Inspur K-DB Utility说明书

K-DB 11g

Inspur Electronic Information Industry Co., Ltd.

Inspur Electronic Information Industry Co., Ltd.

中国山东省济南市浪潮路 1036 号 浪潮电子信息产业股份有限公司

Restricted Rights Legend

Trademarks

Open Source Software Notice

Some modules or files of this product are subject to the terms of the following licenses. : OpenSSL, RSA Data Security, Inc., Apache Foundation, Jean-loup Gailly and Mark Adler, Paul Hsieh's hash 本产品的部分文件或模块遵循如下许可证：OpenSSL, RSA Data Security, Inc., Apache Foundation, Jean-loup Gailly and Mark Adler, Paul Hsieh's hash Detailed Information related to the license can be found in the following directory : ${INSTALL_PATH}/license/oss_licenses

相关详细信息，请参阅产品目录中的记载事项：${INSTALL_PATH}/license/oss_licenses

说明书信息

说明书标题: Inspur K-DB　Utility说明书

发行时间: 2015-04-30

软件版本: K-DB 11g

说明书版本: v2.1.1

目录

图片目录

示例目录

关于本说明书

说明书对象

本说明书是针对使用K-DB®（以下统称K-DB)中提供的各种utility的所有数据库用户进行说明。

说明书的前提条件

为了灵活理解本说明书，需要首先了解以下事项。

- 数据库的理解
- RDBMS的理解
- SQL的理解
- Eclipse工具的理解

  Eclipse工具的使用方法请参阅网站(http://www.eclipse.org/documentation)或相关文献。

说明书的限制条件

本说明书不包括实际适用或操作K-DB时所需的全部事项。因此，有关安装、环境设置等操作与管理的内容请参考各产品说明书。

参考

K-DB的安装与环境设置的相关内容请参考"K-DB安装说明书"。

说明书构成

K-DB Utility说明书由以下6个章节构成。

各章的主要内容如下。

- 第1章：kdSQL

  介绍对话型SQL命令处理Utility kdSQL，并记述使用方法。

- 第2章：kdMigrator 2.0

  介绍将现存数据库变更为K-DB的Migration Utility kdMigrator 2.0，并记述使用方法。

- 第3章：kdExport

  介绍抽出保存在K-DB的数据库对象的全部或一部分并保存的Utility kdExport，并记述使用方法。

- 第4章：kdImport

  介绍从通过kdExport生成的Export文件中，将数据库对象保存在K-DB数据库的Utility kdImport，并记述使用方法。

- 第5章：kdLoader

  介绍将大容量的数据一次装载在K-DB数据库的Utility kdLoader，并记述使用方法。

- 第5章：kddv

  介绍了kddv并记述了其使用方法。

- 第7章：Utility API

  介绍从应用程序里调出K-DB时所需的函数。

说明书规约

标示	意义
<AaBbCc123>	程序源代码的文件名
<Ctrl>+C	同时按Ctrl和C键
[Button]	GUI的按键或菜单名
加粗	强调
" "(双引号)	提及其他相关说明书或说明书里的其他章节
'输入项目'	UI画面上输入项的相关说明
超链接	电子邮件账户、网站链接
>	菜单的进行顺序
+----	有下级目录或下级文件
|----	无下级目录或下级文件
参考	参考或注意事项
[图 1.1]	图片名称
[表 1.1]	图表名称
`AaBbCc123`	指令、执行指令后界面上输出的结果、示例代码
{ }	所需参数值
[ ]	选项参数值
|	选择参数值

