系列读物之二

# ORACLE关系数据库管理系统

# 用 户 手 册

（四）

# 实用程序用户指南

《计 算 机 技 术》编 辑 部

# 前　言

**目的**

本指南描述ORACLE RDBMS的两种实用程序。

ODL ORACLE数据装入程序(Data Loader)是从现有操作系统数据文件向ORACLE表装入数据的实用程序。

Export/Import（卸出/重装）实用程序用于从ORACLE表向操作系统文件卸出数据，并将那些文件中的数据重新装入ORACLE数据库。

内容包括输入文件的要求和输出文件的描述，控制哪些数据被装入、卸出或重装。所建立或使用的实例与Export/Import和ODL会话的实例一起列出。

**读者对象**

本指南是为下列用户编写的：

· 想从操作系统文件（使用ODL）把数据装入到ORACLE数据库中。

· 从ORACLE数据库或向ORACLE数据库传送数据（用Export和Import）。

用户必须具有基本的使用SQL语言的工作经验和ORACLE数据库的基础知识。此外，ODL还要求用户能够建立和编辑向ODL输入的操作系统文件和数据文件。

**本手册是怎样组成的**

这本手册包括两大部分：

1.ORACLE数据装入程序（ODL）

2.卸出/重装（Export/Import）

附录部分包括ODL可能产生的错误信息。

**有关的出版物**

用这本指南时，可以参考下列由ORACLE Corporation出版的资料。

· ORACLE RDBMS Release Notes ORACLE Part No.3001

· ORACLE Database Administrator's Guide ORACLE Part No.3601

· Error Messages and Codes ORACLE part No.3605

# 目　录

**图的清单**

# 第一章　ORACLE数据装入程序(ODL)

ORACLE数据装入程序（也称ODL）是用来从操作系统文件把原始数据装入ORACLE数据库表中的ORACLE实用程序。调用ODL时，此表必须存在。ODL不建立表。

### ODL的特性

ODL的用户可以：

- 从文件向ORACLE表中装入全部数据
- 从数据文件选择性地装入部分数据
- 在表中有选择地装入某些列（包括所有已被说明为NOT NULL）的列
- 使用一个函数为每个记录生成唯一的整数键
- 装入指定数的记录
- 在开始装入之前跳过指定数的记录
- 在达到指定错误数的情况下指出是停止还是继续

### ODL过程的概述

为了调用ODL，只要输入带有某些自变量的ODL命令。所有ODL所需要的信息在命令行上和在命令行上指定的文件中加以说明。图1为ODL过程的原理图。

输入时，你要准备两个或更多的文件：

控制文件，这一文件包括执行装入的一般说明。指出原始数据文件以及ORACLE表和被装入的列，并为相应的表列安排文件的数据字段。

原始数据文件　这一文件包含要装入的数据。通过一个控制文件可以装入一个或更多的数据文件。数据文件在控制文件中命名。

用ODL命令调用时，ODL按照控制文件中的指令，读原始数据文件，使用SQL INSERT语句把数据装入已命名的ORACLE表。

ODL运行时，为跟踪其过程，它建立一个或两个文件：

登录文件　ODL总是写一个登录文件，该文件包括被读出、装入或被拒绝的记录数和

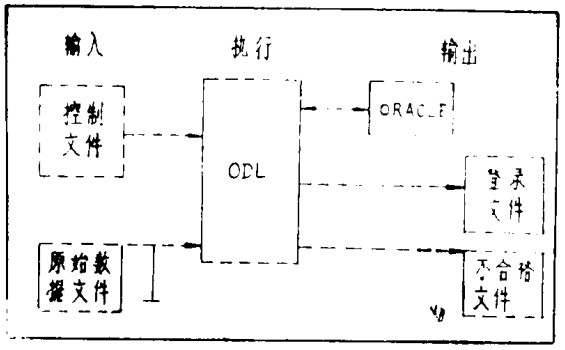


图1　ODL的操作原理

错误信息。

不合格文件　凡是遇到不合格记录，ODL就建立一个不合格文件。例如，如果ODL遇到了一个记录带有对列无效的数据（比如要向数字列内装入字符数据），ODL就把该记录写入这不合格文件，并继续装入下一个记录。为了改正错误数据，可以稍后一些对不合格文件进行编辑，并重新进行ORACLE数据装入。

