# 文本导出工具

文本方式有什么作用？文本方式是不同类型数据库间进行数据交换的重要方式!

将数据导出成文本文件，在其他数据库（如 Sybase、SQL Server、MySQL）中看起来是一件很容易的事，因为有相应的工具（Bulk Copy，简称 bcp）或命令(SELECT INTO)。Oracle 既没有这样的工具，也没有提供这样的命令，因此经常看到论坛上有人在问如何将 Oracle 的数据快速导出成文本文件。从大学毕业从事 IT 行业开始到现在，在许多项目中也都遇到了这样的需求，使用过数种不同的编程语言(Java、PowerBuilder、Perl 等)来实现过这个功能，但以前已经实现的每一种都不是最理想的，始终有些性能上的缺憾，不能胜任上千万条记录，甚至上亿条记录的场合。

刚毕业时当程序员，第一年用 Developer 2000，面对的所有数据库都是 Oracle，没有文本导出要求，学会了 Oracle 的导入导出（exp/imp）功能。第二年用 Power Builder，面对的数据库就有 Oracle 和 Sybase，就遇到过文本导出的需求，用 PowerBuilder，面对的数据库有 Oracle 和 Sybase，就遇到过文本导出的需求，用 Power Builder 的数据管道实现过，都是结合在具体项目中使用，单独用的很少，也不方便，为了一个文本导出功能需要一个很大的运行环境，并且速度也不快。后来 Java 流行，用 JDBC 实现了一个，算是比较轻量级的了，基本上 Java 的运行环境哪儿都有，在测试速度时，发现与 Oracle Call Interface（简称 OCI）程序比要达到同样的速度，运行 Java 程序的机器上多耗了四倍的 CPU，也就是用 Java 版本的文本导出程序常常让客户端的 CPU 跑满，因此无法面对更大量（上 GB）的数据导出。

开始做专职的 DBA 后，一开始也在寻找能实现快速文本导出的好工具，和许多 DBA 一样在 asktom.oracle.com 上找到了 Tom Kyte 写的两个脚本，一个是用 SQL*Plus 的 spool 功能来实现的，另一个是用 Pro*C 语言来写的程序。用 spool 实现的不是很喜欢，第一速度不快，第二个不能指定任意字段分隔符，及记录分隔符。Pro*C 版本的在性能上很好，但在灵活性上，还是比较差，要实现当时想到的需求，一样要改源程序，并且用 Pro*C 的程序在编译后很难跨版本通用，而这一点我比较看重，于是就想到以这个高效程序为样本，写一个更灵活通用的文本导出程序。

刚好对 C 语言及 Oracle Call Interface 产生了一些兴趣，就拿这个任务来练手吧，这个程充的名字为 SQL * UnLoader，刚好和 Oracle 用于文本导入的工具 SQL * Loader 相对应。没

**批注 [xdx1]:** 头两章，两个工具的介绍，是不是可以只介绍主要功能和用途，把具体的使用方法放到书后作为附录？和作者确认。
补：这里主要要表示的是那一类的工具，可以帮到 DBA 进行工作，并不是真的一定要用这两个工具，其他人也可以根据这样的需求去开发类似的工具。

**批注 [baiaiping2]:** 全章没有章序号，也没有节序号。建议作者加上。
补：章节定后，再来加上

**批注 [lispython3]:** 放在全章的开头来说

**批注 [lispython4]:** 加脚注把这个工具介绍一下。
补：这个工具，SQL Server 和 Sybase 等都有，作为有数据库经验的，应当不是很难理解。

有想到的是，为了这么一个小工具，前前后后五六年了，还在不停地维护，根据不同用户提出的各种需求来完善程序，是当时没有预料到的事情。当然在整个过程中，得到了很多人的感谢邮件，而我个人也体会到了坚持的乐趣，也让我在更多的事情上多一分坚持。

## SQL*UnLoader 简介

Sqluldr2（SQL * UnLoader 第二版）是灵活与强大的 Oracle 文本导出程序，已被大众使用许多年了，有上千个国内外 DBA 在使用它或使用过它，并在使用的过程中提出了宝贵实用的改进需求，在完善了众多真实客户的需求后，最终形成了现在的稳定版本。Oracle 有一个工具叫 SQL*Loader(sqlldr)是用来将文本文件装载到数据库中的，而这个工具是用来将 Oracle 中的数据导出成文本的，因此取名为 SQL*UnLoader(sqluldr)，而最后的 2 是因为第一版是用 OCI 7 接口写的，而现在发布的是用 OCI 8 接口重写的，是第二代的意思。学会这样的好工具，在遇到数据迁移需求时，可以让 DBA 轻松自在地完成任务。

从小的地方说，例如将一些 Oracle 性能数据收集到 MySQL 中，进行分析和展示；从大的地方讲，sqluldr2 曾被用来在 Oracle 系统中导出大批量的数据，然后装载到数据仓库（NCR 或 Sybase IQ）中，参与了众多数据仓库项目的 ETL 工作。阿里巴巴（Alibaba）集团的 DBA 几年前就开始用 sqluldr 从生产系统中导出巨量数据给数据仓库分析，并在真实应用环境下创下了我所见过的文本导出速度的最高记录，总共用了 595 秒钟导出了 171135273 条记录，平均每秒 287622 条记录。

```
0                 rows exported at 2007-11-28 15:43:09
171135273 rows exported at 2007-11-28 15:53:04
```

或者用在不同字符集的同种数据库之间进行数据交换，如 US7ASCII 字符集的 Oracle 数据库和 ZHS16GBK 字符集的 Oracle 数据库进行中文信息交换，通过 Database Link 会比较复杂。在某些系统中为了防止软件版本变迁的问题，选择用最简可靠的文本方式来归档和永久保存历史数据。最近也有不少人用 sqluldr2 来快速将数据导出成文本，为 GreenPlum 这样的海量数据分析处理系统提供数据。也有做搜索的，结合 sqluldr 的早期源代码，结合到商品搜索引擎的数据 dump 程序中。也有人整合 sqluldr 的早期源代码到普通数据库与内存数据库的数据同步程序中。

更有国外的工程师，发邮件来谢谢我免费发布这个小工具，说这个工具被成功地用于某

**批注 [baiaiping5]:** 这个是作者写的那个工具的名字吗？前文没有说明，建议指明。
补：前面提了一下工具的名字。

**批注 [baiaiping6]:** 什么场合，建议扩展内容。
补：数据迁移

些项目中，帮他们解决了数据迁移的问题。从个人网站的统计来看，源代码和预编译的可执行文件的累计下载量已经突破了 20000 大关。

Sqluldr2 工具的主要优点如下：

- 用 Oracle 的 C 语言接口写成，短小精干，运行速度快。
- 可指定任意字段分隔符与记录分隔符，分隔符不一定是可见的字符。
- 可自动生成 SQL * Loader 的控制文件，方便在其他 Oracle 库上进行导入。
- 丰富的性能调整及功能选项，简化了调优任务。
- 可根据运行返回值确定是否成功执行导出任务，有利于用脚本来处理。
- 用标准 C 语言写成，多个平台编译执行，如 Windows/Linux/AIX/Solaris/HPUX 等。
- 直接导出数据到 GZIP 格式的压缩文件中，降低磁盘空间，节约成本。
- 口令加密功能，降底脚本调度中的安全风险。
- 将数据导出成 MySQL 或 Oracle 数据库上的 Insert 语句。

当然 sqluldr2 也有缺点，曾经有人写邮件问，为什么鼠标双击，只看见窗口一闪，就退出了，那是因为这个程序不是图形界面(GUI)的。对于大部份终端用户来讲，没有向导的非图形化命令行工具，会显得不太好用（有国内用户提了这个需求），但不同平台下的图形编程环境不一样，要支持不容易。但这刚好成就了两个优点，既可以在多种平台编译和执行（Windows，Linux，AIX，Solaris，HPUX 等），又容易嵌入在其他的脚本中，如 Shell 或 Perl 中，或其他编程语言（如 Java，来自国内的感谢邮件中看到的）中，只要用几次后，就会习惯它了。

访问如下连接获得部份平台的可执行程序和早期源代码（可参考编写 OCI 程序）：

- 最新软件下载：http://www.anysql.net/software/sqluldr.zip
- 早期源码下载：http://www.anysql.net/software/ociuldr.c

## 使用方法

进入到命令行中，如 Windows 的 DOS 命令窗口，或 Linux/Unix 的 Shell 窗口，通过各种命令行参数来告诉 sqluldr2 要做什么，不带任何参数运行 sqludlr2 就可以获得命令行的帮助。默认情况下会显示比较精简的命令行选项，如下所示：

```
SQL*UnLoader: Fast Oracle Text Unloader (GZIP), Release 3.0.1
(@) Copyright Lou Fangxin (AnySQL.net) 2004 - 2010, all rights reserved.

Usage: SQLULDR2 keyword=value [,keyword=value,...]

Valid Keywords:
   user    = username/password@tnsname
   sql     = SQL file name
   query   = select statement
   field   = separator string between fields
   record  = separator string between records
   rows    = print progress for every given rows (default, 1000000)
   file    = output file name(default: uldrdata.txt)
   log     = log file name, prefix with + to append mode
   fast    = auto tuning the session level parameters(YES)
   text    = output type (MYSQL, CSV, MYSQLINS, ORACLEINS, FORM, SEARCH).
   parfile = read command option from parameter file

  for field and record, you can use '0x' to specify hex character code,
  \r=0x0d \n=0x0a |=0x7c ,=0x2c, \t=0x09, :=0x3a, #=0x23, "=0x22 '=0x27
```

