ociuldr 来自 AnySQL.net 的文本导出工具

文本格式在不同的数据库之间进行数据交换还是很有用的，我第一次用 Java 写具有这个功能的程序就是为了将 Oracle 数据库中的数据导出成格式化文本，然后交给数据仓库组，将数据装载到 Sybase IQ 中进行分析。后来发现文本格式在不同字符集的 Oracle 数据库之间交换数据也是很有用的。然而 Oracle 却没有发布高效的工具，以让人方便地将数据导出成文本格式，估计是不愿意人家将数据移别的数据库上吧！

最具名气的具有这个功能的工具是 AskTom 上发布的段程序，一个是用 SQL*Plus 的 Spool 功能实现的角本程序；另一个是 Tom Kyte 写的一段 unload.pc 程序。目前，ociuldr 还不太为人们所知，即然有了这两个好的程序，为什么还要写 ociuldr 呢？用 Spool 实现的方法不太方便，而他的 unload.pc 程序在不同的 Oracle 客户端版本需要重新编译，并且功能有限。ociuldr 同他们相比具有以下特有功能：

- 用 OCI 写成，性能不亚于 Tom 的 unload.pc 程序
- 在 Oracle 8i/9i/10g 的客户端下不需要重新编译
- 可以指定任意的字段分隔符和记录分隔符
- 可以自动生成用于装载记录的 SQL*Loader 控制文件
- 更多的性能调整选项

下面就来学习和使用一下这个免费的小工具吧！

ociuldr 的语法及命令行选项

ociuldr 是一个命令行工具，不是一个图形向导式的工具，这一点可能不为你所喜欢，不过因为是命令行工具，使得写出来的程序可以在 Linux/Unix 直接进行编译，以支持多种平台，已经编译好的平台有 Windows、RedHat Linux、Solaris

Sparc 64bit，另外还在 AIX 上编译过，由于没有足够的编译环境，因此将源代码直接放在个人主页(AnySQL.net)上进行下载。

- 软件下载：**http://www.anysql.net/software/ociuldr.zip**
- 源码下载：**http://www.anysql.net/software/ociuldr.c**