### ODL不能做什么

ODL不能：

- 执行一次控制文件向一个以上的表装入
- 装入记录，这些记录跨越数据文件的一个以上的物理记录

例如，下列原始数据文件的记录格式可以通过ODL装入：

```
记录 1
记录 2
记录 3
……
```

而下面的记录格式则不能装入：

```
记录 1
…记录 1 续
记录 2
…记录 2 续
记录 3
…记录 3 续
…
```

### ODL的必要条件

数据库中应预先存在要装入的表，不同于Import（重装入），ODL不能为用户建立表。

运行ODL的用户仅仅需具有CONNECT特权访问数据库(不要求有RESOURCE特权)。用户不需要是装入表的建立者（所有者）；不过，用户必须要被允许INSERT访问表。

用户必须提供控制文件和数据文件。控制文件在ODL命令中命名。调用ODL时，两个文件都必须是可读的。

ODL要求，为装入而读出的第一个记录是有效的记录；如果该记录无效，ODL将终止。

### ODL命令

当你有了适当的输入文件，并且在ORACLE数据库中建立了要装入的表之后，即可调用ODL。用ODL命令的选择项，可以指定：

- 在开始装入之前，跳过X个记录（选择项S）
- 每次提交X个记录（选择项C）
- 仅仅装入X个记录（选择项L）

·处理所遇到的任何错误（选择项E）

·遇到X个错误之后，停止处理（选择项E）

这些选择项是在运行时指定的，并不是在控制文件中指定。控制文件只指定某些与数据装入有关的情况，其它情况在命令被真正输入时加以指定。

ODL命令语法如图2所示，命令的选择项描述如下。

```
ODL control-file log-file[uid][-options]
```

**图2 ODL命令语法**

必须在命令行上说明控制文件和登录文件的名字。

控制文件　对控制文件命名，控制文件这里包含有原始数据文件的名字。因此不能在命令行中指定任何数据文件。

登录文件　对记录错误信息和装入处理统计的文件命名。

uid　是被装入表的所有人或表的同义词的所有人的ORACLE用户名/口令。如果未输入uid，ODL将为此发出提示。

ODL选择项可以用任何一种组合加以指定。每一个选择项都采用整数n作为自变量。

S　在开始装入之前跳过几个输入记录，缺省值是不跳过任何记录。如果多个数据文件被装入，被跳过的记录仅在第一个文件中。

L　装入n个记录（在跳过由S指出的任何记录之后）。如果没有指定L，缺省值是指一直装入到文件结束。

n>0　读n个记录供输入

n=0　一直装入到文件结束（和缺省值一样）

C　每装入n个记录之后提交一次。缺省值表示在二个提交之间要装入100个记录。

n>0　在n个成功的插入之后提交

n=0　只有在表被完全装入之后提交

E　允许出n个错误，错误多于n个ODL应停止处理。按缺省值，50个错误就将导致ODL停止装入数据。

n>0　停止之前允许n个错误

n=0　装入数据，不管有多少错误

B　为赋值（bind）数组和输入缓冲区所使用的字节数。缺省值是16K，通常情况下是足够用的，在装入特别大的表时，你可能希望加大此值。这里不置最大值；最大值与可用的内存有关。

**举例**

下面是调用ODL的两个例子。第一个例子将读JUNECONT控制文件，而JUNELOG文件将包含有关实际装入的信息。在装入数据文件中的剩余记录之前，ODL将跳过1000个记录。如果遇到20个错误，ODL就终止运行。每100个记录就执行一次提交。

```
ODL junecont、ctl junelog.log shir ley/bear-S1000-E20
```

下面的ODL的调用将提交每一个记录，并装入整个数据文件，而不管遇到多少个错误。

```
ODL sample.ctl logsamp.log scott/tiger-c1-E0
```