要显示所有的命令行选项，需要输入“sqluldr2 help=yes”。最少需要传入两个参数，数据库登录信息和 SQL 查询语句，例如：

批注 [baiaiping7]: 似乎有点罗嗦，建议改为：要显示所有的命令行选项，需要输入“sqluldr2 help=yes”。

```
D:\>sqluldr2 scott/tiger query="select * from emp"
       0 rows exported at 2009-11-10 09:20:49, size 0 MB.
      12 rows exported at 2009-11-10 09:20:49, size 0 MB.
         output file uldrdata.txt closed at 12 rows, size 0 MB.

D:\>sqluldr2 scott/tiger query=emp
       0 rows exported at 2009-11-10 09:20:55, size 0 MB.
      12 rows exported at 2009-11-10 09:20:55, size 0 MB.
         output file uldrdata.txt closed at 12 rows, size 0 MB.
```

第一个导出命令通过指定 SELECT 语句来实现，如果是整表导出，则可以很方便地只指定一个表名，如第二条命令所示。

要使用其基本功能并不复杂，只要两个参数就可以了。但工具中出现的这么多的命令行选项是很多人使用后提出来的需求的总结，可以很好地满足工们各种各样的 Oracle 数据导出到文本的需求。

## 命令行选项

命令行参数的基本格式如下：

```
sqluldr2 prop1=value1 prop2=value2 ……
```

众多的命令行参数中，提供了比较多的功能，可能觉得记不住，但还是有规律可以依的，可以分为以下几类：

- 快捷选项

  用来快速设置一些常用的功能的选项，可以大大方便人们使用。

- 数据库登录选项

  指定目标数据库的登录信息：用户名和密码。或用于生成加密后的登录信息，或是指定给定的连接信息是否加密的。

- 数据来源选项

  指定 SQL 语句，或指定包含 SQL 的文件，以及 long 类型列导出的最大长度。

- 格式控制选项

  指定生成文本文件的字段分隔符，记录分隔符，文本文件的名字，是否分成多个文件导出，以及是否在文本文件的第一行打印字段名。

- 性能调优选项

  一些性能参数，如批量读出的大小，以及控制会话级 sqlldr 数据库连接的性能优化参数，如连续读取时的数据块数量、排序区的大小等。

- sqlldr 设置选项

  SQL*Loader 是 Oracle 的一个工具，可以将文本装载到数据库中。这里面的参数可以控制生成 sqlldr 导入工具用的参数文件。

- 使用参数文件

  用于指定参数设置文件位置。

- MySQL 相关选项

  在迁移数据到 MySQL 的实践中，发现需要处理特殊字符及 NULL 值，才能方便地导入数据到 MySQL 中。

有些命令行选项相当简单，有些则稍微复杂，还是来看更详细的介绍吧。需要多练习练习，才能熟练运用。

## 快捷选项

在 sqluldr2 的众多命令行选项中，有些选项是经常组合起来用的，以方便用户使用，有了这些选项，就不再需要在命令行输入很多的参数，而用一个选项来代替多个选项的设置。

主要有两个选项：

| 选项 | 格式 | 作用 |
| --- | --- | --- |
| fast | yes | 自动设定会话级数据库调优参数，比如 db file multiple block read, sort area size, hash area size 等。 |
| text | mysql，csv，mysqlins，oracleins，form | 用于控制导出的文本文件的格式。 |

批注 [baiaiping8]: 图表要有序号及图表标题。全章统一补充。如“表 1-1 表名”。图和表分别排序。
补：这个所有表格的序号是否必须？

比如 CSV 文件格式，则第一行显示为字段名，非数字段值用双引号引起，以登录分隔等等，要生成 CSV 文件，则只需要简单地使用如下命令。 比如 CSV 文件格式，则第一行显示为字段名，非数字段值用双引号引起，以逗号分隔等等，要生成 CSV 文件，则只需要简单地使用如下命令。

```
sqluldr2 scott/tiger query=emp text=csv
```

生成的文件如下所示：

```
EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO
7369,"SMITH","CLERK",7902,"1980-12-17 00:00:00",800,,20
7499,"ALLEN","SALESMAN",7698,"1981-02-20 00:00:00",1600,300,30
7521,"WARD","SALESMAN",7698,"1981-02-22 00:00:00",1250,500,30
7566,"JONES","MANAGER",7839,"1981-04-02 00:00:00",2975,,20
7654,"MARTIN","SALESMAN",7698,"1981-09-28 00:00:00",1250,1400,30
7698,"BLAKE","MANAGER",7839,"1981-05-01 00:00:00",2850,,30
7782,"CLARK","MANAGER",7839,"1981-06-09 00:00:00",2450,,10
7839,"KING","PRESIDENT",,"1981-11-17 00:00:00",5000,,10
7844,"TURNER","SALESMAN",7698,"1981-09-08 00:00:00",1500,0,30
7900,"JAMES","CLERK",7698,"1981-12-03 00:00:00",950,,30
7902,"FORD","ANALYST",7566,"1981-12-03 00:00:00",3000,,20
7934,"MILLER","CLERK",7782,"1982-01-23 00:00:00",1300,,10
```

如果没有快捷选项，则需要指定很多的命令行选项。不使用快捷选项时需要增加如下 5 个命令行参数：

```
head=yes quote=0x22 escape=0x22 field=, record=0x0a
```

CSV 文件是我们常用的文本格式，可以直接被 Excel 电子表格软件打开。其他的一些快捷选项会在后面陆续讲到。

## 数据库登录选项

Sqluldr2 不支持交互式提示输入用户名和密码，因此在运行 sqluldr2 程序时如果用 ps 去查看进程或在系统日志中查看执行过的命令，是可以看到用户名和密码的。为了安全，如果数据库的版本低于 9i，可以指定用 sys 用户登录；或者在 Oracle 上创建操作系统验证的用户，不需要输入密码就可以登录进行导出；或者使用登录加密功能，在脚本中指定加密后的连接信息；也可以将连接信息写在参数文件中，然后在命令行中指定参数文件的位置，以防在 ps 信息中看到密码信息，造成安全漏洞。

和登录有关的有两个命令行选项：

| 选项 | 格式 | 作用 |
|---|---|---|
| user | user/pass@tns | 指定登录到特定的用户名。 |
| crypt | YES 或 NO | 只做口令加密工作，而不真正导出数据。 |

使用方法如下，第一行以 SYS 用户登录（不需要密码），第二行以操作系统验证方式登录，第三行指定用户名及口的方式。

```
sqluldr2  user=sys ……
sqluldr2  user=/  ……
sqluldr2  user=scott/tiger@test ……
```

在使用 Oracle 的工具，如 exp/imp 时，输入登录信息时，并不需要指定选项的名字，为了保持一样的习惯，当登录信息是第一个命令行选项时，可以不指定选项名称。如下所示：

```
sqluldr2  sys ……
sqluldr2  /  ……
sqluldr2  scott/tiger@test ……
```

直接指定反斜杆为登录信息时，是指用操作系统难证的用户进行登录，如何建这样的用户，可以参考有关 Oracle 用户管理的文档，在大部份 Oracle 管理的图书中都有详尽的解释

和说明，在 DBA 的学习过程中必然会学到这一点。

以 SYS 用户登录或以操作系统验证方式登录一般都要求在数据库服务器上运行，但实际上，更多的导出不是在数据库服务器上执行的，而是在远程客户端机器上执行的，这就要求提供连接信息，包括用户名和密码。这样就牵涉到了数据库安全的问题，将连接信息放在严格权限控制的参数文件（后面会介绍）中，也不是提高安全性的最佳做法，这时可以使用 sqluldr2 提供的连接信息加密功能。

首先是对用口令进行加密，得到 sqluldr2 加密后的字符串，给 sqluldr2 指定连接信息，并指定“crypt”选项值为“YES”，如下所示：

```
D:\>sqluldr2 user=scott/tiger crypt=yes
user=4899919fa603950bfafcb74dd6616903
```

得到了加密后的字符串，接下来使用时，用加密后的字符串来提供连接信息，如下所示：

```
D:\>sqluldr2 4899919fa603950bfafcb74dd6616903 query=emp
       0 rows exported at 2009-11-10 09:49:57, size 0 MB.
      12 rows exported at 2009-11-10 09:49:57, size 0 MB.
         output file uldrdata.txt closed at 12 rows, size 0 MB.
```

这个安全需求，使得 sqluldr2 可以被用于关键场合。现在越来越多的企业对系统的安全，尤其是数据库的安全，提出了更高的要求，对于脚本或程序中的密码保护十分重视，而 sqluldr2 是一个附合企业安全需求的 DBA 工具。

## 数据来源选项

要导出数据就必须指定一个 SQL 语句，作为导出的数据来源，就算是全表导出，也要写成一个 SQL 语句；另外 sqluldr2 并不完全支持 long 类型，只能导出指定长度以内的 long 类型数值。有四个选项和这个两个能功有关。有三个选项和这个功能有关。

| 选项 | 格式 | 作用 |
|---|---|---|
| query | “select …” | 指定一个 SQL 语句，通常用双引号括直来。 |
| sql | 文件名 | 指定包括 SQL 语句的文本文件名。 |
| long | 字节数 | 指定导出 long 类型时的最大长度，最大 32K。 |