不带任何参数运行 ociuldr 就可以获得命令帮助：

```
C:\MYDUL>ociuldr
Usage: ociuldr user=... query=... field=... record=... file=...
(@) Copyright Lou Fangxin 2004/2005, all rights reserved.
Notes:
       -si   = enable logon as SYSDBA
       user  = username/password@tnsname
       sql   = SQL file name
       query = select statement
       field = seperator string between fields
       record= seperator string between records
       file  = output file name(default: uldrdata.txt)
       read  = set DB_FILE_MULTIBLOCK_READ_COUNT at session level
       sort  = set SORT_AREA_SIZE & SORT_AREA_RETAINED_SIZE
               at session level (UNIT:MB)
       hash  = set HASH_AREA_SIZE at session level (UNIT:MB)
       serial= set _serial_direct_read to TRUE at session level
       trace = set event 10046 to given level at session level
       table = table name in the sqlldr control file

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

下面来介绍一下各个命令行参数：

- **-si** 和 **user=**

-si 让 ociuldr 以 sys as sysdba 用户进行连接，user 则指定“用户名/口令@连接串”进行连接，这两个选项必需指定其中一个。指定 user 选项时，也必须指定口令，不会象 SQL*Plus 一样出现输入口令的提示。

- **sql=** 和 **query=**

这两个选项用于指定一个 select 语句，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。

- **field=** 和 **record=**

文本格式需要一个特殊的字符或字符串来区分字段和记录，field 用于指定字段间的分隔符，默认值是竖线；而 record 则用于指定多条记录之间的分隔

符，默认值是回车加换行(Windows 风格)。需要告诉大家的是，在指定这两个选项的值时，要选用不会出现在字段值中的字符或字符串，另外还可以用 ASCII 代码(**http://www.anysql.net/developer/ascii_hex_table.html**)来指定任何字符，例如我常指定 0x07 来作为字段间的分隔符，用 0x06 来作为记录间的分隔符，这两个值都是很难从键盘输入到数据库中的。

最近常看到网上有人问如何生成 Excel CSV 格式的文件，其实只是一个以逗号分隔字段，以换行符分隔记录的文本文件。

- **file=**

这个选项用于指定生成的文本文件的名字，默认文件名为 uldrdata.txt。当你指定了 batch 选项以生成多个文本文件时，可以用”%d”来代表生成的文件的序号，如”test_%d.txt”。

- **read=**

在 Oracle 数据库中，DB_FILE_MULTIBLOCK_READ_COUNT 参数可以在会话级更改不同的值，以进行对 SQL 的优化。当设定这个选项时，ociuldr 会在执行查询时先在会话级更改设置。

- **sort=**

在 Oracle 数据库中，SORT_AREA_SIZE 和 SORT_AREA_RETAINED_SIZE 参数可以在会话级更改不同的值，以进行对 SQL 的优化，当设定这个选项时(单位：MB)，ociuldr 会在执行查询时先在会话级更改设置。

- **hash=**

在 Oracle 数据库中，HASH_AREA_SIZE 参数可以在会话级更改不同的值，以进行对 SQL 的优化，当设定这个选项时(单位：MB)，ociuldr 会在执行查询时先在会话级更改设置。

- **serial=**

在 Oracle 数据库中，_serial_direct_read 参数可以在会话级更改不同的值，当设为 TRUE 时，ociuldr 会在执行查询时先在会话级更改设置，使得 Oracle 在进行全表访问时采用 Direct Path Read 方式。

- **trace=**

在 Oracle 数据库中，10046 事件可以在会话级设定不同的值(1,4,8,12)以生成不同详细程度的 SQL Trace 文件，以跟踪 SQL 语句的执行情况。

- **table=**

  用这个选项指定目标表名后，ociuldr 可以生成用 SQL*Loader 装载数据用的控制文件，这个选项的值会出现在控制文件的“INTO TABLE 目标表名”一行中。如果不指定这个选项的值，则不会生成控制文件。

- **head=**

  用于指定是否要在文本文件的第一行打印字段名，默认值为 OFF。当设为 YES 或 ON 时，就会在第一行打印字段名，这时从第二行开始才是导出的数据，你在装载时就需要注意了。生成的控制文件名字为“目标表名_sqlldr.ctl”。

- **batch**

  指定 ociuldr 将数据写出到多个文件。在写出指定的 batch 数(一个 batch 为 50 万条记录)后，ociuldr 将重新生成一个新的文件来存放导出的数据，请指定一个大于 1 的数。当指定这个选项时，请注意 file 选项的值中要包括”%d”，如：

  TEST_%d.TXT

  这时生成的文件将会依次是，TEST_1.TXT、TEST_2.TXT 等.

光从命令行的众多选项就可以看出，ociuldr 更能适应不同的需求，也不是简单地将 unload.pc 拿过来改改就完成的，而是我参考 OCI 样本程序，重新写的。

Select * from tab

接下来我们来做一个最简单的例子，我们选择以”#”作为字段的分隔符，以默认的回车换行作为记录分隔符。由于是用 OCI 编译的程序，所以需要确定你已经安装了 Oracle 客户端，以及执行了必要的网络配置。

在 Windows 上以 anysql 用户(口令：anysql)连接 test 数据库，执行情况如下所示：

```
C:\MYDUL>ociuldr user=anysql/anysql@test query="select * from tab"
field=#

       0 rows exported at 2007-03-06 11:40:33
      39 rows exported at 2007-03-06 11:40:33
         output file uldrdata.txt closed at 39 rows.
```

来看一下默认文件 uldrdata.txt 中的内容：

```
C:\MYDUL>cat uldrdata.txt
A#TABLE#
A_V#VIEW#
CCC#TABLE#
FACT_SALES#TABLE#
MV_FACT_SACLES#TABLE#
OBJD_LIST#TABLE#
STAT_T_HASH#TABLE#
STAT_T_HASH_P2#TABLE#
STAT_T_HASH_P3#TABLE#
STAT_T_PARTDEMO#TABLE#
……
```

再次 Select * from tab

接一来再来运行一次这个简单的 SQL 语句，这一次我们加上更多的选项：

- 字段分隔符为 0x07
- 记录分隔符为 0x06
- 文件名为 anysql.txt
- 生成 SQL*Loader 控制文件，目标表名为 ANYSQL_TAB
- 将 SQL 语句放在一个文本文件(tab.sql)中

现在来准备 SQL 文件 tab.sql：

```
C:\MYDUL>cat tab.sql
select *
from
tab
```

执行情况如下所示：

```
C:\MYDUL>ociuldr user=anysql/anysql@test sql=tab.sql field=0x07
record=0x06 file=anysql.txt table=anysql_tab

       0 rows exported at 2007-03-06 11:54:49
      39 rows exported at 2007-03-06 11:54:49
         output file anysql.txt closed at 39 rows.
```

再来看一下生成的 SQL*Loader 的控制文件 anysql_tab_sqlldr.ctl：