**控制文件**

ODL要求至少有两个文件在运行：控制文件和一个或几个原始数据文件。控制文件定义ODL将为特定的会话做些什么。在ODL命令中你命名控制文件，而控制文件命名数据文件。

对于文件的扩充或控制文件的类型没有缺省值；因此你应该在ODL命令中指定完整的文件名。

控制文件必须包括三部分：

1.DEFINE RECORD 语句

2.DEFINE SOURCE 语句

3.FOR EACH 语句

图3示出一个ODL控制文件的实例，该文件定义一个被称之为EMPLOYEE的记录，其中包括雇员的名字（ENAME），级别（EGRADE）和工资（ESALARY）这三个字段。

```
DEFINE RECORD employee AS
    ename   (char (20) ),
    egrade  (integer(2) ),
    esalary (float(4), loc( +10) );
DEFINE SOURCE TAPE
     FROM file1, file2
     LENGTH 80
     CONTAINING employee ;
FOR EACH RECORD
     INSERT INTO empmain
     ( name, performance, grade, gross,
        nulcol, salary, start) VALUES
     (ename, 87.6543, egrade, .86e - 5,
        null, esalary, 356 )
  NEXT RECORD
```

**图3 ODL控制文件举例**

ENAME字段开始于记录起始处，长达20个字节。下两个节是EGRADE字段。ESALARY字段位于EGRADE右边第10个字节。称之为TAPE的输入源有两个文件，每个文件有一个长度为80字节的记录。用来自EMPLOYEE原始数据记录或规定常数装入数据库表EMPMAIN中的7个列。

**DEFINE RECORD语句**

DEFINE RECORD语句描述在原始数据文件中碰到的数据。它：

- 命名将被装入的记录
- 命名将被装入的记录的字段
- 把数据类型与每个字段联系起来
- 在记录中指定各个字段的位置

图4所示为DEFINE RECORD语句的一般语法

```
DEFINE RECORD rec-name AS fld-name(fld-type[, LOC(fld-loc)])
                          fld-name(fld-type[, LOC(fld-loc)]);
```

**图4 DEFINE RECORD语句的语法**

变量的描述如下：

rec-name　是对原始数据记录进行引用所使用的名字

fld-name　是对原始数据字段进行引用所使用的名字

fld-type　按照对应于浮点，二进制整数，和字符
{FLOAT|INTEGER|CHAR}[(size)]之一，定义数据类型。

括在括弧中以字节为单位的尺寸是任选的，并且对尺寸的要求不同于缺省。容许的尺寸和缺省值是：

| | | MACHINE ——DEFAULTS—— | | | |
|---|---|---|---|---|---|
| Datatype | Sizes | RSX | VAX | IBM | |
| Datatype | Sizes | 11-M | VMS | MVS,CMS | UNIX |
| FLOAT | 8, 4 | 8 | 8 | 8 | 可变的 |
| INTEGER | 4,2,1 | 2 | 4 | 4 | 可变的 |
| CHAR | 1-240 | 1 | 1 | 1 | 1 |

fld-Loc　是原始数据记录中字段的位置

```
          | +number
Loc  (    | -number  )
          |  number
```

需要注意的是，每个记录的第一个位置是0号列，而不是1。不带任何符号的整数是从记录开始的绝对位置。数字前面的符号表示位置是相对于前一字段的结束位置（负号表示向记录开始方向，正号表示向记录结束方向。缺省位置是在前一字段的结尾或者第一个字段的记录的开始。

如果数据库列被定义为NUMBER类型，原始数据字段是一个表示数字量的ASCII字符串，那么数据类型应该是CHAR，ORACLE负责转换成数字格式。

**举例**

```
DEFINE RECORD REC1 AS
  FLD1 (CHAR(6)),
  FLD2 (CHAR(25)),
  FLD3 (FLOAT, LOC(-3));
```

**DEFINE SOURCE语句**

DEFINE SOURCE语句指出输入介质（如文件或磁带）和包括在这一资源中的记录。该语句的语法如图5所示。

```
DEFINE SOURCE    src-name
        FROM     filename[, filename..., filename]
        LENGTH   length
        CONTAINING rec-name;
```