“query”选项用于将整个 SQL 语句写在命令行，适合于比较简单的 SQL 语句；而“sql”选项则指定一个包括 SQL 语句的文本文件，适合于比较复杂的 SQL 语句。需要注意的是在

SQL 语句的最后不要加分号。long 选项用于控制读取 long 类型数据的最大长度，在编写程序时，为了简单，就没有提供对 long 类型的全部支持，因为 long 类型不能实现 Array Fetch，必须要一条一条处理，对于一个 long 类型的值可能要分能多次 Fetch，有一定的复杂性，另外考虑 long 类型的使用情况不多，就打折处理了。最大可指定的长度为 1MB，默认是 4000。

使用 query 参数在命令行指定一个简单的 SELECT 语句：

```
D:\>sqluldr2 scott/tiger query="select * from emp"
       0 rows exported at 2009-11-10 09:20:49, size 0 MB.
      12 rows exported at 2009-11-10 09:20:49, size 0 MB.
         output file uldrdata.txt closed at 12 rows, size 0 MB.
```

如果要导出一整个表的数据，简单起见，也可以直接在 query 中指定表名，如下所示：

```
D:\>sqluldr2 scott/tiger query=emp
       0 rows exported at 2009-11-10 09:20:55, size 0 MB.
      12 rows exported at 2009-11-10 09:20:55, size 0 MB.
         output file uldrdata.txt closed at 12 rows, size 0 MB.
```

如果 SQL 语句很复杂或很长，则很难在命令行指定，这时可以通过编写一个 SQL 文件，文件中包含复杂的 SQL（要注意的是，SQL 语句的最后不要有分号或反斜杆），然后告诉 sqluldr2 从指定 SQL 文件中读取复杂的 SQL 语句，进行数据导出，如下所示：

```
D:\>cat emp.sql
select
  *
  from emp
D:\>sqluldr2 scott/tiger sql=emp.sql

       0 rows exported at 2009-11-10 09:20:55, size 0 MB.
      12 rows exported at 2009-11-10 09:20:55, size 0 MB.
         output file uldrdata.txt closed at 12 rows, size 0 MB.
```

在 Linux 或 Shell 中，使用文件指定复杂的 SQL 还是比较麻烦，我们还可以让 sqluldr2 从标准读入设备读取 SQL 语句，只需要指定“SQL”选项的值为减号（“-”）即可。例如，下面的 Shell 脚本：

```
$ cat test.sh
#!/bin/sh

sqluldr2 sys sql=- file=- <<EOF
select
  *
  from emp
EOF
```

可以将复杂的 SQL 语句，直接写到脚本中，而不需要创建额外的 SQL 文件，可以使脚本变得更加通用。

## 格式控制选项

固定格式的文本需要能区分出不同的字段和记录，要区分不同的字段，就要在两个字段的值之间加入特定的符号进行区分，如逗号或竖线，只要这些值不会出现在字段值中，就可以用来区分；要区分不同的记录，就需要在两条记录之间插入特定的符号进行分隔，如换行符，如果在字段值中没有换行符，那么在文本文件中的一行就表示一条记录。例如很多人熟悉的用 Excel 可以直接打开的 CSV 文件，如果字段值中不会出现逗号和换行符，简单地说其实就是用逗号分隔字段，用换行符分隔记录的文本文件。例如：

```
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,,10
7788,SCOTT,ANALYST,7566,1987-04-19 00:00:00,3000,,20
7839,KING,PRESIDENT,,1981-11-17 00:00:00,5000,,10
……
```

或者不用分隔符来区分字段，使每个字段都导出成固定长度的值，如下所示：

```
7369SMITH      CLERK     7902 1980-12-17 00:00:00 800            20
7499ALLEN      SALESMAN  7698 1981-02-20 00:00:00 1600   300    30
7521WARD       SALESMAN  7698 1981-02-22 00:00:00 1250   500    30
7566JONES      MANAGER   7839 1981-04-02 00:00:00 2975            20
7654MARTIN     SALESMAN  7698 1981-09-28 00:00:00 1250   1400   30
7698BLAKE      MANAGER   7839 1981-05-01 00:00:00 2850            30
7782CLARK      MANAGER   7839 1981-06-09 00:00:00 2450            10
7788SCOTT      ANALYST   7566 1987-04-19 00:00:00 3000            20
7839KING       PRESIDENT      1981-11-17 00:00:00 5000            10
……
```

有四个选项可以用分别用来指定字段分隔符和记录分隔符：

| 选项 | 格式 | 作用 |
|---|---|---|
| field | 分隔符 | 指定字段分隔符，默认为逗号 |
| record | 分隔符 | 指定记录分隔符，默认为回车换行，Windows 下的换行 |
| quote | 引号符 | 指定非数字字段前后的引号符 |
| Head | Yes 或 No | 指定第一行是否输出列名，默认 No |

例如现在要改变默认的字段分隔符，用“#”来分隔记录，导出的命令如下所示：

```
sqluldr2  scott/tiger sql=emp.sql field=#
```

导出的文本中的内容如下所示：

```
7369#SMITH#CLERK#7902#1980-12-17 00:00:00#800##20
7499#ALLEN#SALESMAN#7698#1981-02-20 00:00:00#1600#300#30
7521#WARD#SALESMAN#7698#1981-02-22 00:00:00#1250#500#30
7566#JONES#MANAGER#7839#1981-04-02 00:00:00#2975##20
7654#MARTIN#SALESMAN#7698#1981-09-28 00:00:00#1250#1400#30
7698#BLAKE#MANAGER#7839#1981-05-01 00:00:00#2850##30
7782#CLARK#MANAGER#7839#1981-06-09 00:00:00#2450##10
7788#SCOTT#ANALYST#7566#1987-04-19 00:00:00#3000##20
7839#KING#PRESIDENT##1981-11-17 00:00:00#5000##10
……
```

字段分隔符虽然通常是一个字符，但实际上不一定是只有一个字符，可以用多个字符；并且也不是只能使用可见的或可以打印的字符或字符串来作分隔符，不可见的字符也可以。有些字符虽然是可见或可以打印的，但在操作系统中有特殊的用处，如管道符（|）或与符号（&），想用它们来作分隔符时，直接传入字符当参数时，命令行就要特定处理，在这些字符前面加转义字符。为了可以使用不可见的字符及方便命令行参数传入，在指定分隔符时，

可以用字符的 ASCII 代码（0xXX，大写的 XX 为 16 进制的 ASCII 码值）来指定一个字符，常用的字符的 ASCII 代码如下：

| 字符 | ASCII 代码 |
|---|---|
| 回车 | 0x0d |
| 换行 | 0x0a |
| TAB 键 | 0x09 |
| \| | 0x7c |
| & | 0x26 |
| 双引号 | 0x22 |
| 单引号 | 0x27 |

下面是用 ASCII 码来指定不同字段分隔符的例子：

```
Sqluldr2  scott/tiger sql=emp.sql field=0x09
Sqluldr2  scott/tiger sql=emp.sql field=0x7c
Sqluldr2  scott/tiger sql=emp.sql field=0x26
```

也可以指定多个 ASCII 值来作为分隔符，例如：

```
Sqluldr2  scott/tiger sql=emp.sql field=0x090x09
sqluldr2  scott/tiger sql=emp.sql field=0x7c0x7c
sqluldr2  scott/tiger sql=emp.sql field=0x260x26
```

比如我在导出 GBK 数据库时，常用不可见字符 0x07 作为字段分隔符，用另一个不可见字符 0x06 作为记录分隔符，因为在 GBK 字符集时，这两个值很难作为字段的值被录入进去，这样的话可以处理记录值中有换行的情况。在选择分隔符时，一定不能选择会在字段值中出现的字符组合，如常见的单词等，很多次导入时报错，回过头来找原因时，都发现是因为分隔符出现在字段值中了。

有时我们需要在非数字字段值的前后加上一个双引号或单引号，这个可以通过指定“QUOTE”选项来实现，只能选择一个字符来作为这个选项的值。如下所示：

```
D:\>sqluldr2 scott/tiger query=emp quote=0x22
       0 rows exported at 2009-11-10 17:46:36, size 0 MB.
      12 rows exported at 2009-11-10 17:46:36, size 0 MB.
         output file uldrdata.txt closed at 12 rows, size 0 MB.

D:\>type uldrdata.txt
7369,"SMITH","CLERK",7902,"1980-12-17 00:00:00",800,,20
7499,"ALLEN","SALESMAN",7698,"1981-02-20 00:00:00",1600,300,30
7521,"WARD","SALESMAN",7698,"1981-02-22 00:00:00",1250,500,30
7566,"JONES","MANAGER",7839,"1981-04-02 00:00:00",2975,,20
7654,"MARTIN","SALESMAN",7698,"1981-09-28 00:00:00",1250,1400,30
7698,"BLAKE","MANAGER",7839,"1981-05-01 00:00:00",2850,,30
7782,"CLARK","MANAGER",7839,"1981-06-09 00:00:00",2450,,10
7839,"KING","PRESIDENT",,"1981-11-17 00:00:00",5000,,10
7844,"TURNER","SALESMAN",7698,"1981-09-08 00:00:00",1500,0,30
7900,"JAMES","CLERK",7698,"1981-12-03 00:00:00",950,,30
7902,"FORD","ANALYST",7566,"1981-12-03 00:00:00",3000,,20
7934,"MILLER","CLERK",7782,"1982-01-23 00:00:00",1300,,10
```

