

第31章 SQL*Loader

本章要点：

运行SQL*Loader

SQL*Loader组件

SQL*Loader示例一览

常规路径装载与直接路径装载

31.1 运行SQL*Loader

今天，数据库在复杂程度和大小上都不断增加，G字节大小的数据库已经非常普通，并且数据仓库通常达到上T字节的存储容量。随着这些数据库的增长，使用外部数据快速有效地填充数据库的需求变得非常重要。为迎接这个挑战，Oracle提供了一个称为SQL*Loader的工具，它可以将数据从外部数据文件装载进一个Oracle数据库。

SQL*Loader具有许多功能，包括以下能力：

可以从不同文件类型的多个输入数据文件中装载数据。

输入记录可以是定长的或变长的记录。

可以在同一次运行中加载多个表，还可以逻辑地将选定的记录载入到每个不同的表中。

在输入数据载入表之前，可以对其使用SQL函数。

多个物理记录可以被编译成一个逻辑记录，同样，SQL可以提取一条物理记录并把它作为多个逻辑记录加载。

支持嵌套列、嵌套表、VARRAYS和LOBS（巨型对象，包括BLOB、CLOB、NCLOB和BFILES）。

可以通过在命令行输入sqlload、sqlldr或sqlldr80激活SQL*Loader。确切的命令根据操作系统（OS）的不同可能会有所区别。参阅Oracle操作系统说明手册获取精确的语法使用说明。请注意，本章中的全部程序清单和服务响应可能会不同于你的结果，这完全取决于正在使用的操作系统。Sqlldr命令接受数个命令行参数。不使用任何参数激活SQL*Loader，将会显示全部可用合法参数的帮助信息（参见清单31-1）。

清单31-1 SQL*Loader帮助信息

```
Invoking SQL*Loader without parameters:  
$ sqlldr
```

```
The server responds with help information because SQL*Loader was invoked without  
parameters:
```

```
SQL*Loader: Release 8.1.5.0.0 - Production on Wed Mar 10 7:20:11 1999  
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

```
Usage: SQLLOAD keyword=value [,keyword=value,...]
```

Valid Keywords:

```
userid -- ORACLE username/password
control -- Control file name
log -- Log file name
bad -- Bad file name
data -- Data file name
discard -- Discard file name
discardmax -- Number of discards to allow (Default all)
skip -- Number of logical records to skip (Default 0)
load -- Number of logical records to load (Default all)
errors -- Number of errors to allow (Default 50)
rows -- Number of rows in conventional path bind array or between
        direct path data saves
        (Default: Conventional path 64, Direct path all)
bindsize -- Size of conventional path bind array in bytes (Default
65536)
silent -- Suppress messages during run
        (header, feedback, errors, discards, partitions)
direct -- use direct path (Default FALSE)
parfile -- parameter file: name of file that contains parameter
        specifications
parallel -- do parallel load (Default FALSE)
readsize -- Size (in bytes) of the read buffer
file -- File to allocate extents from
skip_unusable_indexes -- disallow/allow unusable indexes or index
        partitions (Default FALSE)
skip_index_maintenance -- do not maintain indexes, mark affected indexes
        as unusable (Default FALSE)
commit_discontinued -- commit loaded rows when load is discontinued
        (Default FALSE)
```

PLEASE NOTE: Command-line parameters may be specified either by position or by keywords. An example of the former case is 'sqlload scott/tiger foo'; an example of the latter is 'sqlload control=foo userid=scott/tiger'. One may specify parameters by position before but not after parameters specified by keywords. For example,

'sqlload scott/tiger control=foo logfile=log' is allowed, but 'sqlload scott/tiger control=foo log' is not, even though the position of the parameter 'log' is correct.

31.2 SQL*Loader组件

SQL*Loader是一个Oracle工具，能够将数据从外部数据文件装载到数据库中。图 31-1显示了SQL*Loader的不同组件。

控制文件

控制文件是SQL*Loader的中枢核心，控制文件能够控制外部数据文件中的数据如何映射到Oracle的表和列。注意，SQL*Loader输入数据类型完全独立于它们将被载入的数据库列的数据类型。如果需要，将进行隐式的数据类型转换，如果转换失败，将捕捉错误消息。在控制文件中使用的语言是SQL*Loader数据定义语言（Data Definition Language，DDL）。控制文件由带有许多参数的多个部分组成，在本章中无法深入地涉及到每个部分。参阅《Oracle服务器工具手册》（Oracle Server Utilities manual）了解控制文件全部合法语法及参数的有关文档。基本的语法包含在31.3节“SQL*Loader示例一览”部分，用来说明SQL*Loader的各种不同应用。

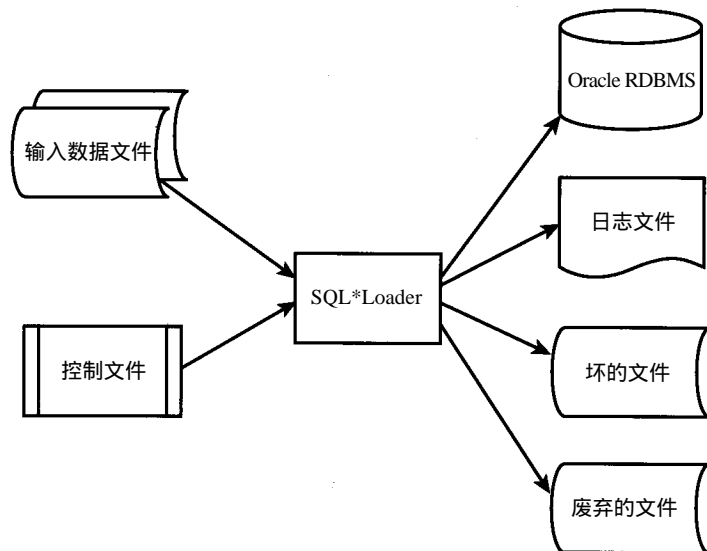


图31-1 SQL*Loader组件

31.2.1 SQL*Loader输入数据

SQL*Loader能够接收多种不同格式的数据文件。文件可以存储在磁盘或磁带上，或记录本身可以被嵌套到控制文件中。记录格式可以是定长的或变长的，定长记录是指这样的记录：每条记录具有相同的固定长度，并且每条记录中的数据域也具有相同的固定长度、数据类型和位置。例如，一条记录中的 PART_NBR数据项总是占据第10到19列，而不管数据的实际长度。如果部件号为12345，在15列至19列中剩余的部分将为空白。对于变长记录，数据项只占据每条记录实际需要的空间。

在PART_NBR例子中，数据项只需要5个字节，没有尾随的空格。在变长格式文件中的每条记录可以拥有PART_NBR项不同的空间，这完全取决于该数据项的实际长度。虽然变长记录可能在数据文件中使用较少的空间，但是每条记录上的数据项必须有一个间隔符以分隔每个数据项，意识到这点非常重要。

31.2.2 SQL*Loader输出

下面几节讨论由一个SQL*Loader会话创建的各种文件。

1. Oracle表和索引

SQL*Loader可以在同一个装载会话中向Oracle数据库装载多个表和索引，SQL*Loader插入数据和创建索引的行为将在31.4.1节“使用常规路径装载”与31.4.2节“使用直接路径装载”部分讨论。

2. 坏文件

SQL*Loader验证插入数据库中的数据要经历两阶段过程。第一阶段是按照控制文件中的说明验证数据的格式。如果数据的格式或长度与说明不一致，SQL*Loader将该记录写入坏文件。当记录通过第一阶段的验证后，它们被传递给数据库用于插入。

第二阶段的验证工作发生在数据库内部，数据库出于各种原因可能会拒绝记录，某些原

因可能是数据库检查约束错误和数据类型转换错误。第二阶段验证拒绝也被写入坏文件。如果拒绝的数目达到一定阈值（默认值为 50），SQL*Loader会话将终止退出，这个阈值可以在命令行上使用 errors 参数设置，采用与初始输入数据文件相同的格式写入坏文件，这样一来，在做出了必要的改正后，可以使用同一个控制文件装载坏文件记录。

3. 废弃文件

如果控制文件中含有条件并且记录不能满足全部的条件，SQL*Loader将这些记录写入废弃文件。例如，控制文件中的一项条件说明在列 1 必须拥有一个 X，没有 X 的记录将不会被插入数据库，而且要将它们写入废弃文件。与坏文件不同，废弃记录的缺省阈值为允许全部的废弃记录，可以在命令行使用 discardmax 参数将该阈值调低。

4. 日志文件

在SQL*Loader执行时，它创建一个日志文件。妨碍该日志文件成功地创建的任意情形都将终止装载会话。日志文件的缺省名称为带有 .log 扩展名的控制文件名。如果在命令行没有使用 log 参数给出一个新的文件名，装载会话自动地使用相同名称重写上一个日志文件，记住这一点很重要。日志文件含有多个部分，使用结果和小结统计信息显示装载会话运行的环境。在31.3节“SQL*Loader示例一览”部分中的程序清单 31-8显示了一个日志文件中的内容。

31.2.3 控制文件语法

大部分控制文件使用下列的关键字开头：

LOAD DATA

可能在它们之前出现的其他关键字是 ' - '，它是注释，是可选择的，它使得先前讨论的命令行选项能够被包含进控制文件。

后面紧跟着定义，以指明用于装载的源外部数据文件：

INFILE 'mydata.dat'

通过指定多个 INFILE 语句，可以在同一个会话中装载多个数据文件：

INFILE 'mydata1.dat'

INFILE 'mydata2.dat'

如果没有指定文件的扩展名，SQL*Loader默认文件的扩展名为 .dat。虽然没有要求使用单引号将数据文件括起来，但当指定完全数据路径时，我们强烈建议使用单引号将数据文件括起来，以避免错误的特殊字符转换。在装载会话中所有表使用的装载方法（参见表 31-1）也被指定。

表31-1 表装载方法

方 法	说 明
INSERT	这是缺省方法。该方法假设在数据装载前表是空的。如果在表中仍然有记录，SQL*Loader退出
APPEND 行	这种方法允许记录被添加到数据库表中，而且不影响已经存在的记录
REPLACE	这种方法首先删除表中已经存在的记录，然后开始装载新的记录。注意，当老记录被删除时，表上的任意删除触发器将被触发
TRUNCATE	这种方法在装载数据前，使用 SQL 命令 TRUNCATE 删除老的记录，因为去除了触发器的触发并且没有创建回滚，所以这种方法要比 REPLACE 快得多。TRUNCATE 不是一个可恢复命令。为了使用这种方法，表引用完整性约束必须被禁止，并且要授予特定的权限

后面跟着表定义：

INTO TABLE 表名称 方法

在这个例子中，方法与上面表 31-1 中说明的相同，但是这种方法只适用于在 INTO TABLE 行上指定的表。当从同一个 SQL*Loader 会话中装载多个表时，这给了用户较好的控制。

跟在 INTO TABLE 关键字后面的是列与数据类型说明。不需要回顾全部不同的选项，你可以较为容易地浏览本章后面部分中的不同例子。

31.3 SQL*Loader 示例一览

所有的示例使用下面四个表（清单 31-2）组成的模式，这个模式使用客户、帐户和事务表模拟了一个银行模式。为了表明向一个分区表装载数据，partition_xact 表是事务表的副本，具有基于事务发生时刻的数据分区。

清单 31-2 LIST1.1——模式示例

```
create table customer (
    cust_nbr      number(7)      not null,
    cust_name     varchar2(100)   not null,
    cust_addr1    varchar2(50),
    cust_addr2    varchar2(50),
    cust_city     varchar2(30),
    cust_state    varchar2(2),
    cust_zip      varchar2(10),
    cust_phone    varchar2(20),
    cust_birthday date)
/

create table account (
    cust_nbr      number(7)      not null,
    acct_nbr      number(10)     not null,
    acct_name     varchar2(40)   not null)
/

create table transaction (
    acct_nbr      number(10)     not null,
    xact_amt      number(10,2)   not null,
    xact_flag     char           not null,
    xact_date     date           not null)
/

create table partition_xact (
    acct_nbr      number(10)     not null,
    xact_amt      number(10,2)   not null,
    xact_flag     char           not null,
    xact_date     date           not null)