**图5 DEFINE SOURCE语句的语法**

有关自变量的描述如下：

src—name 是用于资源的符号名；可以是任意一个标识符，对于这个参数，一般可使用FILE或TAPE。

filename 是包含原始数据的操作系统文件的文件名，由逗号分开的多个文件名可用于需要输入的连在一起的文件。

length 是原始数据记录长度。所指定的记录长度应是大记录长度。如果任何记录超过了记录长度，ODL将发出下面的信息：

```
Read error, Skipping to next file
```

并继续向下执行。只要最后一个列是一个长度可变的列或可变数据项之后的所有列都是null，就可以正确处理可变长度记录。逻辑记录长度不是根据文件属性确定，而是与物理记录长度相同。例如，RSX—11M和VAX/VMS允许文件有一个隐含的回车。

如果源文件是带有固定长度记录的可变长度记录文件，那么就不应该包含一个LOC，跳过记录前字节计数。当计算字段位置时，ODL识别字节记数字段并使用实际逻辑记录长度。

rec-na 是在DEFINE RECORD语句中使用的记录名。

**举例**

```
DEFINE SOURCE SRC1
    FROM FILE1, FILE, FILE3
    LENGTH 80
    CONTAINING REC1;
```

**FOR EACH语句**

FOR EACH语句读每一个原始数据记录并将其插入指定的表中。该语句的语法如图6所示。

```
FOR EACH RECORD INSERT INTO table-name(col-name, col-name...)
VALUES(fld-name, fld-name...)
NEXT RECORD
```

**图6 FOR EACH语句的语法**

为每一个记录都需要执行标准的SQL INSERT语句。EACH语句是一个简单的标准SQL插入语句，其格式如下：

```
INSERT INTO tab-name
(COL-name,...,COL-name)VALUES(src-data,...)
```

这儿的tab-name是用户拥有的表，或者是用户为其它人的表所建立了的同义词，表名

SCOTT、DEPT

的下述格式：

不能被装入。为装入其它人的表，你可以使用表的同义词，对该同义词你已被赋于INSERT访问。你还必须指定需要插入的列的清单，并且使用关键词VALUES。然后指定由逗号分开的原始数据字段名或数据值的清单。你可以指定下列任何一个当作要被插入的数据：

- 原始数据文件中一个字段的名字
- 字符串
- NUMBER（不是数字串就是科学符号）
- NULL
- GENSEQ函数（为每一个记录产生新的整数的函数）。其格式为：

GENSEQ (initial-value, increment-value)

例如：

```
FOR EACH RECORD
INSERT INTO CODE-TABLE(GENKEY, DESCRIPTION)
                VALUES(GENSEQ(100,5), PURPOSE)
NEXT RECORD
```

这里，用PURPOSE字段中的值加载表CODE-TABLE中的列DESCRIPTION，并赋于列GENKEY在加载时生成的唯一的值，第一个值将是100，而且每个值将增加5；这样GENKEY的第一个值将是100，105，110，115等等。

为了结束插入循环，要求用NEXT RECORD短语。

**举例**

```
FOR EACH RECORD
INSERT INTO EMP(EMPNO, EMPNAME, SALARY, COMM)
                VALUES(FLD1, FLD2, FLD3, NULL)
NEXT RECORD
```

**原始数据文件**

一个ODL控制文件可以引用一个或多个数据文件，只要以下内容是真的，数据文件可以是由操作系统定义的任何记录的格式来表述：

- 记录可以是（固定长或可变长度）。
- 文件中的每个物理记录对应于一个新的逻辑记录。
- ODL在数据文件中读出的第一个记录必须是有效的，否则ODL就不再继续。如果你指出，在开始加载之前ODL应该跳过100个记录，那么第101个记录必须是有效的。
- 记录中，除最后一个字段外，每一个字段必须是固定长度的。
- 最后一个字段可以是可变长度的，如果是可变长度，那么在开始向表中插入之前，用空格填充数据值。
- 不是所有的源记录字段都必须装入，任何列都可以跳过，这由控制文件确定。
- 源记录的数据列不需要与表的列对应；然而，它们必须与控制文件中的DEFINE