当以指定空格（0x20）作为分隔符时，sqluldr2 会生成固定长度格式的文本文件，sqlldr 在导入固定长度格式的文本文件时，性能可以提升数倍。生成的字段的长度由 sqluldr2 根据字段的类型及表结构定义来决定，不能手工设定。

```
sqluldr2 scott/tiger query="select ename, job, hiredate from emp" field=0x20
table=emp
```

检查一下生成的文件（默认文件 uldrdata.txt）中的内容：

```
SMITH     CLERK    1980-12-17 00:00:00
ALLEN     SALESMAN 1981-02-20 00:00:00
WARD      SALESMAN 1981-02-22 00:00:00
JONES     MANAGER  1981-04-02 00:00:00
MARTIN    SALESMAN 1981-09-28 00:00:00
BLAKE     MANAGER  1981-05-01 00:00:00
CLARK     MANAGER  1981-06-09 00:00:00
KING      PRESIDENT1981-11-17 00:00:00
TURNER    SALESMAN 1981-09-08 00:00:00
……
```

通过 TABLE 选项来自动生成控制文件（文件名：emp_sqlldr.ctl），就可以知道生成的数据文件中每个列的宽度了：

```
--
-- SQL*UnLoader: Fast Oracle Text Unloader (GZIP), Release 3.0.0
-- (@) Copyright Lou Fangxin 2004/2009, all rights reserved.
--
OPTIONS(BINDSIZE=2097152,READSIZE=2097152,ERRORS=-1,ROWS=50000)
LOAD DATA
INFILE 'uldrdata.txt' "FIX 39"
INSERT INTO TABLE emp
FIELDS TERMINATED BY X'20' TRAILING NULLCOLS
(
  ENAME POSITION(1:10) CHAR(10) NULLIF ENAME=BLANKS,
  JOB POSITION(11:19) CHAR(9) NULLIF JOB=BLANKS,
  HIREDATE  POSITION(20:38)  DATE  "YYYY-MM-DD  HH24:MI:SS"  NULLIF
HIREDATE=BLANKS
)
```

只要指定的分隔符中第一个字符是空格（0x20）就会生成固定长度格式。刚推出不久，就有网友建议再增加一个选项来指定每个字段的长度，因为有时数据库中的列定义可能比真实值大很多，自动生成时就多了很多不必要的空格。

有网友要求在文本文件的第一行能输出字段名，以方便在导入到Excel中时能区分字段。如果我们要用文本方式来长期保存历史数据，最好也在文件的第一行加上字段名，以免时间长了，不知道各个列的函义。这是第一个来自外部的用户需求，由淘宝网的DBA提出。可以用head选项来实现，如下所示：

```
sqluldr2  scott/tiger sql=emp.sql field=, head=yes
```

来看一下生成的文件中的内容：

```
EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,,10
……
```

在客户端字符集与服务器端字符集完全相同时，sqluldr2可以准确地获得数据的长度，从而在客户端分配合适的内存。但如果字符集设置不一样，比如我们从中文字符集

（ZHS16GBK）的数据库中，以 UTF8 的字符集导出成 UTF8 编码格式的文件，就有可能出现 ORA-24345 错误，表示导出的文本中的字符串的长度大于程序中声明的长度。

```
ORA-24345: A Truncation or null fetch error occurred
Cause: A truncation or a null fetch error″

Action: Please ensure that the buffer size is long enough to store the returned
data.
```

遇到这种情况，有两个参数可以用来控制每个列的客户端缓冲区的长度：

| 选项 | 格式 | 作用 |
|---|---|---|
| Safe | (Yes\|No) | 如果指定为 Yes，则客户端缓冲区加倍，称为安全模式。 |
| width | 各列长度 | 用冒号分隔的各列的客户端缓冲区长度。 |

比如我们运行在安全模式，会将所有的字符字段的缓冲约加倍，例如：

```
sqluldr2 scott/tiger query=″select ename, job, hiredate from emp″ field=0x20
table=emp safe=yes
```

我们来看一下生成的文件，同前面生成的固定长度的文本文件相比，发现字段间距加大了：

```
SMITH               CLERK               1980-12-17 00:00:00
ALLEN               SALESMAN            1981-02-20 00:00:00
WARD                SALESMAN            1981-02-22 00:00:00
JONES               MANAGER             1981-04-02 00:00:00
MARTIN              SALESMAN            1981-09-28 00:00:00
BLAKE               MANAGER             1981-05-01 00:00:00
CLARK               MANAGER             1981-06-09 00:00:00
KING                PRESIDENT           1981-11-17 00:00:00
TURNER              SALESMAN            1981-09-08 00:00:00
JAMES               CLERK               1981-12-03 00:00:00
FORD                ANALYST             1981-12-03 00:00:00
MILLER              CLERK               1982-01-23 00:00:00
```

或者我们可以手工控制每个列的长度，如下所示：

```
sqluldr2 scott/tiger query=″select ename, job, hiredate from emp″ field=0x20
table=emp width=12:12:20
```

重新来看一下生成的文本文件，注意字段间的间距，如下所示：

```
SMITH       CLERK       1980-12-17 00:00:00
ALLEN       SALESMAN    1981-02-20 00:00:00
WARD        SALESMAN    1981-02-22 00:00:00
JONES       MANAGER     1981-04-02 00:00:00
MARTIN      SALESMAN    1981-09-28 00:00:00
BLAKE       MANAGER     1981-05-01 00:00:00
CLARK       MANAGER     1981-06-09 00:00:00
KING        PRESIDENT   1981-11-17 00:00:00
TURNER      SALESMAN    1981-09-08 00:00:00
JAMES       CLERK       1981-12-03 00:00:00
FORD        ANALYST     1981-12-03 00:00:00
MILLER      CLERK       1982-01-23 00:00:00
```

手工指定各列的长度，有助于我们美化输出，系统中定义的字段的长度比真实的值大很多，如果按声明的长度去显示，一行会太长，会让输出很难看。

## 输出文件选项

导出的数据会存放到磁盘文件上，如果一个文件太大，则可以生成多个文件，在生成数据文件的同时，默认设置下，每导出 100 万条记录都会输出日志信息，日志信息可以输出到屏幕，也可以输出到文件，和输出文件有关的有四个选项：

| 选项 | 格式 | 作用 |
|---|---|---|
| file | 数据文件名 | 指定数据文件 |
| batch | YES 或 NO | 为每个批次（一个批次 100 万条记录）生成一个文件 |
| size | 大小（单位：MB） | 控制生成的文本文件的大小，以输出到多个文件 |
| log | 日志文件名 | 指定日志文件 |

导出的数据会存放到磁盘文件上，如果一个文件太大，则可以生成多个文件，在生成数据文件的同时，默认设置下，每导出 100 万条记录都会输出日志信息，日志信息可以输出到屏蔽，也可以输出到文件，和输出文件有关的有三个选项：

file 选项指定导出的文件名，如果文件面的第一个字符为加号（+），则表示在现有的文件后面添加，否则会生成新文件，如果已经有同名文件则覆盖现有文件，如果没有指定“FILE”选项则默认为小写的“**uldrdata.txt**”，存放路径为运行 sqluldr2 的当前目录。例如：

```
sqluldr2  scott/tiger query=”select * from emp”  file=emp.txt ……
sqluldr2  scott/tiger sql=emp.sql file=+emp.txt ……
```

“file”选项中还有几个很特别的格式码，可以用来指定动态文件名（使生成的文件名中带日期信息），合理使用这些格式码，可以满足一些特定的需求，减少脚本编写的工作量，如下表所示：

| 格式码 | 作用 | 例子 |
|---|---|---|
| %Y、%y | 表示当前时间的四位年份 | File=A%y%m%d.txt |
| %M、%m | 表示当前时间的两位月份 | |
| %D、%d | 表示当前时间的两位日份 | |
| %W、%w | 表示当前时间的一星期中的天数 | File=A%w.txt |
| %B、%b | 在多个文件导出时，表示文件序号 | File=A%b.txt |

批注 [baiaiping9]: 建议改为“格式码”，与前面表述一致

当要导出的记录数很多达上亿条时，将所有文件导出成一个文件可能太大，不方便其他程序处理，因此有网友提出了导出到多个文件的功能需求，这个需要通过指定“batch”选项来实现，一个批次的记录表示 100 万条记录（由 ROWS 参数决定），而“batch”选项用于指定是否为每个批次的记录生成独立的文本文件，如指定“YES”则表示每个批次分成一个文件。指定这个选项时，在“file”中请包含“%b”格式码（会生成从 1 开始的批次号）。例如：

批注 [baiaiping10]: 表格中列出的“log”选项，是放到最后来讲的，建议调整表格中选项的位置。
补：已放到后面。

```
Sqluldr2 scott/tiger sql=emp_his.sql file=emp_%b.txt batch=yes ……
```