PARTITION BY RANGE (xact_date)
(PARTITION P1 VALUES LESS THAN (to_date('01-APR-1999','DD-MON-YYYY')),
PARTITION P2 VALUES LESS THAN (to_date('01-JUL-1999','DD-MON-YYYY')),
PARTITION P3 VALUES LESS THAN (to_date('01-OCT-1999','DD-MON-YYYY')),
PARTITION P4 VALUES LESS THAN (MAXVALUE))
/
```

所有的示例使用下面的数据文件（参见清单 31-3、清单 31-4 与清单 31-5）。

清单 31-3 cust.dat——清单说明

0000001BOB MARIN	123 MAIN ST.	TOPEKA	KS12345
999-555-1234	20-APR-55		
0000002MARY JOHNSON	18 HOPE LANE	SAN FRANCISCO	CA94054
415-555-1299	32-JAN-69		
0000003RICHARD WILLIAMS	1225 DAFFODIL LANE	BOSTON	MA98377
0000004WALTER SIMS	1888 PROSPECT AVE.	BROOKLYN	NY11218
718-555-3420			
0000005LARRY HATFIELD	TWO FIELDS CT.	SOMERSET	NJ07689
732-555-2454	25-DEC-60		
0000006LAURA LAU	25 CHRISTOPHER LN	SAN BRUNO	CA90234
510-555-4834			
0000123PRISCILLA WONG	888 FORTUNE COURT	PHILADELPHIA	PA35545
01-JAN-65			
0000068SONNY BALRUP	27 KAMA ST.	JACKSON HEIGHTS	NY10199
718-555-9876	07-MAY-61		
0023494RUPAL PARIKH	2 FORCE BLVD	NEW YORK	NY10105
212-555-5887	31-DEC-72		
0000324CRAIG SILVEIRA	1674 ISLAND ST	SMITHTOWN	NY12467
516-555-5534	27-OCT-74		
0000010DANIEL SMITH	35 DIRECT DRIVE	BERGEN	NJ07899
201-555-3734			
0011102STEPHEN LEUNG	16 STANFORD CT	STANFORD	CA96688
650-555-1248	05-SEP-76		
0011102ALICIA LOWRY	5678 TIMOTHY DR	ATLANTA	GA47730
0002340JENNIFER LEUNG	1 MURRAY HILL	GREENWICH	CT78835
203-555-7564			
1003423HORACE MICHAEL	90 MINISTER ST	MINNEAPOLIS	MN77788
18-MAR-65			
0000223CHRISTOPHER YEE	9077 MUSIC AVE	DETROIT	MI45345
777-555-7785	22-JUL-75		
0009032JAMES BORIOTTI	65 FIREMENS LANE	COLUMBUS	OH37485
904-555-5674			
0000088HIREN PATEL	69 CLUB ST.	NEW YORK	NY12445
212-555-7822	12-APR-70		
0000100RICHARD JI	1225 STEER ST	KOBE	KS12009
999-555-5824	10-OCT-74		
0000046DAVID CHOW	49 HUGO DRIVE	FLUSHING	NY10199
718-555-4367			
0000758HENRY WALKER	12 SIGMUND ST.	CHICAGO	IL33890
312-555-5567	09-APR-45		
0002993GEORGE BLOOM	28 BRIDGEWATER ST	SAN MATEO	CA90475
650-555-2838	25-MAY-63		
0009488LISA JONES	30 MISSION ST	UNITY	FL23899

清单 31-4 acct.dat——清单说明

0000001,459023,SAVINGS
0000001,459024,CHECKING
0000003,211108,SAVINGS
0000003,211123,CHECKING
0000006,23388,SAVINGS
0000123,43992,CHECKING
0000123,50699390,LINE OF CREDIT
0000068,23330,SAVINGS
0023494,433020,SAVINGS
0000010,4566,SAVINGS
0000010,4599,CHECKING
0000223,8887544,SAVINGS

清单31-5 xact.dat——清单说明

0000459023	123.45D01-FEB-99
0000459023	1233.86C01-MAR-99
0000459023	987.00P01-DEC-99
0000459024	1000C03-JUN-99
0000211108	875.27D23-JUL-99
0000211123	20987.88C30-DEC-99
0000211123	12500.16D10-JAN-99
0000023388	1.75C19-MAY-99
0000043992	350.00C12-MAR-99
0050699390	2899.09D01-SEP-99
0000023330	100D26-JAN-99
0000433020	60.99C20-NOV-99
0000004566	230.23C20-AUG-99
0000004599	14.96D05-JUN-99
0000004599	14.AAD07-JUN-99
0008887544	9999.99D11-JUL-99

31.3.1 例子1：装载定长数据

这个示例将cust.dat数据文件中的数据装载到客户表，因为数据是定长格式，所以控制文件（参见清单31-6）按列的位置将数据映射到数据库。

清单31-6 load1.ctl——控制文件

```

LOAD DATA
INFILE 'cust.dat'
INTO TABLE customer
(cust_nbr    POSITION(01:07)    INTEGER EXTERNAL,
 cust_name   POSITION(08:27)    CHAR,
 cust_addr1  POSITION(28:47)    CHAR,
 cust_city   POSITION(48:67)    CHAR,
 cust_state  POSITION(68:69)    CHAR,
 cust_zip    POSITION(70:79)    CHAR,
 cust_phone  POSITION(80:91)    CHAR,
 cust_birthdate POSITION(100:108) DATE "DD-MON-YY" NULLIF cust_birthdate=BLANKS)

```

使用example/expass作为用户名和口令激活SQL*Loader，正如前面在“SQL*Loader输出”部分讨论的，控制文件、日志文件、坏文件和废弃文件作为命令行参数传递给SQL*Loader。（见清单31-7、清单31-8）

清单31-7 load1.ctl——激活SQL*Loader

```

$ sqlldr example/expass control=load1.ctl log=load1.log bad=load1.bad
discards=load1.dis

```

下面是服务器的响应信息：

```

SQL*Loader: Release 8.0.3.0.0 - Production on Wed Mar 10 8:10:23 1999
(c) Copyright 1997 Oracle Corporation. All rights reserved.