RECORD和FOR EACH…INSERT语句相对应。

·所支持的数据类型：

——字符串

——二进制整数

——浮点数

ODL当前尚不接受压缩十进制数。

有关原始数据文件的其它信息，请见与你的特定的操作系统相适应的“安装和用户指南。”

**登录文件（LOG FILE）**

ODL执行一个装入时，写一个登录文件，在该登录文件中记录了在装入时统计被加载的记录个数，以及装入时产生的错误信息。调用ODL时，指定登录文件的名字；这里没有缺省文件扩展或文件类型，因此，最好是明确地指出一个。（不使用文件扩展或BAD文件类型。因此ODL使用带有扩展的登录文件名字或BAD文件类型来自动地命名不合格文件。）

记录文件中的信息之一指出多少个来自记录缓冲区的记录被插入；如果希望调整缓冲区尺寸，以每次插入持有较多或较少的行，在ODL每运行一次之后，可以使用这一反馈。由于调整缓冲区而持有了较多的行，装入将进行得更快些。

图7为登录文件的实例。

```
ORACLE DATA LOADER: Version 5.0.12 on Wed Sep 25 11:35:03 1985

Copyright (c) 1985, Oracle Corporation, California, USA.  All rights
                                                          reserved.

LOGGED INTO ORACLE V5.0.12 - Beta
          16384 TOTAL BIND SPACE
            654 MAXIMUM BIND ARRAY DIMENSION
          16368 MAXIMUM BIND ARRAY SIZE IN BYTES
            654 ACTUAL BIND ARRAY DIMENSION LIMITED BY MEMORY
          16368 ACTUAL BIND ARRAY SIZE IN BYTES
LOGGED OUT FROM ORACLE
STATISTICS
            574 BYTES ALLOCATED
              0 RECORDS SKIPPED
            254 RECORDS READ
              0 RECORDS REJECTED
            254 ROWS LOADED
              0 ERRORS
END ORACLE DATA LOADER  Wed Sep 25 11:35:13 1985
```

**图7 登录文件实例**

如果ODL在第一个记录中遇到一个错误，执行就立即停止，也不会有记录被装入。一个如下所示的信息将出现在登录文件中：

```
EXEC ERROR  ORA-1438: Value Larger than specified
                      precision allows for this column
```

RECORD 1 REJECTED
LOGGED OUT FROM ORACLE

**不合格文件(BAD FILE)**

如果ODL在原始数据文件中遇到任何不能成功地插入的记录，就建立一个不合格文件来保存被拒绝的记录。当一个输入的记录与控制文件中的定义不一致时，输入的记录就被拒绝。可以编辑一个不合格文件去改正错误，然后用同一个ODL控制文件重新装入它。不合格文件的名字自动地带有扩充或文件类型BAD的登录文件的名字。因而，如果遇到任何不合格记录，下面的ODL调用将建立一个被称之为IM.BAD的不合格文件。

ODL SURVEY CTL IM.LOG RODNEY/HERRING

**ODL信息**

为登录文件所写的ODL处理信息包括以下的统计信息：

number of records read (被读的记录数)
number of records skipped (被跳过的记录数)
number of records loaded (被装入的记录数)
number of records rejected (被拒绝的记录数)

当处理记录发生错误时，就会产生以下的信息（这里的#是被读记录的个数）：

RECORD(#)REJECTED

**ODL提示**

加载DATE数据

为了加载DATE数据，首先向在一个临时表中定义为CHAR的列中加载数据。然后用一个带有嵌套选择的INSERT语句向一个永久表中插入行(或者一个带有嵌套选择的CREATE TABLE)，用TO DATE功能将字符——日期转换成标准的ORACLE DATE数据格式。