在做这个试验时，向 emp 表中重复插入了 300 多万条记录，屏幕上的输出信息如下：

```
      0 rows exported at 2009-03-22 16:38:44
 500000 rows exported at 2009-03-22 16:38:48
1000000 rows exported at 2009-03-22 16:38:52
        output file emp_1.txt closed at 1000000 rows.
 500000 rows exported at 2009-03-22 16:38:56
1000000 rows exported at 2009-03-22 16:39:01
        output file emp_2.txt closed at 1000000 rows.
 500000 rows exported at 2009-03-22 16:39:06
……
```

这个例子中，一共生成了四个文件，单个文件最多 100 万条记录，生成多个文件可以方便其他程序并行来处理这些数据，提高后续处理速度。例如，GreenPlum 中就要求将大的数

据文件拆分成小的，以便不同的机器进行并行处理。

也可以通过“size”参数来控制每个生成的文件的大小，来生成多个文件，比如指定“SIZE”为 20 则表示，每个生成的文件尽量接近 20MB，超过 20MB 后就生成一个新的文件。同“BATCH”选项一样，指定“SIZE”选项时，在“FILE”中请包含“%b”格式码（会生成从 1 开始的批次号）。

```
Sqluldr2 scott/tiger sql=emp_his.sql file=emp_%b.txt size=20 ……
```

当文件名（“FILE”选项）的后缀以小写的“.gz”结尾时，会将记录直接写入到 GZIP 格式的压缩文件中，如果要归档大量数据，这个功能可以节约很多的存贮空间，降低运营成本。例如：

```
D:\>sqluldr2 sys query=dba_objects file=test.txt.gz

       0 rows exported at 2009-11-10 11:39:32, size 0 MB.
    9850 rows exported at 2009-11-10 11:39:32, size 0 MB.
         output file test.txt.gz closed at 9850 rows, size 1 MB..
```

用解压软件解压这个文件，生成“test.txt”文件，然后比较一下两个文件的大小差异：

```
2009-11-10  03:39         1,080,175 test.txt
2009-11-10  11:39            90,085 test.txt.gz
```

可以看到原来 1MB 大小的文件，以 GZIP 压缩方式生成时，大小才 90KB，如果用于归档历史数据，可以节约不少空间，GZIP 目前是性价比非常好的压缩方式之一。

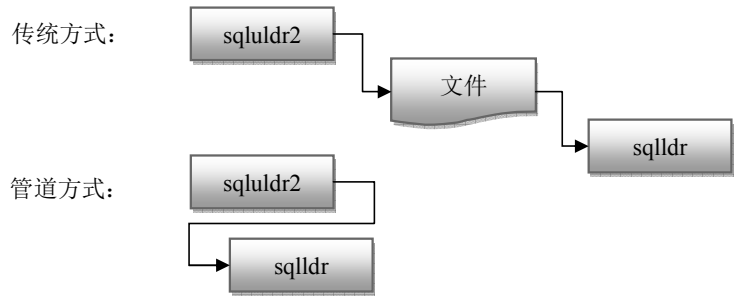
2008 年时，一个外国的网友在使用 sqluldr2 的过程中，向我提出能不能将记录输出到标准输出设备，以便利用 Linux/Unix 上强大的管道（pipe）功能，一来可以节约处理时间，省去生成文本文件这一步，另一方面也不需空间来临时保存一下文件了。这个功能的确是十分重要的，以至于我在接到需求后，放下手边其他的工作，马上去完善 sqluldr2 了。

我们来显示一下“EMP”这个表的内容吧，只需要将输出文件名指定为减号(“-”)就可，这时 sqluldr2 就会输出记录到标准输出设备了，通常是指我们使用的屏幕。

```
D:\>sqluldr2 parfile=parfile.txt file=-
EMPNO#ENAME#JOB#MGR#HIREDATE#SAL#COMM#DEPTNO#
7369#SMITH#CLERK#7902#1980-12-17 00:00:00#800##20
7499#ALLEN#SALESMAN#7698#1981-02-20 00:00:00#1600#300#30
7521#WARD#SALESMAN#7698#1981-02-22 00:00:00#1250#500#30
7566#JONES#MANAGER#7839#1981-04-02 00:00:00#2975##20
7654#MARTIN#SALESMAN#7698#1981-09-28 00:00:00#1250#1400#30
7698#BLAKE#MANAGER#7839#1981-05-01 00:00:00#2850##30
7782#CLARK#MANAGER#7839#1981-06-09 00:00:00#2450##10
7839#KING#PRESIDENT##1981-11-17 00:00:00#5000##10
……
```

有了这个功能后，从 Oracle 向 Oracle 迁移数据时，就可以节约大量的时间，原理如下图所示：

批注 [l11]: 图加上图题，图序补：这个图，后面也不会有引用，后面章节后，每个图都加序的话，好象看起来也是很爽。




在传统模式下，必须等 sqluldr2 将所有数据生成到文件后，才能进行 SQL*Loader 的数据装载，而采用管道方式后，sqluldr2 边产生数据，同时 SQL*Loader 就将数据进行装载，通过串行到并行的过程来节约大量的空间和时间。使用方法极为简单：

```
sqluldr2  scott/tiger … file=- | sqlldr control=…
sqluldr2  scott/tiger … file=- | mysql …
```

在前面的例子中，可以发现当运行 sqluldr2 导出时，在屏蔽上都会输出日志信息，如：

```
D:\>sqluldr2 scott/tiger query="select * from emp"
       0 rows exported at 2009-11-10 11:22:29, size 0 MB.
      12 rows exported at 2009-11-10 11:22:29, size 0 MB.
         output file uldrdata.txt closed at 12 rows, size 0 MB.
```

当集成 sqluldr2 在脚本中时，就希望屏幕上不输出这些信息，但又希望这些信息能保留，这时可以用“log”选项来指定日志文件名。同“FILE”选项一样，如果文件面的第一个字符为加号（+），则表示在现有的文件后面添加，否则会生成新文件，如果已经有同名文件

则覆盖现有文件；也同样可以用如下格式码：

| 选项名 | 作用 | 例子 |
| --- | --- | --- |
| %Y、%y | 表示当前时间的四位年份 | log=A%y%m%d.log |
| %M、%m | 表示当前时间的两位月份 | |
| %D、%d | 表示当前时间的两位日份 | |
| %W、%w | 表示当前时间的一星期中的天数 | log=A%w.log |

例如连续运行两次如下命令：

```
sqluldr2  scott/tiger sql=emp.sql log=+emp.log
sqluldr2  scott/tiger sql=emp.sql log=+emp.log
```

然后来看一下生成的日志文件：

```
D:\>cat emp.log

       0 rows exported at 2009-03-18 20:05:44
      14 rows exported at 2009-03-18 20:05:44
         output file uldrdata.txt closed at 14 rows.


       0 rows exported at 2009-03-18 20:05:45
      14 rows exported at 2009-03-18 20:05:45
         output file uldrdata.txt closed at 14 rows.
```

一般在脚本中都用这个选项，以方便查看历史记录及性能数据。

# 性能调优选项

为了能更快地导出大量的数据，sqluldr2 中有几个可以用于性能优化的命令行选项，虽然决定性能的是 SQL 语句本身及服务器的处理能力，但合理的优化还是可以产生明显的效果。有 5 个和性能有关的选项：

| 选项 | 格式 | 作用 |
| --- | --- | --- |
| array | 数字 | 指定 Array Fetch 的大小 |
| read | 数字 | Oracle 一次读的最大块数 |
| sort | 数字（单位：MB） | 指定排序区的大小 |

| | | |
|---|---|---|
| hash | 数字（单位：MB） | 指定 Hash 区的大小 |
| serial | yes | 指定当全表扫描时用 direct path read |

当客户端程序从数据库服务器上获取数据时，可以要求服务器商一次调用返回多条记录到客户端程序的 Cache 中以快速处理大量记录，而不是很慢地一条一条地处理，称为 Array Fetch。array 选项用于指定一次调用返回的记录数的多少，默认值是 50，对于大部份情况下足够用了，如果网络延迟特别严重，或记录特别短时可以指定更大的值，最大可以指定到 2000。下面是我在以前公司，客户端从中国访问美国的数据库时，不同 array 大小下返回 100 条记录的时间对比，虽然是用 Java 程序做的测试，但原理相同，因此结果对 sqluldr2 的 array 选项取值也具有参考意义：

| Array | Time | Consistent Gets | Bytes | Roundtrips |
|---|---|---|---|---|
| 1 | 16.858 | 582 | 10K | 204 |
| 2 | 8.529 | 516 | 8323 | 104 |
| 10 | 1.852 | 463 | 6923 | 24 |
| 20 | 1.021 | 457 | 6748 | 14 |
| 50 | 0.531 | 451 | 6642 | 8 |
| 100 | 0.370 | 450 | 6608 | 6 |

可以看到 array 选项对于大量记录时的作用非常明显，使用方法如下所示：

```
sqluldr2  scott/tiger query=”select * from emp”array=500 ……
```

Sqluldr2 常被用于大量数据的文本导出，多数情况下是要扫描一个表或分区的大部份记录，这种情况下，如果增大 Oracle 数据库一次从磁盘读取的数据块数量，可以有效地加快导出速度，可以用 read 选项来指定这个块数，如下面的命令：

```
sqluldr2  scott/tiger query=”select * from emp”  read=128 ……
```