```
C:\MYDUL>cat anysql_tab_sqlldr.ctl
--
-- Generated by OCIULDR
--
OPTIONS(BINDSIZE=4194304,READSIZE=4194304,ERRORS=-1,ROWS=50000)
LOAD DATA
INFILE 'anysql.txt' "STR X'06'"
INTO TABLE anysql_tab
FIELDS TERMINATED BY X'07' TRAILING NULLCOLS
(
  TNAME CHAR(30),
  TABTYPE CHAR(7),
  CLUSTERID CHAR(40)
)
```

现在来创建一目标表 anysql_tab：

```
SQL> desc anysql_tab
 Name                           Null?    Type
 ------------------------------ -------- ----------------
 TNAME                          NOT NULL VARCHAR2(30)
 TABTYPE                                 VARCHAR2(7)
 CLUSTERID                               NUMBER
```

并且用 sqlldr 将刚导出的记录再装载回去：

```
C:\MYDUL>sqlldr anysql/anysql@test control=anysql_tab_sqlldr.ctl

SQL*Loader: Release 10.2.0.1.0 - Production on Tue Mar 6 12:01:21
2007

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

Commit point reached - logical record count 39
```

你可以到数据库中去查一下装载后的记录是否准确。

如何运行在 Unix/Linux 下

在 64 位的 Oracle 数据库中，可以分为 64 位的客户端和 32 位的客户端，在 Oracle 9i 以前，32 位的客户端库文件位于$ORACLE_HOME/lib 下面，64 位的客户端位于$ORACLE_HOME/lib64 下面；而 9i 及以上的版本中，32 位的客户端位于$ORACLE_HOME/lib32 目录中，64 位的客户端程序位于$ORACLE_HOME/lib 目录中，需要准确配置 LD_LIBRARY_PATH(AIX 下是 SHLIB)环境变量，才能让 ociuldr 在 Unix/Linux 下顺利地跑起来。

我主页上下载的 Linux 和 Solaris 下的版本都是编译成 32 位的程序，因此需要用到 32 位的 Oracle 客户端库文件，我将 ociuldr 的可执行文件命名为 ociuldr.bin，然后编写一个 Shell 文件(ociuldr)来起动程序：

```
#!/bin/sh

NLS_DATE_FORMAT="YYYY-MM-DD HH24:MI:SS"

if [ "A${ORACLE_HOME}A" = "AA" ]; then
   echo "ORACLE_HOME environment variable not setted."
   exit
fi

if [ "A${LD_LIBRARY_PATH}A" = "AA" ];then
   LD_LIBRARY_PATH=/lib:/usr/lib
fi

if [ -d ${ORACLE_HOME}/lib32 ]; then

LD_LIBRARY_PATH=${ORACLE_HOME}/lib32:${ORACLE_HOME}/lib:${LD_LIBRARY_
PATH}
else

LD_LIBRARY_PATH=${ORACLE_HOME}/lib:${ORACLE_HOME}/lib64:${LD_LIBRARY_
PATH}
fi

export LD_LIBRARY_PATH NLS_DATE_FORMAT

ociuldr.bin "$1" "$2" "$3" "$4" "$5" "$6" "$7" "$8" "$8"
```

这个程序主要是判断 ORACLE_HOME 环境变量是否设置，如果已经设置，则继续设置正确的 LD_LIBRARY_PATH 值以指向 32 位的客户端库文件。

如果发现在网上找不到你当前的平台，则可以下载源代码并自行编译。

在 Unix/Linux 下编译 ociuldr

ociuldr 的源文件只有一个 ociuldr.c，你需要首先安装由厂商提供的 c 编译器，或者你可以使用 gcc，下面是我在 Solaris 上编译 ociuldr 的命令，写在这儿以供参考：

```
gcc -m32 -g -Bsymbolic -t -D_LARGEFILE64_SOURCE -
D_FILE_OFFSET_BITS=64 -I$ORACLE_HOME/rdbms/demo -
I$ORACLE_HOME/rdbms/public -L${ORACLE_HOME}/lib32 -Wl,-i -o
ociuldr.bin ociuldr.c -lm -lclntsh -Wl,-Bdynamic
```

如果你发现不能编译，则可以在网上找找如何编译 OCI 程序的例子，也可以联系一下我。

意见和反馈

如果你对这个小程序有什么意见和建议，可以告诉我，也可以直接更改源代码，不过请你告诉我你的改进。

谢谢你使用这个工具，并请你常常访问 AnySQL.net 以获得我的最新更新信息。