范围或值的检查

为把数据的有效性，诸如范围一检查，强加于被加载数据，首先向临时表中加载数据。然后用带有嵌套选择的INSERT语句（或带有嵌套选择的CREATE TABLE）将数据装入永久表中。嵌套选择的WHERE子句应该指定数据检查，如：

```
…WHERE col 1 in('A', 'B', 'C') and
        col 2 BETWEEN 1 AND 10…
```

# 第二章　卸出和重装入

卸出（Export）实用程序提供了一种将数据库中的数据复制成备用文件的手段。随同使用的实用程序Import将卸出的数据重新贮存到一个ORACLE数据库中。

## 卸出／重装入的特征

卸出和重装入可以用于：

- 在与任何ORACLE数据库无关的操作系统文件中存储ORACLE数据
- 存储以下的ORACLE数据
  - —表的定义
  - —表数据
  - —授权
  - —同义词
  - —视图定义
  - —空间定义
  - —索引
- 释放数据库中的空间
- 存储较老的或临时的数据
- 在ORACLE数据库之间传送数据
- 在ASCII和EBCDIC之间转换数据
- 从较老的ORACLE数据库向新的版本传送数据
- 从一个拥有者向另一个传送数据

## 卸出／重装入不能做什么

大多数ORACLE用户只能卸出他们自己拥有的目标；DBA是唯一能够卸出或重装入属于任一用户的表或数据的用户。

因为卸出文件是一种特殊的压缩格式，由卸出建立的文件不能被编辑，也不能由其它的ORACLE实用程序或诸如ODL这样的产品或编程接口来使用。

## 错误处理

在遇到错误之后，卸出和重装入企图继续进行。如果在重装入插入一行时，出现错误，不合格行被标上记号，重装入表的剩余部分；对卸出完全是一样的。如果正当重装入或卸出单个表时出现致命的错误，则在下一个表上继续进行处理。如果出现DDL错误，处理继续进行。

## 卸出

卸出用于从ORACLE数据库向操作系统文件写数据，以便以后用重装入向ORACLE数据库存储数据或恢复据据。卸出是一个使用方便的交互操作的实用程序；它只提少数的几个

问题，并根据回答进行处理。

**卸出的方式**

第一个提示要求回答。你想使用哪一种卸出方式。你可以根据自己所选择的方式卸出信息。大多数用户有两种选择；DBA也可以选择全数据库方式：

- 用户方式
- 表方式
- 全数据库方式

图 8 概括三种卸出方式中的每一种什么能够卸出和什么不能卸出的情况。

| 方　式： | 用户 | 表 | 全数据库 |
|---|---|---|---|
| 可卸出 | 表定义 | 表定义 | 表定义 |
| | 表数据 | 表数据 | 表数据 |
| | 群 (clusters) | 空间定义 | 群 (clusters) |
| | 索引 | | 索引 |
| | 第一级授权 | | 授权 |
| | 空间定义 | | 视图 |
| | | | 同义词 |
| | | | 空间定义 |
| 不可卸出 | 视图 | 视图 | 属于SYS的目标 |
| | 同义词 | 同义词 | |
| | | 授权 | |
| | | 索引 | |
| | | 群 (clusters) | |

**图 8 卸出方式概述**

**谁可以卸出**

ORACLE用户必须对ORACLE系统具有RESOURCE或DBA特权，以便能使用卸出。仅仅具有CONNECT特权的用户不能从数据库中卸出数据。

执行卸出的用户必须确信他有足够的磁盘空间用来写卸出文件。

**卸出命令**

卸出命令开始简短的交互式会话，会话中，在处理卸出之前，Export 向用户提出几个需要回答的问题。卸出语法的两种格式如图 9 所示。

```
EXP
EXP [username/password]
```

**图 9　EXP命令语法**

在输入EXP 命令之后，输入继续给出几个提示。一般情况下，提示指出缺省回答；如果缺省是可以接收的，只要按ENTER或回车键。在任何时候，用户都可输入句号（period）—RETURN以示输入结束。