Sqluldr2 会在执行查询语句之前，执行如下 SQL 语句改变当前会话的多块读参数：

```
ALTER SESSION SET DB_FILE_MULTIBLOCK_READ_COUNT=128
```

Sort 和 hash 选项则用于改变数据库的排序运算内存和哈希运算内存区域的大小，相当于在执行查询语句之前执行如下语句：

```
ALTER SESSION SET SORT_AREA_SIZE=nM
ALTER SESSION SET SORT_AREA_RETAINED_SIZE=nM
ALTER SESSION SET HASH_AREA_SIZE=nM
```

**批注 [baiaiping12]:** 大小写与前面不一致。
补：工具中使用是各选项名不区分大小写的，因此比较随意，sorry。

当在 Oracle 数据库中执行一个全表扫描操作时，一般情况下要做 db file scatter read，这时读进来的大量数据块会冲击现有的在数据缓冲区（DB Cache）中访问较频繁的数据块，引起更多的 IO 操作，可以通过在会话级设置隐含参数“_serial_direct_read”为 TRUE 来防止这种情况，让访问表的操作变成 direct path read，数据块不经过数据缓冲区直接处理，在比较忙的时间段导出大量数据时，尤其显得重要，将 serial 选项设 YES 就可以。例如：

```
Sqluldr2 scott/tiger query=emp file=emp.txt serial=yes ……
```

相当于在执行查询 SQL 以前执行了以下 SQL 语句：

```
ALTER SESSION SET “_serial_direct_read”=TRUE
```

这些命令行参数，极大地方便了优化工作的进行。前面提到的快捷选项“FAST”，其实就是通过一个选项同时设置了这 5 个选项的经验值，其中：read=128 sort=128 hash=128 array=2000 serial=yes。

在导出时，sqluldr2 默认会每隔 100 万条记录打印一条日志信息，包括时间及生成文件的大小。如下所示：

```
D:\>sqluldr2 sys query=″select rownum from all_objects a, all_objects b″
       0 rows exported at 2009-11-10 12:55:58, size 0 MB.
 1000000 rows exported at 2009-11-10 12:56:00, size 4 MB.
 2000000 rows exported at 2009-11-10 12:56:02, size 12 MB.
 3000000 rows exported at 2009-11-10 12:56:05, size 20 MB.
 4000000 rows exported at 2009-11-10 12:56:07, size 28 MB.
 5000000 rows exported at 2009-11-10 12:56:09, size 36 MB.
```

如果希望改成每 10 万条打印一条日志，可以用“ROWS”选项来控制，如下所示：

```
D:\>sqluldr2 sys rows=100000 query=″select rownum from all_objects a,
all_objects b″
       0 rows exported at 2009-11-10 13:00:17, size 0 MB.
  100000 rows exported at 2009-11-10 13:00:18, size 0 MB.
  200000 rows exported at 2009-11-10 13:00:18, size 0 MB.
  300000 rows exported at 2009-11-10 13:00:18, size 0 MB.
  400000 rows exported at 2009-11-10 13:00:18, size 0 MB.
```

“ROWS”和真实的性能无关，更改的只是 sqluldr2 报告进度的频率。

## Sqlldr 设置选项

Oracle 虽然不提供文本导出工具，但却提供了文本导入工具 SQL * Loader，但这个工具对于大部份人来讲，还是过于复杂了，因为它需要一个控制文件（也可以称为参数文件）来说明要导入的文本文件的格式，如字段的分隔符，记录的分隔符。为了方便，将导出的文本导入到新的数据库中，sqluldr2 可以直接生成这个控制文件，在导出结束后，可以很方便地用 SQL*Loader 将数据导入到新的数据库中。有三个选项和这个功能有关：

| 选项 | 格式 | 作用 |
| --- | --- | --- |
| table | 表名 | 指定 sqlldr 的目标表名 |
| mode | 方式 | 装载方式，INSERT、APPEND、REPLACE、TRUNCATE |
| buffer | 缓冲区大小 | 指定装载时的 readsize 和 bindsize，单位：MB |

“table”选项用于指定将文件导入的目标表的名字，例如我们将 EMP 表的数据导入到 EMP_HIS 表中，假设这两个表的表结构一致，先用如下命令导出数据：

```
sqluldr2 … query=”select * from emp” file=emp.txt table=emp_his ……
```

这时在当前目录下生成了一个名为“目标表名_sqlldr.ctl”的控制文件，这里就是“emp_his_sqlldr.ctl”，用文本编辑器打开“emp_his_sqlldr.ctl”，可以看到如下内容：

```
--
-- SQL*UnLoader: Fast Oracle Text Unloader (GZIP), Release 3.0.1
-- (@) Copyright Lou Fangxin (AnySQL.net) 2004 - 2010, all rights reserved.
--
--  CREATE TABLE emp_his (
--    EMPNO NUMBER(4),
--    ENAME VARCHAR2(10),
--    JOB VARCHAR2(9),
--    MGR NUMBER(4),
--    HIREDATE DATE,
--    SAL NUMBER(7,2),
--    COMM NUMBER(7,2),
--    DEPTNO NUMBER(2)
--  );
--
OPTIONS(BINDSIZE=2097152,READSIZE=2097152,ERRORS=-1,ROWS=50000)
LOAD DATA
INFILE 'uldrdata.txt' "STR X'0a'"
INSERT INTO TABLE emp_his
FIELDS TERMINATED BY X'2c' TRAILING NULLCOLS
(
  "EMPNO" CHAR(6) NULLIF "EMPNO"=BLANKS,
  "ENAME" CHAR(10) NULLIF "ENAME"=BLANKS,
  "JOB" CHAR(9) NULLIF "JOB"=BLANKS,
  "MGR" CHAR(6) NULLIF "MGR"=BLANKS,
  "HIREDATE" DATE "YYYY-MM-DD HH24:MI:SS" NULLIF "HIREDATE"=BLANKS,
  "SAL" CHAR(10) NULLIF "SAL"=BLANKS,
  "COMM" CHAR(10) NULLIF "COMM"=BLANKS,
  "DEPTNO" CHAR(4) NULLIF "DEPTNO"=BLANKS
)
```

不需要任何编辑，就可以用 SQL*Loader 将数据装载到 EMP_HIS 表中：

```
D:\>sqlldr scott/tiger control=emp_his_sqlldr.ctl

SQL*Loader: Release 10.2.0.1.0 - Production on Mon Feb 2 22:30:53 2009

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Commit point reached - logical record count 14
```

这个功能可以用来做许多的事，如编写数据收集脚本，或在不同字符集之间数据的迁移，

后者只要在导出时按源数据库字符集设定 NLS_LANG，而在用 SQL*Loader 装载时按目标库的字符集设定相应的 NLS_LANG 即可以。例如将 US7ASCII 库下的汉字，转换到 UTF8 字符集的数据库下：

```
SET NLS_LANG=.US7ASCII
sqluldr2 ……
SET NLS_LANG=.ZHS16GBK
sqlldr ……
```

这种转换通过 DB LINK 是很难实现的，通过 Shell 脚本和 sqluldr2 工具则可以批量实现，简单的工具有时也能帮上大忙。用 sqlldr 装载数据时，有四种模式（INSERT、APPEND、REPLACE、TRUNCATE）可以选择，可以通过 mode 参数指定。各种类型的含义如下表：

| 模式 | 说明 |
| --- | --- |
| INSERT | 装载到空表，如果目标表中有记录，则报错。 |
| APPEND | 装载记录到现有表。 |
| REPLACE | 先用 DELETE 语句删除所有记录，然后装载记录。 |
| TRUNCATE | 先 TRUNCATE 目标表，然后装载记录。 |

在装载时，可以指定一个缓冲区，一般而言缓冲区越大，装载速度越快，默认值为 2MB，大多数情况下已经够用了，在导入巨量数据时，可以用“BUFFER 选项”指定更大的值。

## 使用参数文件

sqluldr2 累计已经有超过 30 个不同的命令行选项，常用的也有接近十个，要在命令行指定这些选择，不仅命令输入很复杂，而且看起来也不爽。在 Oracle 的导入导出工具中，都可以将很多参数放到一个参数文件中，然后指定一个参数文件就可以跑了。在 sqluldr2 中也可以这样，比如我们创建如下参数文件（“parfile.txt”）：

```
user=scott/tiger
query=select * from emp
field=#
head=yes
file=+emp.txt
```

我们可以用“PARFILE”选项来批定参数文件的位置，sqluldr2 会从参数文件中读取选

项值。如下所示：

```
D:\>sqluldr2 parfile=parfile.txt
       0 rows exported at 2009-11-10 13:13:04, size 0 MB.
      12 rows exported at 2009-11-10 13:13:04, size 0 MB.
         output file +emp.txt closed at 12 rows, size 0 MB.
```

我们可以利用参数文件来保存一些公共的属性，然后在命令行指定少数几个经常变化的选项值，以简化命令行的输入。在介绍登录相关的命令行参数时，我们曾提过将登录信息放在参数文件中，然后控制参数文件的访问权限，以达到提高安全性的作用。例如：

```
D:\>sqluldr2 parfile=parfile.txt sql=file1.sql file=file1.txt
D:\>sqluldr2 parfile=parfile.txt sql=file2.sql file=file2.txt
D:\>sqluldr2 parfile=parfile.txt sql=file3.sql file=file3.txt
……
```