Commit point reached - logical record count 23

```

清单31-8 load1.ctl——例子1日志文件内容

```

SQL*Loader: Release 8.0.3.0.0 - Production on Wed Mar 10 8:11:03 1999

(c) Copyright 1997 Oracle Corporation. All rights reserved.

```

Control File: load1.ctl
 Data File: cust.dat
 Bad File: load1.bad
 Discard File: load1.dis
 (Allow all discards)

Number to load: ALL
 Number to skip: 0
 Errors allowed: 50
 Bind array: 64 rows, maximum of 65536 bytes
 Continuation: none specified
 Path used: Conventional

Table CUSTOMER, loaded from every logical record.
 Insert option in effect for this table: INSERT

Column Name	Position	Len	Term	Encl	Datatype
CUST_NBR	1:7	7			
CHARACTER					
CUST_NAME	8:27	20			CHARACTER
CUST_ADDR1	28:47	20			CHARACTER
CUST_CITY	48:67	20			CHARACTER
CUST_STATE	68:69	2			CHARACTER
CUST_ZIP	70:79	10			CHARACTER
CUST_PHONE	80:91	12			CHARACTER
CUST_BIRTHDAY	100:108	9			DATE DD-MON-YY

Column CUST_NAME is NULL if CUST_NAME = BLANKS
 Column CUST_ADDR1 is NULL if CUST_ADDR1 = BLANKS
 Column CUST_CITY is NULL if CUST_CITY = BLANKS
 Column CUST_STATE is NULL if CUST_STATE = BLANKS
 Column CUST_ZIP is NULL if CUST_ZIP = BLANKS
 Column CUST_PHONE is NULL if CUST_PHONE = BLANKS
 Column CUST_BIRTHDAY is NULL if CUST_BIRTHDAY = BLANKS

Record 2: Rejected - Error on table CUSTOMER, column CUST_BIRTHDAY.
 ORA-01847: day of month must be between 1 and last day of month

Table CUSTOMER:

22 Rows successfully loaded.
 1 Row not loaded due to data errors.
 0 Rows not loaded because all WHEN clauses were failed.
 0 Rows not loaded because all fields were null.
 Space allocated for bind array: 9216 bytes(64 rows)
 Space allocated for memory besides bind array: 0 bytes

Total logical records skipped: 0
 Total logical records read: 23
 Total logical records rejected: 1
 Total logical records discarded: 0

Run began on Wed Mar 10 8:11:03 1999
 Run ended on Wed Mar 10 8:11:03 1999

Elapsed time was: 00:00:00.21
 CPU time was: 00:00:00.04

例子1注释：由于非法的日期，记录2被拒绝并写入坏文件（load1.bad）。这条记录可以在坏文件中被改正并通过下列方法装载到数据库：对同一个控制文件做出更改，使用坏文件作为输入文件并在关键字 INTO TABLE前添加关键字 APPEND。还有，请注意在控制文件中

NULLIF子句的使用，没有这个子句，日期中含有空格的记录将不能进行数据库日期检查。

31.3.2 例子2：装载变长数据

这个示例将数据文件 acct.dat 中的数据装载到客户表。因为日期采用变长格式，控制文件定义了分隔符用于区分不同的数据项（见清单 31-9、清单 31-10）。

清单31-9 LOAD2.CTL——例子2控制文件

```
LOAD DATA

INFILE 'acct.dat'

INTO TABLE account

FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'

(cust_nbr, acct_nbr, acct_name)
```

为例子2在命令行激活SQL*Loader的方法如下所示：

```
$ sqlldr example/express control=load2.ctl log=load2.log bad=load2.bad
discard=load2.dis
```

下面是服务器的响应信息：

```
SQL*Loader: Release 8.0.3.0.0 - Production on Wed Mar 10 8:15:51 1999
(c) Copyright 1997 Oracle Corporation. All rights reserved.
Commit point reached - logical record count 12
```

清单31-10 例子2日志文件内容

```
SQL*Loader: Release 8.0.3.0.0 - Production on Wed Mar 10 8:16:03 1999
(c) Copyright 1997 Oracle Corporation. All rights reserved.
```

```
Control File:   load2.ctl
Data File:      acct.dat
Bad File:       load2.bad
Discard File:   load2.dis
(Allow all discards)
```

```
Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:     64 rows, maximum of 65536 bytes
Continuation:   none specified
Path used:      Conventional
```

```
Table ACCOUNT, loaded from every logical record.
Insert option in effect for this table: INSERT
```

Column Name	Position	Len	Term	Encl	Datatype
CUST_NBR	FIRST	*	,	0(")	CHARACTER
ACCT_NBR	NEXT	*	,	0(")	CHARACTER
ACCT_NAME	NEXT	*	,	0(")	CHARACTER

Table ACCOUNT:

```

12 Rows successfully loaded.
0 Rows not loaded due to data errors.
0 Rows not loaded because all WHEN clauses were failed.
0 Rows not loaded because all fields were null.

Space allocated for bind array:          35328 bytes(64 rows)
Space allocated for memory besides bind array:    0 bytes

Total logical records skipped:          0
Total logical records read:             12
Total logical records rejected:         0
Total logical records discarded:        0

Run began on Wed Mar 10 8:16:03 1999
Run ended on Wed Mar 10 8:16:28 1999

Elapsed time was:      00:00:00.23
CPU time was:         00:00:00.04

```

例子2注释：全部记录都被成功地装载。记住，变长数据必须含有分隔符以分隔输入数据文件内部的数据项。当装载分隔的字符数据时，定义每个字符数据域的最大长度更有效率。在这个例子中，应该在控制文件中使用 `acct_name char(20)`。如果没有定义最大长度，SQL*Loader使用255字节作为长度的缺省值，这会影响在每个赋值数组执行中有多少记录被插入。

31.3.3 例子3：装载嵌套数据

这个示例表明数据不一定必须在一个数据文件中，数据可以直接安置在控制文件中。BEGINDATA关键字指示在它之后的全部信息行都是数据记录，可以被用作该控制文件的输入源（见清单31-11、清单31-12）。

清单31-11 LOAD3.CTL——例子3控制文件

```

LOAD DATA
INFILE *
APPEND
INTO TABLE account
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
(cust_nbr, acct_nbr, acct_name)

BEGINDATA

0000324,89073,SAVINGS
0000324,89074,CHECKING
0000075,111,SAVINGS
0011102,800,CHECKING
0000068,23338,CHECKING

```

从命令行为例子3激活SQL*Loader，如下所示：

```
$ sqlldr example/epass control=load3.ctl log=load3.log bad=load3.bad
discard=load3.dis
```

下面是服务器的响应信息：