关于说明书

说明书	说明
K-DB 安装说明书	介绍了安装所需的系统要求以及安装和卸载方法。
K-DB kdCLI说明书	介绍了Call Level Interface，即kdCLI的概念和构成要素、程序结构，以及编辑kdCLI程序时所需的数据类型、函数、error消息的说明书。
K-DB 应用开发员说明书	介绍了如何利用各种应用程序库来开发应用程序。
K-DB External Procedure 说明书	介绍了External Procedure并且记述了如何创建和使用External Procedure。
K-DB JDBC开发员说明书	介绍了如何利用K-DB的JDBC功能来开发应用程序。
K-DB kdESQL/C说明书	介绍了使用C程序语言编写执行数据库操作所需的各种应用程序的方法。
K-DB kdESQL/COBOL说明书	介绍了使用COBOL程序语言编写执行数据库操作所需的各种应用程序的方法。
K-DB kdPSM说明书	介绍了存储过程模块kdPSM的概念和语法、组件以及编程所需的控制结构、组合式类型、子程序、程序包、SQL语句的执行方法和出错处理方法等。
K-DB kdPSM参考说明书	介绍了存储过程模块kdPSM的程序包，记述了这些程序包里的各存储过程(Procedure)和函数的原型(Proto)类型、参数、示例等。
K-DB 管理员说明书	为保证K-DB主要功能的正常运行，在逻辑或物理层面上说明了DBA所要掌握的管理方法，并介绍了辅助管理的各种工具。
K-DB kdAdmin说明书	介绍了用于处理SQL/PSM和DBA提供系统管理功能的基于GUI的工具kdAdmin以及它的安装和使用方法。
K-DB 错误参考说明书	对在使用K-DB过程中可能发生的各种错误进行原因分析并介绍了其解决方法。

说明书	说明
K-DB 参考说明书	介绍了K-DB的运作和使用所需的初始化参数和数据字典、静态视图、动态视图。
K-DB SQL参考说明书	介绍了执行数据库操作或编写引用程序时所需的SQL语句。

联系方式

中国

浪潮电子信息产业股份有限公司
中国山东省济南市浪潮路 1036号
Tel: 400-860-0011
Email: K-DB@inspur.com
官网: www.inspur.com

第1章 kdSQL

本章将介绍kdSQL Utility，并说明其使用方法。

1.1. 概要

kdSQL是处理K-DB®（以下统称K-DB）里提供的SQL语句的对话型Utility。该Utility可以运行SQL查询、数据定义语言(DDL: Data Definition Language)、事务等相关SQL语句。此外，还可以生成并运行PSM程序，DBA可以运行K-DB系统管理的命令。

kdSQL除了这种基本功能以外还提供设置自动提交，操作系统相关命令的运行、输出保存、脚本等功能。特别是脚本功能可以将多种SQL语句及PSM程序还有kdSQL Utility的命令语句使用一个脚本文件来生成，因此提供了便利性。

kdSQL Utility是K-DB的Utililty中使用最频繁的Utility之一，处理运行SQL语句还提供以下功能。

- 一般SQL语句与PSM程序的输入、编辑、保存及运行
- 事务的设置与结束
- 通过脚本运行一揽子操作
- 利用DBA管理数据库
- 数据库的启动与结束
- 外部Utility与程序的运行
- kdSQL Utility的环境设置

1.2. 快速开始

kdSQL Utility在安装K-DB的过程中一起安装，若卸载K-DB则将被一起卸载。

1.2.1. 运行

kdSQL Utility的运行方法如下。

[例 **1.1] kdSQL Utility**的运行

```
$ kdsql

kdSQL 11

Inspur Corporation Copyright (c) 2008-. All rights reserved.

SQL>
```

kdSQL Utility若正常运行，则会如上出现**SQL**提示符。在该提示符里数据库用户可以运行SQL语句。

kdSQL Utility的命令语句的语法如下。

```
kdsql [[options]|[connect_string]|[start_script]]
```

下表是可包含在**options**的选项。

选项	说明
-h, --help	输出帮助界面。
-v, --version	输出版本。
-s, --silent	界面里不输出开始消息和提示符。
-i, --ignore	不运行注册脚本（kdsql.logon）。

connect_string包括要连接K-DB的用户的账户信息，并且如下面的形式指定。

```
username[/password[@connect_identifier]]
```

下面是用于**connect_string**的项目。

选项	说明
username	是用户名，不区分大小写字母。但是在双引号（" "）里输入用户名的情况除外。
password	作为密码，输入时区分大小写字母。
connect_identifier	是拥有数据库连接信息的DSN（Data Source Name），或指定的规则连接明细。

start_script可以与kdSQL Utility的开始的同时一起设置要运行的脚本文件，并且如下面的形式指定。

```
@filename[.ext]
```

下面是用于**start_script**的项目。

选项	说明
filename	是文件名。
ext	是文件的扩展名，没有指定时SUFFIX系统变量里所指定的扩展名为默认值。
parameter	.