通过参数文件，也可以使得配置简单起来，脚本调度也比较好处理。

## MySQL 相关选项

MySQL 是目前数据库中最明亮的星星，给许多企业，尤其是互联网企业，带来了控制运行成本的数据库架构，越来越多的企业选择将部份或大部份数据从传统的 Oracle 数据迁移到 MySQL，sqluldr2 的文本导出功能，也必能在这一场变迁中留下几个印迹。

对 MySQL 增加了几个特殊参数后，使得从 Oracle 迁移数据到 MySQL 变得极为简单。在过去的数据迁移实践中发现主要有两个问题，第一个是 NULL 值问题，MySQL 在装载文本文件到数据库时，NULL 值，不象 Oracle 一样，在 Oracle 中长度为 0 的字符串等于空值，而在 MySQL 中长度为 0 的字符串不同于 NULL 值，而需要用转义符加大写的 N 字母来表示。第二个是转义符问题，要处理 NULL 值，就一定要用到转义符，转义符是一个字母，很容易出现在字段值中，在处理海量数据时可能根本找不到一个适合（不会出现在字段值中）的字母来充当转义符。

比如我们选择反斜杆（“\”）来作为转义符，那如果反斜杆（“\”）出现在了字段值中，在为 MySQL 生成文本文件时，就要做相应的转义处理。如下表所示：

| 转义前 | 转义后 |
| --- | --- |
| NULL | \N |

| | |
|---|---|
| \ | \\ |

否则我们将无法轻松地将数据从 Oracle 迁移到 MySQL 中。可以用以下五个选项来完这个功能，以文本方式或 INSERT 语句方式，做到轻松迁移大量数据到 MySQL 中。

| 模式 | 说明 |
|---|---|
| NULL | 指定 NULL 值的表达式，默认为空，Oracle 的风格。 |
| ESCAPE | 指定转义字符，默认为空格（不做转义）。 |
| ESCF | 指定需要转义的字符列表 |
| ESCT | 指定转义后的字符列表 |
| FORMAT | 导出成 MySQL 的多行 INSERT 语句。 |

在 EMP 的 MGR 和 COMM 字段中，有不少空值，我们在导出时替换成 MySQL 风格式的 NULL 值时，需要指定“NULL”选项，如下所示：

```
D:\>sqluldr2 scott/tiger query=emp null=0x5cN file=-
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,\N,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,\N,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,\N,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,\N,10
7839,KING,PRESIDENT,\N,1981-11-17 00:00:00,5000,\N,10
……
```

当数据中出现转义符时，使用”ESCAPE”选项就可以在 sqluldr2 中进行相应的转义处理了，淘宝 DBA 在用早期版本的 sqluldr2 做 Oracle 到 MySQL 的大数据量迁移时，因为没有用这个功能，用操作系统的 sed 命令来处理，使迁移脚本变得比较复杂，不能说不是一次比较痛苦的经历。

比如我们要在员工名字后加一个反斜杆，并将数据导入到 MySQL 中，则 sqluldr2 在导出时，必须要处理一下字段值中的反斜杆。

```
select empno, ename||'\' ename, mgr from emp
```

通过指定“ESCAPE”选项值及需要转义的字符列表。对于 MySQL，常用的有 5 个需要转义的字符：

| 转义前 | 转义后 |
|---|---|

| | |
|---|---|
| 转义符 | 转义符+转义符 |
| 回车 | 转义符+r |
| 换行 | 转义符+n |
| 单引号 | 转义符+单引号 |
| 双引号 | 转义符+双引号 |

通过指定“ESCF/ESCT”选项值，可以达到上面的转义效果：\

```
ESCF=0x0d0x0a0x270x22
ESCT=rn0x270x22
```

针对上面的 SQL 语句对应的数据源，用以下的导出命令：

```
D:\>sqluldr2 scott/tiger query="……" null=0x5cN escape=0x5c
SEDF=0x0d0x0a0x270x22 SEDT=rn0x270x22 file=-
7369,SMITH\\,7902
7499,ALLEN\\,7698
7521,WARD\\,7698
7566,JONES\\,7839
……
```

通过指定以下选择，就可以生成兼容 MySQL 数据装载命令（LOAD DATA）的文本文件了。前面提到的快捷选项（“TEXT”），如果设置选项值为“MYSQL”，即是以更方便的方式定如下选项。再也不用费尽心机去挑选可用的分隔符了，也不用再在操作系统层面做文本替换了。

```
escape=0x5c
ESCF=0x0d0x0a0x270x22
ESCT=rn0x270x22
quote=0x22
null=0x5cN
field=0x2c
record=0x0a
```

相对应的 MySQL 装载命令为：

```
LOAD DATA LOCAL INFILE '文件名'  INTO TABLE 表名
  FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
  ESCAPED BY '\\' LINES TERMINATED BY '\n';
```

在迁移数据到 MySQL 的项目中，我们发现没有办法通过管道将文本文件传给 MySQL 的数据装载（LOAD DATA）命令，也就是说 MySQL 的 LOAD DATA 命令无法从标准输入读入数据，因此没有办法利用 sqluldr2 输出到标准输出的功能，无法发挥管道来节约空间或时间。但 MySQL 的 Insert 命令一次也可以插入多条数据（这个功能实在很爽，其他的数据库也应当学习学习），并且 MySQL 中并不存在绑定变量这个概念，执行文本方式的 SQL 语句也是很快的，因此可以将数据导出成 MySQL 的 INSERT 语句，然后就可以利用管道机制了。

通过“FORMAT”参数，可以生成 MySQL 的 INSERT 语句，比如我们要将“DEPT”这个表的数据移到 MySQL 中，则可以将这个表的数据导出成 INSERT 语句。使用如下参数文件。

```
escape=0x5c
ESCF=0x0d0x0a0x270x22
ESCT=rn0x270x22
quote=0x27
null=null
format=mysql
```

前面提到的快捷选项（“TEXT”），如果设置选项值为“MYSQLINS”，即是以更方便的方式设定如下选项。然后运行如下命令进行导出：

```
D:\>sqluldr2 scott/tiger parfile=parfile.txt query=dept table=dept file=-
INSERT INTO dept (DEPTNO,DNAME,LOC) VALUES
(10,'ACCOUNTING','NEW YORK'),
(20,'RESEARCH','DALLAS'),
(30,'SALES','CHICAGO'),
(40,'OPERATIONS','BOSTON');
```

如果在 MySQL 上已经建好了“DEPT”表，则直接用管道符就可以迁移数据了。

```
D:\>sqluldr2 scott/tiger …… text=mysqlins file=- | mysql test
```

推出这个功能后，有人来责怪我，为什么没有早点推出来！

# 命令返回值

在脚本编程语言中，结合 sqluldr2 和 sqlldr 或其他文本导入工具，可以很方便实现你想要的数据收集或迁移的功能。但要在 Windows 批处理、Shell 脚本或 Perl 程序中调用 sqluldr2 并进行智能控制，编写更强的功能，就需要知道一个命令执行的结果，可以通过 sqluldr2 执行的返回值来判断。Sqluldr2 设计的返回值如下表所示：

| 返回值 | 含义 |
|---|---|
| 0 | 操作成功 |
| 1 | 不能登录数据库 |
| 2 | 不能创建 Cursor |
| 3 | 不能分析 SQL |
| 4 | 不能执行 SQL |
| 5 | 不能解释返回的记录结果集 |
| 6 | 不能生成输出文件 |
| 7 | 在导出过程中遇到数据库错误，如 ORA-01555 |

各种脚本语言都可以方便地取得上一个命令执行的退出代码（通常表示执行的结果），在 Windows 中可以用 ErrorLevel 来判断 sqluldr2 是否成功运行，当返回值大于 0 时表示不成功，测试代码如下所示：

```
@echo off
Sqluldr2 user=system/oracle query="select * from tab" ……
if ERRORLEVEL 1 (
   echo error ) ELSE (
   echo Success )
```

在 Linux/Unix 的 Shell 中，则可以用如下代码进行测试：

```
#!/bin/sh

Sqluldr2 user=system/oracle query="select * from tab" ……
if [ $?==0 ]; then
   echo success
else
   echo error
#fi
```

在 Perl 中可以用 system 函数来调用执行 sqluldr2 命令，system 函数的返回值就是 sqluldr2 的退出代码，控制更加灵活，脚本可以更加强壮。

## 编译 OCI 程序

基于 OCI 8 接口规范开发的 Sqluldr2 源代码还没有公开，不过你可以下载 OCI 7 接口版本的源代码（这个早期版本已经停止更新，因此并不具有上面所说的全部功能，下载地址：http://www.anysql.net/software/ociuldr.c），并自行编译它。

早期源程序因为用 OCI 7 的接口规范开发，目前只能编译成 32 位的程序，在 Windows 下可以用 Microsoft Visual C++ 6.0 及以上的开发环境下进行编译。而在 Linux/Unix 平台下，则可以用 gcc 来进行编译。

在 Windows 下，安装 Oracle 时需要安装 OCI 的开发包，所有文件安装在在 Oracle 主目录下的 oci 子目录下，然后用以下命令进行编译：

```
set OH=Oracle 安装目录
cl –I%OH%\oci\include –L%OH%\oci\lib\msvc ociuldr.c oci.lib
```

