用的索引。语法为：SKIP_UNUSABLE_INDEXES=[Y | N]
SQLFILE	指定将导入操作中要执行的 DDL 语句写入一个 SQL 脚本文件中。语法为： SQLFILE=[directory_object:]file_name
STATUS	指定显示导入作业状态的时间间隔。语法为 STATUS=[integer]。默认值为 0，表示操作结束时显示
STREAMS_ CONFIGURATION	指定是否导入转储文件中生成的流元数据。语法为：STREAMS_CONFIGURATION=[Y | N]
TABLE_EXISTS_ ACTION	指定导入过程中要创建的表已经存在时该如何操作。语法为：TABLE_EXISTS_ACTION=[SKIP | APPEND | TRUNCATE | REPLACE]。默认值为 SKIP
TABLES	指定表模式导入及表列表。语法为：TABLES=[schema_name.]table_name[:partition_name] [, ...]
TABLESPACES	指定进行表空间模式导入及表空间列表。语法为：TABLESPACES=tablespace_name [, ...]
TRANSFORM	指定是否修改创建对象的 DDL 语句。语法为：TRANSFORM = transform_name:value[:object_type]
TRANSPORT_ DATAFILES	指定在传输表空间导入模式中导入目标数据库的数据文件列表。语法为： TRANSPORT_DATAFILES=datafile_name
TRANSPORT_ FULL_CHECK	用于在传输表空间模式导入中指定是否进行导入表空间中的对象与非导入表空间对象间依赖关系的检查。语法为：TRANSPORT_FULL_CHECK=[YES|NO]。默认值为 NO
TRANSPORT_ TABLESPACES	指定进行传输表空间模式导入及表空间列表。语法为： TRANSPORT_TABLESPACES=tablespace_name [, ...]
VERSION	指定被导入的数据库对象的版本。语法为： VERSION=[COMPATIBLE | LATEST | version_string]。默认值为 COMPATIBLE

4．Impdp 应用实例

（1）命令行方式导入

① 表导入模式。如果 scott 模式下的 emp 表和 dept 表中数据丢失，可以使用逻辑备份文件 emp_dept.dmp 进行恢复。如果表结构存在，则只需要导入数据。可以按下列命令进行：

```
C:\>impdp scott/tiger DIRECTORY=dumpdir DUMPFILE=emp_dept.dmp
    TABLES=emp,dept NOLOGFILE=YES CONTENT=DATA_ONLY
Import: Release 10.2.0.1.0 - Production on 星期六, 14 3月, 2016 14:37:38
Copyright (c) 2003, 2005, Oracle.  All rights reserved.
连接到: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
已成功加载/卸载了主表 "SCOTT"."SYS_IMPORT_TABLE_01"
启动 "SCOTT"."SYS_IMPORT_TABLE_01":  scott/******** DIRECTORY=dumpdir
DUMPFILE=emp_dept.dmp TABLES=emp,dept NOLOGFILE=Y CONTENT=DATA_ONLY
处理对象类型 TABLE_EXPORT/TABLE/TABLE_DATA
. . 导入了 "SCOTT"."DEPT"                              5.656 KB       4 行
. . 导入了 "SCOTT"."EMP"                               7.820 KB      14 行
作业 "SCOTT"."SYS_IMPORT_TABLE_01" 已于 14:37:44 成功完成
```

如果表结构也不存在了，则应该导入表的定义以及数据。命令为：

```
C:\>impdp scott/tiger DIRECTORY=dumpdir DUMPFILE=emp_dept.dmp
    TABLES=emp,dept NOLOGFILE=Y
```

② 模式导入模式。如果模式所有数据丢失，可以使用该模式的备份进行恢复。例如，如果 scott 模式中所有信息都丢失了，可以使用备份文件 scott.dmp 进行恢复，命令为：

```
C:\>impdp scott/tiger DIRECTORY=dumpdir DUMPFILE=scott.dmp
    SCHEMAS=scott JOB_NAME=imp_scott_schema
```

如果要将一个备份模式的所有对象导入另一个模式中，可以使用 REMAP_SCHEMAN 参数设置。例如，将备份的 scott 模式对象导入 oe 模式中，命令为：

```
C:\>impdp scott/tiger DIRECTORY=dumpdir DUMPFILE=scott.dmp
    LOGFILE=scott.log REMAP_SCHEMAN=scott:oe  JOB_NAME=imp_oe_schema
```

③ 表空间导入模式。如果一个表空间的所有对象及数据都丢失了，可以使用该表空间的逻辑备份进行恢复。例如，利用 EXAMPLE，USERS 表空间的逻辑备份 tsp.dmp 恢复 USERS，EXAMPLE 表空间，命令为：

```
C:\>impdp scott/tiger DIRECTORY=dumpdir DUMPFILE=tsp.dmp
    TABLESPACES=example,users
```

如果要将备份的表空间导入另一个表空间中，可以使用 REMAP_TABLESPACE 参数设置。例如，将 USERS 表空间的逻辑备份导入 IMP_TBS 表空间，命令为：

```
C:\>impdp scott/tiger DIRECTORY=dumpdir DUMPFILE=tsp.dmp
    REMAP_TABLESPACE=users:imptbs
```

④ 传输表空间导入模式。将表空间 USERS 导入数据库链接 source_dblink 所对应的远程数据库中。

```
C:\>impdp scott/tiger DIRECTORY=dumpdir NETWORK_LINK=source_dblink
    TRANSPORT_TABLESPACES=users TRANSPORT_FULL_CHECK=NO
    RANSPORT_DATAFILES='D:\ORACLE\USERS01.DBF'
```

⑤ 数据库导入模式，可以利用完整数据库的逻辑备份恢复数据库。例如：

```
C:\>impdp scott/tiger DIRECTORY=dumpdir DUMPFILE=expfull.dmp FULL=Y
    NOLOGFILE=Y
```

⑥ 按条件查询导入，可以对导入的数据进行选择过滤。例如：

```
C:\>impdp scott/tiger DIRECTORY=dumpdir DUMPFILE=emp_dept.dmp
    TABLES=emp,dept QUERY='emp: "WHERE deptno=20 AND sal>2000"'
    NOLOGFILE=YES
```