1.2.2. 数据库连接

运行kdSQL Utility以后若出现SQL提示符，则可以连接数据库。

kdSQL Utility在开始数据库的会话之前若有要运行的操作时，必须编写kdsql.login文件。如果该文件在当前目录里，则直接运行；若不在，则在设置于环境变量**KD_SQLPATH**的目录里查找。

利用kdSQL Utility连接数据库的方法如下。

[例 **1.2]** 利用**kdSQL Utility**的数据库连接

```
$ kdsql SYS/syspassword

kdSQL 11

Inspur Corporation Copyright (c) 2008-. All rights reserved.

Connected.

SQL>
```

上面的示例中，在UNIX Shell提示符里，运行kdSQL Utility的同时一起输入用户名和密码。

输入用户与密码时有如下规则。

项目	说明
用户名	与Schema Object的名称一样，无需区分大小写。但是在双引号(" ")当中输入用户名时的情况例外。
密码	密码区分大小字符，因此需留意。

如上省略connect_identifier时要连接默认数据库。如果要访问特定数据库时，需要将connect_identifier按照如下两种方式指定。

- DSN(Data Source Name)

  DSN指定定义在kddsn.tbr文件或Windows中的数据原始(ODBC)定义的名称。

- 连接明细

连接明细可以不使用kddsn.tbr文件，直接指名访问信息的如下两种方式来使用。

```
...connect_identifier...
(INSTANCE=(HOST=host)(PORT=port)(DB_NAME=dbname))

...示例...
$ kdsql 'tibero/inspur@(INSTANCE=(HOST=192.168.36.42)(PORT=8629)(DB_NAME=kdb11))'
```

```
...connect_identifier...
host:port/dbname

...示例...
$ kdsql 'tibero/inspur@192.168.36.42:8629/kdb11'
```

1.2.3. 接口

下面是运行kdSQL Utility时的界面。

```
$ kdsql

kdSQL 11

Inspur Corporation Copyright (c) 2008-. All rights reserved.

SQL> CONNECT dbuser
Enter password : dbuserpassword
Connected to K-DB.

SQL>
```

上例中运行kdSQL Utility之后，通过CONNECT命令，用名为dbuser的用户名连接了数据库。kdSQL Utility在这样的文本模式的界面里接收输入，并根据用户的要求输出结果。

参考

本说明书除了特别情况之外，所有SQL语句、PSM程序及kdSQL Utility的命令都由大写字母表示。小写字母作为命令语句的参数将会是用其他参数扩展的情况。

---

kdSQL Utility通过具有以下特性的接口运行。

- kdSQL Utility如果正常运行，则输出SQL提示符。

  在SQL提示符里可以输入SQL语句、PSM程序、kdSQL Utility的命令。

- 可以跳过多个行输入。

  SQL语句和PSM程序可以分离输入和运行。但是，kdSQL Utility的命令语句将在输入的同时被运行。

- 不区分大小写字母。

  正如SQL语句内的字符串数据，除了特殊情况之外都不区分大小写字母。

  例如，下面的两个语句的意义一样。

```
SQL> SET AUTOCOMMIT ON
SQL> set autocommit on
```

1.2.4. 环境设置

若要设置kdSQL Utility的使用环境，需要使用**SET**命令。通过SET命令，可以设置运行SQL查询的结果、输出格式及事务的提交。

下面是SET命令的语法。

```
SET [system_variable] [system_variable_value]
```

参考

详细内容请参考「1.3. 系统变量」。

1.2.5. 结束

若要结束kdSQL Utility，在SQL提示符当中需输入**EXIT**或**QUIT**命令。

```
SQL> EXIT
```

参考

kdSQL提供的命令语句相关详细内容请参考「1.6. 命令」。

1.3. 系统变量

本节将简单介绍kdSQL Utility的系统变量，并且与SET命令语句一起列出语法。kdSQL Utility的系统变量里所要设置的值用SET命令语句设置并且用SHOW命令语句输出。