Enter array fetch buffer size (default is 4096) >这一提示确定缓冲区行可使用的尺寸。卸出使用数组取数程序接口调用以加速执行。通常情况下，缺省值是能满足要求的。如果输入的是零，每次只能取出一行。

Export file name这是由卸出建立的输出文件的名字。如果你仅输入一个文件名，卸出文件的文件扩展或文件类型将是DMP。

Mode of export你的选择取决于你是否有DBA特权。如果你有，你可以从三种方式中选择，如果没有，就必须在两种中选择。

此后再出现的问题取决于你所选择的方式，可能提示有：

Export grants ?如果你回答 yes，被卸出的授权取决于你是否用全数据库方式或用户方式。

Export the rows ?如果你回答 yes，那么，表中的数据与表定义一起被输出。如果你回答No，那么，仅仅表定义被输出。

compress extents (Y／N)：Y>按照缺省值，或者你指定 yes 来回答提示，那么，这次会话中被卸出的每个表的数据将被压缩成一个卸入时的初始长度。如果对这一提示回答No，那么，将用上次建表时起作用的空间定义来建立所有的表。

User to be exported：如果这种提示出现，当卸出者可以输入下一个ORACLE 用户的名字，这下一个用户的目标应该被卸出。为了指出“没有用户”（并终止当前卸出会话），卸出者应该输入一个句号“.”。

Table name 如果这一提示出现，输出正在请求需要卸出的下一个表的名字。表名可以带着或不带它们的拥有者的名字的词头输入；如果不带词头输入，就假定是卸出者或者是最后被卸出表的拥有者在对话。例如，如果用户A是DBA并且正在用表方式输出，那么，所有的表都在用户A的上下文中被解释，直到另一个明确的用户名被指定。

当卸出在安装时被编译，卸出模块指出是否在ASCII或EBCDIC机器上进行。按照常规自动地写所有的卸出文件。当Import企图重装入一个卸出文件时，就辨别是否需要转译，如果是，就执行之。

**用全数据库方式卸出**

仅仅具有DBA 特权用户能用全数据库方式卸出。如果向磁盘上写卸出文件，要确保有足够的磁盘空间来装文件。

**全数据方式卸出的实例**

在本例中，整个数据库带有全部授权和全部数据被卸出到文件EXPDAT．DMP上。

```
EXP SYSTEM/MANAGER
Export: Version 5.0.13 on Wed Oct  9 12:10:32 1985

Copyright  (c) 1985, Oracle Corporation, California, USA.  All rights
                                                            reserved.

Enter array fetch buffer size(default is 4096)>    (RETURN)
Export file: EXPDAT.DMP >   dba.dmp
E(ntire database), U(sers), T(ables): U > e
Export Grants (Y/N): N > y
Export the rows (Y/N): Y >  y
Compress extents (Y/N): Y> y
Exporting the entire data base.
. Exporting user definitions.
. Exporting all space definitions.
. Exporting all clusters.
. Exporting user SYSTEM
. Exporting table AFIBREAKS              10 Rows exported
         [... and so on through remaining tables]
```

**用用户方式卸出**

你可以用用户方式卸出什么和不能卸出什么，请见图 8 。如果 DBA 正在用用户方式卸出，他就可以为多个用户卸出数据。正常的用户只能卸出他自己拥有的目标。

**用户方式卸出的实例**

该例表示用户JOE用用户方式全部卸出他的表。

```
EXP JOEUSER/JOEPASS
Export: Version 5.0.13 on Tue Oct  8 11:12:13 1985

Copyright  (c) 1985, Oracle Corporation, California, USA.  All rights
                                                            reserved.

Enter array fetch buffer size(default is 4096)>    (RETURN)
Export file: EXPDAT.DMP >   joe.dmp
U(sers), or T(ables): U > u
Export Grants (Y/N): N > y
Export the rows (Y/N): Y >  y
Compress extents (Y/N): Y> y
Exporting JOE
. Exporting user JOE

. Exporting table PEOPLE          100 Rows exported
. Exporting table THINGS          666 Rows exported
  ... and so on through JOE's remaining tables
```