```
SQL*Loader: Release 8.0.3.0.0 - Production on Wed Mar 10 8:20:11 1999
(c) Copyright 1997 Oracle Corporation. All rights reserved.
```

Commit point reached - logical record count 5

清单31-12 例子3日志文件内容

SQL*Loader: Release 8.0.3.0.0 - Production on Wed Mar 10 8:21:21 1999

(c) Copyright 1997 Oracle Corporation. All rights reserved.

Control File: load3.ctl
Data File: load3.ctl
Bad File: load3.bad
Discard File: load3.dis
(Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array: 64 rows, maximum of 65536 bytes
Continuation: none specified
Path used: Conventional
Table ACCOUNT, loaded from every logical record.
Insert option in effect for this table: APPEND

Column Name	Position	Len	Term	Encl	Datatype
CUST_NBR	FIRST	*	,	0(")	CHARACTER
ACCT_NBR	NEXT	*	,	0(")	CHARACTER
ACCT_NAME	NEXT	*	,	0(")	CHARACTER

Table ACCOUNT:

5 Rows successfully loaded.
0 Rows not loaded due to data errors.
0 Rows not loaded because all WHEN clauses were failed.
0 Rows not loaded because all fields were null.

Space allocated for bind array: 35328 bytes(64 rows)
Space allocated for memory besides bind array: 0 bytes

Total logical records skipped: 0
Total logical records read: 5
Total logical records rejected: 0
Total logical records discarded: 0

Run began on Wed Mar 10 8:20:11 1999
Run ended on Wed Mar 10 8:20:35 1999

Elapsed time was: 00:00:00.24
CPU time was: 00:00:00.03

例子3注释：要装载放置在控制文件中的数据，在关键字 INFILE后面使用*代替一个输入文件名。这个例子还显示了 APPEND装载方法的使用，使用该方法添加帐户表。如果在控制文件中省略 APPEND关键字，因为 SQL*Loader缺省的装载方法是 INSERT，这要求在装载前表必须为空，所以装载过程终止退出。

31.3.4 例子4：条件装载

这个示例从文件 xact.dat装载数据。控制文件检查标志，查看总额是借方还是贷方，并据此加载数据（见清单31-13、清单31-14）。

清单31-13 例子4控制文件

LOAD DATA

```

INFILE 'xact.dat'
INTO TABLE transaction
WHEN xact_flag = 'D'

```

```

(acct_nbr    POSITION(01:10)  INTEGER EXTERNAL,
 xact_amt    POSITION(11:20)  INTEGER EXTERNAL ":xact_amt * -1",
 xact_flag    POSITION(21:21)  CHAR,
 xact_date    POSITION(22:31)  DATE "DD-MON-YY" NULLIF xact_date=BLANKS)

```

```

INTO TABLE transaction
WHEN xact_flag = 'C'

```

```

(acct_nbr    POSITION(01:10)  INTEGER EXTERNAL,
 xact_amt    POSITION(11:20)  INTEGER EXTERNAL,
 xact_flag    POSITION(21:21)  CHAR,
 xact_date    POSITION(22:31)  DATE "DD-MON-YY" NULLIF xact_date=BLANKS)

```

从命令行为例子4激活SQL*Loader，如下所示：

```

$ sqlldr example/expass control=load4.ctl log=load4.log bad=load4.bad
discards=load4.dis

```

下面是服务器的响应信息：

```

SQL*Loader: Release 8.0.3.0.0 - Production on Wed Mar 10 9:20:11 1999
(c) Copyright 1997 Oracle Corporation. All rights reserved.
Commit point reached - logical record count 16

```

清单31-14 例子4日志文件内容

```

SQL*Loader: Release 8.0.3.0.0 - Production on Wed Mar 10 9:20:11 1999
(c) Copyright 1997 Oracle Corporation. All rights reserved.

```

```

Control File:  load4.ctl
Data File:     xact.dat
Bad File:      load4.bad
Discard File:  load4.dis
(Allow all discards)

```

```

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:     64 rows, maximum of 65536 bytes
Continuation:   none specified
Path used:      Conventional

```

```

Table TRANSACTION, loaded when XACT_FLAG = 0X44(character 'D')
Insert option in effect for this table: INSERT

```

Column Name	Position	Len	Term	Encl	Datatype
ACCT_NBR	1:10	10			CHARACTER
XACT_AMT	11:20	10			CHARACTER
XACT_FLAG	21:21	1			CHARACTER
XACT_DATE	22:31	10			DATE DD-MON-YY

Column XACT_AMT had SQL string

```

":xact_amt * -1"
  applied to it.
Column XACT_DATE is NULL if XACT_DATE = BLANKS
Table TRANSACTION, loaded when XACT_FLAG = 0X43(character 'C')
Insert option in effect for this table: INSERT

```

Column Name	Position	Len	Term	Encl	Datatype
ACCT_NBR	1:10	10			CHARACTER
XACT_AMT	11:20	10			CHARACTER
XACT_FLAG	21:21	1			CHARACTER
XACT_DATE	22:31	10			DATE DD-MON-YY

Column XACT_DATE is NULL if XACT_DATE = BLANKS

Record 3: Discarded - failed all WHEN clauses.

Record 15: Rejected - Error on table TRANSACTION, column XACT_AMT.

ORA-01722: invalid number

Table TRANSACTION:

```

7 Rows successfully loaded.
1 Row not loaded due to data errors.
8 Rows not loaded because all WHEN clauses were failed.
0 Rows not loaded because all fields were null.

```

Table TRANSACTION:

```

7 Rows successfully loaded.
0 Rows not loaded due to data errors.
9 Rows not loaded because all WHEN clauses were failed.
0 Rows not loaded because all fields were null.

```

```

Space allocated for bind array:          7168 bytes(64 rows)
Space allocated for memory besides bind array:  0 bytes

```

```

Total logical records skipped:          0
Total logical records read:              16
Total logical records rejected:          1
Total logical records discarded:         1

```

Run began on Wed Mar 10 9:20:11 1999

Run ended on Wed Mar 10 9:20:46 1999