下面是归纳SET命令语句中可以设置的系统变量的表格。

系统变量	默认值	说明
AUTOCOMMIT	OFF	该系统变量设置是否自动提交。
AUTOTRACE	OFF	该系统变量设置是否输出运行中的查询计划或统计信息。
BLOCKTERMINATOR	"." (0x2E)	该系统变量设置PSM语句中显示最后一个输入符。

系统变量	默认值	说明
COLSEP	" " (0x20)	该系统变量设置显示SQL语句查询结果时区分列的字符。
CONCAT	"." (0x2E)	该系统变量设置显示置换变量名的后端的字符。
DDLSTATS	OFF	该系统变量设置是否显示DDL语句计划或统计信息。
DEFINE	"&"	该系统变量设置定义置换变量时使用的字符。
ECHO	OFF	该系统变量用@或START命令运行脚本文件时，决定是否在画面显示脚本内运行的查询。
EDITFILE	".tbedit.sql"	该系统变量设置EDIT命令语句中使用的文件名默认值。
ESCAPE	OFF	该系统变量设置Escape字符。
EXITCOMMIT	ON	该系统变量设置结束utility时是否提交。
FEEDBACK	0	该系统变量设置是否将SQL语句的运行结果在界面输出。
HEADING	ON	输出查询运行结果时决定是否显示列的标题。
HEADSEP	"|" (0x7C)	设置前言换行字符的系统变量。
HISTORY	50	该系统变量设置命令语句历史大小。
INTERVAL	1	该系统变量设置在LOOP指令中执行各语句后等待的时间。
LINESIZE	80	该系统变量设置一个行里所输出的字符个数。
LONG	80	该系统变量设置标记大于VARCHAR的字符类型数据时的字符数。
NEWPAGE	1	该系统变量设置各页面开始部分需要添加的空行数。
NUMFORMAT	""	该系统变量设置数字型数据的基本列格式。
NUMWIDTH	10	该系统变量设置数字型数据的基本输出长度。
PAGESIZE	24	该系统变量设置界面上所要输出的行个数。
PAUSE	OFF	该系统变量设置输出一个页面后在输出下一个页面之前是否等待用户输入。
RECSEP	WRAPPED	该系统变量指定输出行分隔符的单位。
RECSEPCHAR	" " (0x20)	该系统变量设置用于分隔符的字符。
ROWS	ON	该系统变量设置是否输出查询语句的结果。
SERVEROUTPUT	OFF	该系统变量设置是否输出DBMS_OUTPUT数据包的结果。
SQLPROMPT	"SQL> "	该系统变量设置结束SQL语句的字符。
SQLTERMINATOR	";" (0x3B)	该系统变量设置结束SQL语句的字符。
SUFFIX	"sql"	该系统变量设置文件扩展名的默认值。
TERMOUT	ON	该系统变量设置是否将脚本里运行的命令结果输出在界面上。
TIME	OFF	该系统变量设置是否将当前时间输出在界面上。

系统变量	默认值	说明
TIMEOUT	3	该系统变量设置PING命令中等到服务器响应为止的时间。单位为秒。
TIMING	OFF	该系统变量在每次输出SQL和PSM语句时，设置是否输出执行时间。
TRIMOUT	ON	该系统变量设置是否删除界面的输出行后面的空白。
TRIMSPOOL	OFF	该系统变量设置是否删除Spooling中行后面的空白。
UNDERLINE	"-" (0x2D)	该系统变量设置作为前言下划线来使用的字符。
VERIFY	ON	该系统变量设置运行命令时对应用置换变量的记录设置是否输出。
WRAP	ON	该系统变量设置要输出的行过长时，是否另起一行输出剩余部分。