在 Linux 和 Unix 下，如果是 32 位的 Oracle 或 Oracle 客户端，用如下命令进行编译：

```
# 32 位的 Oracle
export OH=Oracle 安装目录
gcc -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 \
    -I$OH/rdbms/demo -I$OH/rdbms/public          \
    -L${OH}/lib -lm  -lclntsh -o sqluldr.bin     \
    ociuldr.c
```

如果是 64 位的 Oracle 或 Oracle 客户端，则用如下命令：

```
# 32 位的 Oracle
export OH=Oracle 安装目录
gcc -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 \
    -I$OH/rdbms/demo -I$OH/rdbms/public          \
    -L${OH}/lib32 -lm  -lclntsh -o sqluldr.bin   \
    ociuldr.c
```

在 AIX 下为了支持 64 位的 IO，编译选项如下：

```
# 使用 32 位的 Oracle 客户端库
export OH=Oracle 安装目录
gcc  -D_LARGE_FILES \
    -I$OH/rdbms/demo -I$OH/rdbms/public          \
    -L${OH}/lib32 -lm  -lclntsh -o sqluldr.bin   \
    ociuldr.c
```

在 64 位的环境中，因为有 32 位的客户端库和 64 位的客户端库之分，而数据库的安装文档中一般设置的环境变量（LD_LIBRARY_PATH 和 LIBPATH）是指向 64 位的库文件的，因此通常将可执行文件编译成 sqluldr.bin，然后创建一个 Shell 脚本文件 sqluldr2 来启动程序。

在脚本中，先检查一下有没有设置 Oracle 环境变量，然后设置 LD_LIBRARY_PATH 或 LIBPATH（AIX 或某些 Unix 下）变量指向 32 位的客户端库所在的目录。如上所示：

```
#!/bin/sh

if [ "A${ORACLE_HOME}A" = "AA" ]; then
  echo "ORACLE_HOME environment variable not setted."
  exit
fi
if [ -d ${ORACLE_HOME}/lib32 ]; then
   LD_LIBRARY_PATH=${ORACLE_HOME}/lib32:${LD_LIBRARY_PATH}
else
   LD_LIBRARY_PATH=${ORACLE_HOME}/lib:${LD_LIBRARY_PATH}
fi
export LD_LIBRARY_PATH
export LIBPATH=$LD_LIBRARY_PATH
${0}.bin "$@"
```

上面的脚本已经使用几年了，运行结果很好。基于 OCI 8 版本的 sqluldr2 则可以编译成 64 位程序。

# 性能测试

前面提到，阿里巴巴（Alibaba）集团的 DBA 几年前就开始用 sqluldr2 早期版本为数据仓库导出数据，并在真实应用环境下创下了我所见过的文本导出速度的最高记录，可能是觉得是由于他们的机器很强，下面用笔记本来进行测试，会更容易看出效果。

## 导出文本测试

测试所用表有 8 个字段，字段类型有字符、数字、日期，表结构如下：

```
Name                          Null?    Type
----------------------------- -------- -----------------
EMPNO                                  NUMBER(4)
ENAME                                  VARCHAR2(10)
JOB                                    VARCHAR2(9)
MGR                                    NUMBER(4)
HIREDATE                               DATE
SAL                                    NUMBER(7,2)
COMM                                   NUMBER(7,2)
DEPTNO                                 NUMBER(2)
```

以分隔符导出时，生成 186MB 的数据文件，用时 30 秒，日志信息如下：

```
      0 rows exported at 2009-03-22 16:56:23
 500000 rows exported at 2009-03-22 16:56:27
1000000 rows exported at 2009-03-22 16:56:30
1500000 rows exported at 2009-03-22 16:56:35
2000000 rows exported at 2009-03-22 16:56:39
2500000 rows exported at 2009-03-22 16:56:43
3000000 rows exported at 2009-03-22 16:56:47
3500000 rows exported at 2009-03-22 16:56:52
3670044 rows exported at 2009-03-22 16:56:53
        output file uldrdata.txt closed at 3670044 rows.
```

以固定长度格式导出时，生成 220MB 的数据文件，用时 31 秒，日志信息如下：

```
      0 rows exported at 2009-03-22 17:01:38
 500000 rows exported at 2009-03-22 17:01:42
1000000 rows exported at 2009-03-22 17:01:46
1500000 rows exported at 2009-03-22 17:01:50
2000000 rows exported at 2009-03-22 17:01:54
2500000 rows exported at 2009-03-22 17:01:59
3000000 rows exported at 2009-03-22 17:02:03
3500000 rows exported at 2009-03-22 17:02:08
3670044 rows exported at 2009-03-22 17:02:09
        output file uldrdata.txt closed at 3670044 rows.
```

在笔记本电脑上每秒钟导出的速度达到 12 万条记录以上。用 exp 工具测试，导出需要 20 到 24 秒，sqluldr2 差不多有 75%的 exp 的速度了，考虑到导出成文本文件除字符串外的数据类型都要作类型转换，这个速度已经很不错了。

对于 sqlldr 分隔符方式和固定长度格式导入，也来做一个比较。先比较用普通模式导入的时长信息：在笔记本上每秒钟导出的速度达到 12 万条记录以上。用 exp 工具测试，导出需要 20 到 24 秒，sqluldr2 差不多有 75%的 exp 的速度了，考虑到导出成文本文件除字符串外的数据类型都要作类型转换，这个速度已经很不错了。

对于 sqlldr 分隔符方式和固定长度格式导入，也来做一个比较。先比较用普通模式导入的时长信息：

批注 [baiaiping13]: 建议这两个表合并，或者画柱状图，以使对比更加直观。
补：写成表格，而不是直接画出数值，主要是为了能表示，从导入日志取的是那一行数据。

| 普通方式 | 日志信息 |
| --- | --- |
| 分隔符 | Elapsed time was: 00:00:32.73<br>CPU time was: 00:00:13.57 |
| 固定长度 | Elapsed time was: 00:00:29.83<br>CPU time was: 00:00:13.23 |

再来比较一下 Direct 方式的导入时长信息

| Direct 方式 | 日志信息 |
| --- | --- |
| 分隔符 | Elapsed time was: 00:00:29.83<br>CPU time was: 00:00:13.23 |
| 固定长度 | Elapsed time was: 00:00:17.42<br>CPU time was: 00:00:08.93 |

可以看到，固定格式的文本在用 Direct 方式装载时，与分隔符方式相比，速度提升很多。

## 导出 SQL 测试

我们以同样的表来进行导出成 MySQL 用的 INSERT 语句的速度测试，不过数据量小一点，用约 100 万条记录做测试数据进行测试，测试表有 8 个字段，包括字符，数字和日期类型。

首先是测试不做任何转义符处理，生成文本文件的速度，基本上受制于笔记本硬盘写出的速度了。

```
D:\>sqluldr2 scott/tiger query=emp_his
         0 rows exported at 2009-10-30 22:37:14, size 0 MB.
    961307 rows exported at 2009-10-30 22:37:21, size 48 MB.
           output file uldrdata.txt closed at 961307 rows, size 49 MB.
```

接下是生成转义符处理过后的，MySQL 用的文本文件，速度在这次测试环境下没有下

降，转义处理主要会消耗客户端机器的 CPU，如果客户端的 CPU 不是问题，则导出速度不会受到影响。

```
D:\>sqluldr2 scott/tiger query=emp_his text=mysql
         0 rows exported at 2009-10-30 22:38:49, size 0 MB.
    961307 rows exported at 2009-10-30 22:38:56, size 52 MB.
           output file uldrdata.txt closed at 961307 rows, size 56 MB.
```

测试生成 MySQL 插入用的 Insert SQL 文件，速度在这次测试环境下也没有下降。

```
D:\>sqluldr2 scott/tiger query=emp_his text=mysqlins table=emp_his
         0 rows exported at 2009-10-30 22:39:28, size 0 MB.
    961307 rows exported at 2009-10-30 22:39:34, size 56 MB.
           output file uldrdata.txt closed at 961307 rows, size 60 MB.
```

当然我们还要去测试 MySQL 下，通过 LOAD DATA 命令和通过 INSERT 语句的性能差异，如果 INSERT 的性能也足够好，那么导出成 INSERT 语句将是迁移数据到 MySQL 的最佳方式。

## sqluldr2 应用小结

批注 [I14]: 是否举几个应用实例？
补：在后面有使用的例子，所以这里简单引申一下而已，也为了控制工具介绍上的篇幅。

做了多年 DBA 工作后，认为管理数据库要大量依靠数据，而不是纯靠个人经验，系统越复杂越需要数据的支持。在收集数据的工作中，sqluldr2 就帮了很大的忙，例如用 sqluldr2 配合 sqlldr 来收集表空间的空间使用情况，以及表数据库各对象的大小，每天只需要执行一次。在积累了几个星期后，使我对于表空间的空间使用情况了如指掌，积累半年后，使我对于空间的增长了如指掌，更为下一年的存贮采购提供了有力的计算依据。

后来用 sqluldr2 来收集越来越多的信息，对系统及业务的了解也越来深，有好的工具为基础，使我更有时间去专注于那些高级的任务，而不是将时间浪费在编写低级的任务脚本编写中。