```

Elapsed time was:      00:00:00.35
CPU time was:          00:00:00.02

```

例子4注释：这个例子显示了如何在控制文件中使用 WHEN子句进行条件装载。在这个例子中，多个 WHEN子句装载同一个表，但是它们可以是不同的表。当事务标志等于 D时，在插入表前，美元数量被乘以 -1。这是一个说明当数据被装载时，能够对数据使用运算符或 SQL函数的例子。

因为标志既不是 C也不是 D，不满足任何一个条件，所以记录 3被废弃并被放入废弃文件中。因为美元数目不是数字型数据，所以记录 15被拒绝。这两条记录显示了前面提到的两阶段检查。记录3被装载过程废弃，而记录 15直到数据库试图将它插入到表中时才被拒绝。

31.3.5 例子5：装载到分区表

这个示例是例子4的一个变种，要从文件xact.dat加载数据。在此要将数据装载到partition_xact

表，该表有四个分区，按照日历季度存储数据，表的一个分区或全部分区可以在一个装载会话中装载。在本例中，只为第一个季度装载了分区p1（见清单31-15、清单31-16）。

清单31-15 LOAD5.CTL——例子5控制文件

LOAD DATA

```

INFILE 'xact.dat'
INTO TABLE partition_xact PARTITION (P1)
WHEN xact_flag = 'D'

  (acct_nbr    POSITION(01:10)    INTEGER EXTERNAL,
   xact_amt    POSITION(11:20)    INTEGER EXTERNAL ":xact_amt * -1",
   xact_flag    POSITION(21:21)    CHAR,
   xact_date    POSITION(22:31)    DATE "DD-MON-YY" NULLIF xact_date=BLANKS)

INTO TABLE partition_xact PARTITION (P1)
WHEN xact_flag = 'C'
  (acct_nbr    POSITION(01:10)    INTEGER EXTERNAL,
   xact_amt    POSITION(11:20)    INTEGER EXTERNAL,
   xact_flag    POSITION(21:21)    CHAR,
   xact_date    POSITION(22:31)    DATE "DD-MON-YY" NULLIF xact_date=BLANKS)

```

从命令行为例子5激活SQL*Loader的命令如下所示：

```
$ sqlldr example/expass control=load5.ctl log=load5.log bad=load5.bad
discards=load5.dis
```

下面是服务器的响应信息：

```

SQL*Loader: Release 8.0.3.0.0 - Production on Wed Mar 10 10:20:11 1999
(c) Copyright 1997 Oracle Corporation. All rights reserved.
Commit point reached - logical record count 16

```

清单31-16 例子5日志文件内容

```

SQL*Loader: Release 8.0.3.0.0 - Production on Wed Mar 10 10:20:11 1999

```

```
(c) Copyright 1997 Oracle Corporation. All rights reserved.
```

```

Control File:  load5.ctl
Data File:     xact.dat
Bad File:      load5.bad
Discard File:  load5.dis
(Allow all discards)

```

```

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:     64 rows, maximum of 65536 bytes
Continuation:   none specified
Path used:      Conventional

```