下面是设置系统变量的示例。

```
SET AUTOCOMMIT ON
SET PAGESIZE 32
SET TRIMSPOOL ON
```

1.3.1. AUTOCOMMIT

设置在运行INSERT, UPDATE, DELETE, MERGE等的SQL语句以后自动提交。

AUTOCOMMIT的详细内容如下。

- 语法

```
SET AUTO[COMMIT] {ON|OFF}
```

项目	说明
ON	自动提交。
OFF	不进行AUTOTRACE。（默认） 设置为OFF的时候，必须指明进行提交。
n	n个INSERT、UPDATE、DELETE、MERGE或PSM块成功执行后执行提交。 n为0时与OFF相同，1时与ON相同。

1.3.2. AUTOTRACE

显示运行中的查询计划或统计信息。必须要有DBA权限或PLUSTRACE权限才能使用。PLUSTRACE作为包含AUTOTRACE所需特权的权限，要由持有DBA权限的用户来生成赋予其他用户。生成脚本为$KD_HOME/scripts/plustrace.sql。

AUTOTRACE的详细内容如下。

- 语法

```
SET AUTOT[RACE] {OFF|ON|TRACE[ONLY]} [EXP[LAIN]] [STAT[ISTICS]] [PLANS[TAT]]
```

项目	说明
ON	根据查询结果和附加选项输出计划信息和统计信息。
OFF	不输出计划和统计信息。（默认值）
TRACE[ONLY]	不显示查询结果，根据附加选项输出计划信息和统计信息。

下面是指定是否显示计划或统计信息的选项。

选项	说明
	如果不指明任何内容，则显示计划和统计信息。
EXP[LAIN]	显示计划信息。
STAT[ISTICS]	显示统计信息。
PLANS[TAT]	对查询显示各节点的执行信息（执行时间、处理行数、执行次数等）。

1.3.3. BLOCKTERMINATOR

设置PSM语句里输入的最后内容。

BLOCKTERMINATOR的详细内容如下。

- 语法

```
SET BLO[CKTERMINATOR] {c|ON|OFF}
```

项目	说明
c	显示结束PSM程序输入的语句。（默认值： "."）
ON	激活BLOCKTERMINATOR。（默认值）
OFF	禁用BLOCKTERMINATOR。