```

Table PARTITION_XACT, partition P1, loaded when XACT_FLAG = 0X44(character 'D')
Insert option in effect for this partition: INSERT

```

Column Name	Position	Len	Term	Encl	Datatype
ACCT_NBR	1:10	10			CHARACTER
XACT_AMT	11:20	10			CHARACTER
XACT_FLAG	21:21	1			CHARACTER

XACT_DATE 22:31 10 DATE DD-MON-YY

Column XACT_AMT had SQL string

":xact_amt * -1"

applied to it.

Column XACT_DATE is NULL if XACT_DATE = BLANKS

Table PARTITION_XACT, partition P1, loaded when XACT_FLAG = 0X43(character 'C')

Insert option in effect for this partition: INSERT

Column Name	Position	Len	Term Encl	Datatype
ACCT_NBR	1:10	10		CHARACTER
XACT_AMT	11:20	10		CHARACTER
XACT_FLAG	21:21	1		CHARACTER
XACT_DATE	22:31	10		DATE DD-MON-YY

Column XACT_DATE is NULL if XACT_DATE = BLANKS

Record 3: Discarded - failed all WHEN clauses.

Record 5: Rejected - Error on table PARTITION_XACT, partition P1.

ORA-14401: inserted partition key is outside specified partition

Record 10: Rejected - Error on table PARTITION_XACT, partition P1.

ORA-14401: inserted partition key is outside specified partition

Record 14: Rejected - Error on table PARTITION_XACT, partition P1.

ORA-14401: inserted partition key is outside specified partition

Record 15: Rejected - Error on table PARTITION_XACT, column XACT_AMT.

ORA-01722: invalid number

Record 16: Rejected - Error on table PARTITION_XACT, partition P1.

ORA-14401: inserted partition key is outside specified partition

Record 4: Rejected - Error on table PARTITION_XACT, partition P1.

ORA-14401: inserted partition key is outside specified partition

Record 6: Rejected - Error on table PARTITION_XACT, partition P1.

ORA-14401: inserted partition key is outside specified partition

Record 8: Rejected - Error on table PARTITION_XACT, partition P1.

ORA-14401: inserted partition key is outside specified partition

Record 12: Rejected - Error on table PARTITION_XACT, partition P1.

ORA-14401: inserted partition key is outside specified partition

Record 13: Rejected - Error on table PARTITION_XACT, partition P1.

ORA-14401: inserted partition key is outside specified partition

Table PARTITION_XACT, partition P1:

3 Rows successfully loaded.

5 Rows not loaded due to data errors.

8 Rows not loaded because all WHEN clauses were failed.

0 Rows not loaded because all fields were null.

Table PARTITION_XACT, partition P1:

2 Rows successfully loaded.

5 Rows not loaded due to data errors.

9 Rows not loaded because all WHEN clauses were failed.

0 Rows not loaded because all fields were null.

Space allocated for bind array:

7168 bytes(64 rows)

Space allocated for memory besides bind array:

0 bytes

Total logical records skipped: 0

Total logical records read: 16

Total logical records rejected: 10

Total logical records discarded: 1

Run began on Wed Mar 10 10:21:39 1999

Run ended on Wed Mar 10 8:22:01 1999

Elapsed time was: 00:00:00.31

CPU time was: 00:00:00.04

例子5注释：要装载分区表的一个分区，需要在控制文件中这个表名的后面添加关键字 PARTITION。请注意分区名称（在本例中为 P1），必须被圆括号括起来。如果省略关键字 PARTITION，全分区都将被装载。没有在 P1 的分区范围中的全部记录都被写入坏文件中。

31.4 常规路径装载与直接路径装载

SQL*Loader提供了两种方法装载数据——常规路径装载与直接路径装载。常规路径装载是SQL*Loader的默认装载方法。为了能够进行直接路径装载，在激活 SQL*Loader时，必须将 DIRECT=TRUE 添加到命令行参数中。正如在图 31-2 中所看到的，常规路径装载具有直接路径装载不需要的一些额外步骤。这些额外步骤增加了系统处理的额外开销，使得常规路径装载的速度慢于直接路径装载。格式化 SQL INSERT 语句以及搜索 SGA 内存高速缓冲区的额外步骤与其他同时并发运行在数据库上的进程产生竞争。虽然出于速度原因，我们倾向于使用直接路径装载，但是当使用常规路径装载时，还有一些限制和情况。这将在下面的部分中涉及到。

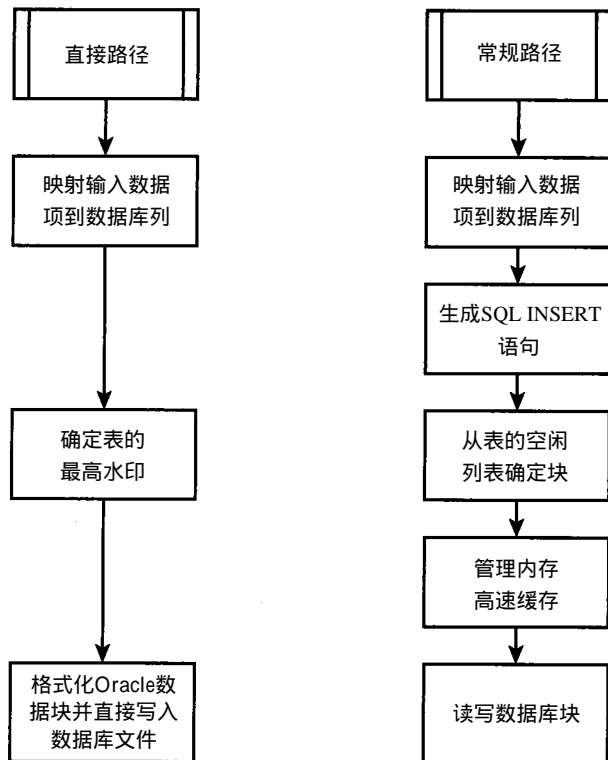


图31-2 常规路径装载与直接路径装载方法

31.4.1 使用常规路径装载

常规路径装载使用 SQL INSERT 语句和内存中的键数组缓存（bond array buffers）将数据

装载到 Oracle 数据库的表中。这个过程与其他进程竞争 SGA 内部的内存资源。如果数据库已经有支持多个并发处理进程的开销，常规路径装载会降低装载的性能。

使用常规路径装载的另外一个开销是装载进程必须搜索数据库，以查找被装载表的部分填充块，并试图填充这些块。这对日常的事务处理是非常有效的，但是它是常规路径装载的一个额外开销。

下面是一些情形与案例，要求最好或有时必须使用常规路径装载方法，而不能使用直接路径装载：

如果被装载的表是被索引的并且被并发访问的，或者如果要对表进行插入或删除，必须使用常规路径装载。

当在控制文件中使用 SQL 函数时，必须使用常规路径装载。当使用直接路径装载时，SQL 函数将不适用。

当装载的表是一个簇表时。

当装载少量记录到一个大型索引表，或当表具有引用完整性或检查约束时。将在下面的“使用直接路径装载”部分解释为什么在这种情形下使用常规路径装载要好一些。

当装载工作是通过 SQL*Net 或 Net8 在不同的平台上进行时，为使用直接路径装载，两个节点必须属于同一个计算机家族并且使用同样的字符集。

31.4.2 使用直接路径装载

不需要使用 SQL INSERT 语句和键数组缓存，直接路径装载格式化输入数据到 Oracle 数据库并将它们直接写入数据库中。注意直接路径装载总是在表的最高水位之上插入数据，这种方式消除了用于搜索部分填充块的时间。如果使用直接路径装载且装载工作没有完成，并且如果装载从开始处重新开始，那么被装载的数据将被截去，或者表会被删除并被重新创建。仅仅简单地删除数据并不能复位最高水印，而且会使用更多的存储空间。

理解使用直接路径装载方法对于表的索引会发生什么非常重要，这可以最大限度地提高直接路径装载的效率。在 Oracle8 中，当装载数据变为可见时（移动最高水印并提交），索引被标志为无用的，因为在这点上索引相对于它们索引的数据来说是过时的。当每个索引被更新为最新时，索引的无用状态便被消除。这使得当直接路径装载进行时，针对表的查询可以发生。但是，在装载结束阶段（移动最高水印并更新索引），如果查询要求一个处于无用状态的索引，这个查询可能会失败。

在装载期间，新的索引值写入临时段，在装载结束时，它们被存储并与旧的索引值合并。这时，索引再次变为可用。这样可以显著地提高对临时空间的需要。为了避免对临时空间的额外需求以进行排序，可以按照索引的序列对数据进行预排序，并且在控制文件中使用关键字 SORTED INDEXES。

在前面有关常规路径装载的部分中，列出阻止使用直接路径装载的限制，装载少量的记录到一个带有索引和/或引用及检查约束的大型表不是使用直接路径装载的一个限制因素，但是使用常规路径装载更为有效。在存在索引的情况下，或许常规路径装载在装载数据时更新索引要快一些，而且不需要进行一个大的排序/组合来创建新的索引。对于引用和检查约束，直接路径装载要求在装载前禁止这些约束。在全部数据装载后，重新允许检查约束，整个数据表都将根据这些约束进行检查，而不仅仅是被装载的数据。

31.4.3 装载嵌套列、嵌套表与变长数组

嵌套列对象是这样的一种对象，在嵌套列中，一个列对象在另外一个列对象之中。

嵌套表是另外一个表内部的表。这个嵌套表作为另外一个表中的列出现。这个表可以当作表被引用，也可以当作其他表中的列被引用。

变长数组是一个称为元素的有序对象的集合。这些元素可以依据它们在数组中的位置被索引。

程序清单 31-17 显示了两个嵌套列对象，em_contact 嵌套在 dept_mgr 中，它是表 dept 的一个嵌套列。

清单 31-17 LOAD6.CTL——例子6控制文件

```
LOAD DATA

INFILE 'example6.dat'

INTO TABLE dept

    (deptno          CHAR(5),
     dept_name        CHAR(30),
     dept_mgr         COLUMN OBJECT
       (name          CHAR(30),
        age            INTEGER EXTERNAL(3),
        emp_id         INTEGER EXTERNAL(7),
        em_contact     COLUMN OBJECT
          (name        CHAR(30),
           PHONE       CHAR(25))
       )
    )
```

例子6注释：装载数据到一个嵌套表中类似于装载数据到一个普通表。注意在前面例子中语法上的差别，在那里嵌套表作为一个列对象出现。

程序清单 31-18 显示了 VARRAY 和嵌套表在控制文件中的使用。

清单 31-18 LOAD7.CTL——例子7控制文件

```
LOAD DATA

INFILE 'example7.dat'

INTO TABLE dept
(deptno          CHAR(4),
dept_name        CHAR(30),
emp_count        FILLER INTEGER EXTERNAL(5),
emp_str          VARRAY COUNT(emp_count)
  (name          FILLER CHAR(30),
   emp_str       COLUMN OBJECT
     (name        CHAR(30),
      age          INTEGER EXTERNAL(3),
      emp_id       INTEGER EXTERNAL(7)
     )
  ),
proj_sid         FILLER CHAR(30),
emp_projects     NEXTED TABLE SDF(CONSTANT 'project.txt', "proj1"
  SID(proj_sid) TERMINATED BY ";")
  (project_id     POSITION(1:6) INTEGET EXTERNAL(6),
   project_name   POSITION(7:31) CHAR
  )
)
```

例子7注释：装载数据到一个变长数组要求 SQL*Loader 确切地知道有多少记录要被装载，这可以使用 COUNT 语法来获得，COUNT 含有一个参数，包含变长数组的元素数目。还要注意 COUNT 的参数域直接出现在 COUNT 语法之前。

注意 变长数组的每个元素在装载到数据库前，需要四个字节的客户内存。

SQL*Loader 可能为每个变长数组需要至少两倍这样的空间。例如，如果有 100 个元素的变长数组，将需要 400 个字节的客户内存，并且 SQL*Loader 可能需要超过 800 个字节的客户内存来处理这个变长数组。如果在装载过程中用光了内存，试一下给 BINDSIZE 或 ROWS 一个小一些的值，然后重新启动装载。

31.4.4 使用 SQL*Loader 装载 LOBS

LOBS 或巨型对象，属于两个类别，一类存储在数据库的内部，而另一类存储在数据库的外部。BLOBS、CLOBs 和 NCLOBs 存储在表中，外部存储的 LOB 存储在 BFILE 中。程序清单 31-19 说明了存储一个内部定义 LOB 的控制文件，程序清单 31-20 说明了存储一个内部定义 LOB 但是从一个 LOBFILE 文件中装载该 LOB 的控制文件，程序清单 31-21 说明了一个使用 BFILE 的控制文件。

清单 31-19 RESUME1.CTL——例子 8 控制文件

```
LOAD DATA

INFILE 'resume1.dat'

INTO TABLE resumes

  (resume_name    POSITION(1:21) CHAR,
   resume         POSITION(22:1000) CHAR
  )
```

例子 8 注释：这个装载控制文件非常直观，因为 resume_name 和 LOB 数据出现在同一个输入数据文件中。

清单 31-20 RESUME2.CTL——例子 9 控制文件

```
LOAD DATA

INFILE 'resume2.dat'

INTO TABLE resumes

  (resume_name    POSITION(1:21) CHAR,
   "RESUME2"      LOBFILE(CONSTANT 'Resume2' VARCHAR(4,2000))
  )
```

例子 9 注释：这个控制文件含有一个次级的数据文件，或称为 SDF，注意 CONSTANT 是 SDF 的名称，还要注意这个文件的头四个字节位置必须含有 LOB 对象的长度，这由 VARCHAR(4, 2000) 指定，这里 4 表示文件 RESUME2 的头四个字节的位位置将含有 LOB 对象的长度。

警告 当从 LOBFILES 文件中装载 LOB 数据时，如果 LOB 不能成功地装载，含有 LOB 数据的记录将被装载，但是含有一个空的 LOB 列。

清单31-21 RESUME3.CTL——例子10控制文件

```
LOAD DATA

INFILE 'resume3.dat'

INTO TABLE resumes
FIELDS TERMINATED BY ','

  (resume_name    CHAR(3),
   file_name      CHAR(50),
   resume         BFILE(CONSTANT "Resume", file_name)
  )
```

例子10注释：BFILE在操作系统文件中存储LOB数据，BFILE列存储真正含有LOB数据的外部文件的定位器。这对于巨型的LOBS数据，例如视频数据很方便。

31.4.5 SQL*Loader性能提示

下面是增加SQL*Loader性能的一些补充技巧：

- 1) 使用定位域而不要使用分隔域，分隔域要求装载机搜索数据以查找分隔符。定位域比较快，因为装载机只需要做简单的指针运算。
- 2) 为终止域指定最大长度，使每个捆绑数组更为有效地插入。
- 3) 预分配足够的存储空间。当数据被装载时，表中需要更多的空间，Oracle分配更多的区间以容纳数据，如果在数据装载期间频繁地做这项操作，处理的开销将非常大。在装载之前计算或估算存储空间需求能够让你预先创建必要的存储空间。
- 4) 如果可能，在控制文件中尽量避免使用 NULLIF和DEFAULTIF子句。这两个子句对于被装载的每条记录都会引起列运算。
- 5) 分割数据文件，并行运行常规路径装载。
- 6) 通过使用命令行参数ROWS，减少提交次数。
- 7) 避免不必要的字符集转换，确保客户端的NLS_LANG环境与服务器端的相同。
- 8) 只要可能，尽量使用直接路径装载方法。
- 9) 当使用直接路径装载方法时，为表的最大索引预先排序并使用 SORTED INDEXES子句。
- 10) 当使用直接路径装载方法时，尽量使用并行直接路径选项。
- 11) 在直接路径装载期间，尽可能少使用重做日志。有三种不同级别的控制实现这点：
 - 禁止数据库归档。
 - 在控制文件中使用关键字 UNRECOVERABLE。
 - 使用NOLOG属性修改表和/或索引。

31.5 小结

SQL*Loader是由Oracle公司提供的功能强大的应用工具。使用Oracle核心（常规路径装载）或使用特别快速的数据装载（直接路径装载），SQL*Loader能够从几乎全部的数据格式中装载几乎全部的数据类型。SQL*Loader消除了编写装载特殊数据类型程序的需要。