1.3.4. COLSEP

设置区分运行SELECT语句后输出的多个列的字符。

COLSEP的详细内容如下。

- 语法

```
SET COLSEP {text}
```

项目	说明
text	区分列的字符串。（默认值：" "）

1.3.5. CONCAT

设置显示置换变量名结尾的字符。

CONCAT的详细内容如下。

- 语法

```
SET CON[CAT] {c|ON|OFF}
```

项目	说明
c	显示对置换变量名的阶数字符。（默认值： "."）
ON	激活CONCAT。（默认值）
OFF	禁用CONCAT。

1.3.6. DDLSTATS

显示运行中的DDL语句的计划或统计信息。但是需要注意的是必须同时激活AUTOTRACE功能。

DDLSTATS的详细内容如下。

- 语法

```
SET DDLSTAT[S] {OFF|ON}
```

项目	说明
ON	禁用DDLSTATS。

项目	说明
OFF	激活DDLSTATS。（默认值）

1.3.7. DEFINE

设置定义置换变量时的字符。

DEFINE的详细内容如下。

- 语法

```
SET DEF[INE] {c|ON|OFF}
```

项目	说明
c	显示置换变量的字符。（默认值："&"）
ON	激活DEFINE。（默认值）
OFF	禁用DEFINE。

1.3.8. DESCRIBE

设置通过DESCRIBE指令显示对象明细的阶段。

DESCRIBE的详细内容如下。

- 语法

```
SET DESCRIBE DEPTH {n}
```

项目	说明
n	递归性输出的阶段。（默认值：10）

1.3.9. ECHO

设置用@或START命令运行脚本文件时脚本内运行的查询是否输出在画面上。

ECHO的详细内容如下。

- 语法

```
SET ECHO {OFF|ON}
```

项目	说明
ON	激活ECHO功能。
OFF	禁用ECHO功能。（默认值）

1.3.10. EDITFILE

设置EDIT命令中要使用的默认文件名。省略扩展名时使用FILEEXT所设置的值。

EDITFILE的详细内容如下。

- 语法

```
SET EDITF[ILE] filename[.ext]
```

项目	说明
filename[.ext]	EDIT命令语句中使用的文件名。（默认值：.tbedit.sql）

1.3.11. ESCAPE

为了忽略DEFINE中定义的置换变量字符设置Escape字符。激活时若将Escape字符写在'&<字符串>'前面，则置换变量不会被识别。

ESCAPE的详细内容如下。

- 语法

```
SET ESC[APE] {c|ON|OFF}
```

项目	说明
c	Escape字符。(默认值："\")
ON	激活Escape字符。
OFF	禁用Escape字符。

1.3.12. EXITCOMMIT

设置结束Utility时是否提交。

EXITCOMMIT的详细内容如下。

- 语法

```
SET EXITC[OMMIT] {ON|OFF}
```

项目	说明
ON	激活EXITCOMMIT。（默认值）
OFF	禁用EXITCOMMIT。

1.3.13. FEEDBACK

设置是否将SQL语句的运行结果输出在界面里。

FEEDBACK的详细内容如下。

- 语法

```
SET FEED[BACK] {ON|OFF}
```

项目	说明
ON	激活FEEDBACK。（默认值）
OFF	禁用FEEDBACK。

1.3.14. HEADING

决定输出查询运行结果时是否显示列的标题。

HEADING的详细内容如下。

- 语法

```
SET HEA[DING] {ON|OFF}
```

项目	说明
ON	激活HEADING。（默认值）
OFF	禁用HEADING。

1.3.15. HEADSEP

设置前言换行字符的系统变量。

HEADSEP的详细内容如下。

- 语法

```
SET HEADS[EP] {c|ON|OFF}
```

项目	说明
c	换行符。（默认值："|"）
ON	激活HEADSEP。（默认值）
OFF	禁用HEADSEP。

1.3.16. HISTORY

设置命令语句的历史大小。

HISTORY的详细内容如下。

- 语法

```
SET HIS[TORY] {n|ON|OFF}
```

项目	说明
n	命令语句历史的大小。（默认值：50）
ON	激活历史功能。（默认值）
OFF	禁用历史功能。

1.3.17. INTERVAL

该系统变量设置在LOOP指令中执行各语句后等待的时间。

INTERVAL的详细内容如下。

- 语法

```
SET INTER[VAL] {n}
```

项目	说明
n	作为待机时间，单位为秒。（默认值：1）

1.3.18. LINESIZE

设置界面上一个行的长度。行长度的最小值为1，最大值随操作系统的不同而不同。

LINESIZE的详细内容如下。

- 语法

```
SET LIN[ESIZE] {n}
```

项目	说明
n	界面上一个行的长度。（默认值：80）

1.3.19. LONG

读取CLOB或LONG类型的数据，设置要输出的长度。长度不能超过2,000,000,000。

LONG的详细内容如下。

- 语法

```
SET LONG {n}
```

项目	说明
n	是大容量数据的默认输出长度。（默认值：80）

1.3.20. NEWPAGE

设置个页面开始部分需要添加的空行数。

NEWPAGE的详细内容如下。

- 语法

```
SET NEWP[AGE] {1|n|NONE}
```

项目	说明
n	空行个数。（默认值：1）

1.3.21. NUMFORMAT

设置NUMBER类型的默认列格式。用COLUMN命令将除掉FORMAT定义来应用在数字型列上。

NUMFORMAT的详细内容如下。

- 语法

```
SET NUMF[ORMAT] {fmt_str}
```

项目	说明
fmt_str	NUMBER类型数据的默认列格式。（默认值：""） 详细内容请参阅「1.7. 列的格式化」的数字型说明。

1.3.22. NUMWIDTH

设置输出NUMBER类型的长度。不能超过LINESIZE。

NUMWIDTH的详细内容如下。

- 语法

```
SET NUM[WIDTH] {n}
```

项目	说明
n	NUMBER类型数据的默认输出长度。（默认值：10）

1.3.23. PAGESIZE

设置包含kdSQL Utility输出内容的各页面上的行个数。

PAGESIZE的详细内容如下。

- 语法

```
SET PAGES[IZE] {n}
```

项目	说明
n	一个页面上的行个数。（默认值：24）

1.3.24. PAUSE

设置输出一个页面后在输出下一个页面之前是否等待用户输入。

PAUSE的详细内容如下。

- 语法

```
SET PAUSE {ON|OFF}
```

项目	说明
ON	激活PAUSE设置。
OFF	禁用PAUSE设置。（默认值）

1.3.25. RECSEP

该系统变量指定输出行分隔符的单位。

RECSEP的详细内容如下。

- 语法

```
SET RECSEP {WR[APPED]|EA[CH]|OFF}
```

项目	说明
WRAPPED	行为WRAPPING时输出分隔符。（默认值）
EACH	以所有行单位输出分隔符。
OFF	禁用RECSEP。

1.3.26. RECSEPCHAR

该系统变量设置用于分隔符的字符。该分隔符以LINESIZE值反复输出。

RECSEPCHAR的详细内容如下。

- 语法

```
SET RECSEPCHAR {c}
```

项目	说明
c	行分隔符。（默认值："）

1.3.27. ROWS

该系统变量设置是否输出查询语句的结果。

ROWS的详细内容如下。

- 语法

```
SET ROWS {ON|OFF}
```

项目	说明
ON	激活ROWS。（默认值）
OFF	禁用ROWS。

1.3.28. SERVEROUTPUT

设置是否输出DBMS_OUTPUT数据包的结果。

SERVEROUTPUT的详细内容如下。

- 语法

```
SET SERVEROUT[PUT] {ON|OFF} [SIZE n]
```

项目	说明
ON	激活SERVEROUTPUT。
OFF	禁用SERVEROUTPUT。（默认值）
n	指定SERVEROUTPUT缓存大小。（默认值：1000000）

1.3.29. SQLPROMPT

设置界面上的提示符。

SQLPROMPT的详细内容如下。

- 语法

```
SET SQLP[ROMPT] {prompt_string}
```

项目	说明
prompt_string	用于提示的字符串。（默认值：SQL>） 如果将该字符串括上中括号（{}）则将被识别为环境变量。例如若指定为'{ISQL_PROMPT}'，则环境变量$ISQL_PROMPT的值被换置使用为提示。此时环境变量的名称区分大小写。

1.3.30. SQLTERMINATOR

设置结束SQL语句的字符。

SQLTMINATOR的详细内容如下。

- 语法

```
SET SQLT[MINATOR} {c|ON|OFF}
```

项目	说明
c	提醒SQL语句结束的字符。（默认值：";"）
ON	激活SQLTERMINATOR。
OFF	禁用SQLTERMINATOR。

1.3.31. SUFFIX

该系统变量设置文件扩展名的默认值。

SUFFIX的详细内容如下。

- 语法

```
SET SUF[FIX] {extension}
```

项目	说明
extension	默认使用的文件扩展名。（默认值：sql）

1.3.32. TERMOUT

设置是否将脚本里运行的命令结果输出在界面上。

TERMOUT的详细内容如下。

- 语法

```
SET TERM[OUT] {ON|OFF}
```

项目	说明
ON	激活TERMOUT。（默认值）
OFF	禁用TERMOUT。

1.3.33. TIME

设置是否将当前时间输出在界面上。

TIME的详细内容如下。

- 语法

```
SET TI[ME] {ON|OFF}
```

项目	说明
ON	激活TIME。
OFF	禁用TIME。（默认值）

1.3.34. TIMEOUT

设置PING命令中等到服务器响应为止的时间。单位为秒。

TIMEOUT的详细内容如下。

- 语法

```
SET TIMEOUT {n}
```

项目	说明
n	等待服务器响应的时间。（默认值：3）

1.3.35. TIMING

设置在每次输出SQL和PSM语句时，设置是否输出执行时间。

TIMING的详细内容如下。

- 语法

```
SET TIMI[NG] {ON|OFF}
```

项目	说明
ON	激活TIMING。
OFF	禁用TIMING。（默认值）

1.3.36. TRIMOUT

设置是否删除界面的输出行后面的空白。

TRIMOUT的详细内容如下。

- 语法

```
SET TRIM[OUT] {ON|OFF}
```

项目	说明
ON	激活TRIMOUT。（默认值）
OFF	禁用TRIMOUT。

1.3.37. TRIMSPOOL

设置是否删除Spooling中行后面的空白。

TRIMSPOOL的详细内容如下。

- 语法

```
SET TRIMS[POOL] {ON|OFF}
```

项目	说明
ON	激活TRIMSPOOL。
OFF	禁用TRIMSPOOL。（默认值）

1.3.38. UNDERLINE

该系统变量设置作为前言下划线来使用的字符。

UNDERLINE的详细内容如下。

- 语法

```
SET UND[ERLINE] {c|ON|OFF}
```

项目	说明
c	下划线字符。（默认值："-"）
ON	激活UNDERLINE。（默认值）
OFF	禁用UNDERLINE。

1.3.39. VERIFY

设置运行命令时对应用置换变量的记录设置是否输出。

VERIFY的详细内容如下。

- 语法

```
SET VER[IFY] {ON|OFF}
```

项目	说明
ON	激活VERIFY。（默认值）
OFF	禁用VERIFY。

1.3.40. WRAP

设置要输出的行过长时，是否另起一行输出剩余部分。

WRAP的详细内容如下。

- 语法

```
SET WRA[P] {ON|OFF}
```

项目	说明
ON	激活WRAP。（默认值）
OFF	禁用WRAP。

1.4. 基本功能

kdSQL Utility主要使用的功能是直接输入并运行SQL语句或PSM程序。在本节中先说明命令语句的输入与运行之后，再说明其他功能。

1.4.1. 命令的输入

kdSQL Utility 命令提示中的输入大体可分为SQL语句、PSM程序、Utility三种命令。命令的输入方法大体类似。

下面将依次说明输入各命令的方法。

SQL语句的输入

输入SQL语句的方法如下。

- 一般输入

  一般SQL语句在kdSQL Utililty 提示中输入。一个SQL语句可以隔多个行输入。如果要取消SQL语句的输入，则在空行里输入。

- 切换行

  一个SQL语句间隔多个行输入时，可以在不是连续字符串的任意位置更改行。一般情况下，为了方便读取和易于更改，最好以子句为单位更改行输入。

- 注释（comment）的插入

输入SQL语句途中可以插入注释。注释可从两个减号（--）开始，包含于该行的结束。注释可以以它自身构成一个行，并且在一个行中可以跟在其他字符串后面。

- 利用以前所保存的语句输入

  输入的SQL语句保存在kdSQL Utility的SQL缓冲区。因此，为了输入相同或类似的SQL语句，可以使用以前保存的语句。如果要按需要更改以前的语句，则新的语句已被输入并保存在SQL缓冲区当中。

  在SQL缓冲区里保存一个SQL语句或PSM程序。按操作体系点击键盘的左侧向上箭头（↑）或向下箭头（↓），则可以重新调出以前输入的语句。每次点击按键时，以前保存的语句将按各行显示，以前保存的SQL语句不仅可以调出全部还可以调出一部分。

下面是在kdSQL Utility输入SQL语句的示例。

```
SQL> SELECT ENAME, SALARY, ADDR
    FROM EMP
    -- this is a comment.
    WHERE DEPTNO = 5;

SQL>
```

PSM程序的输入

PSM程序有多数的SQL语句或PSM语句构成，各SQL语句以分号（;）结束。若开始输入PSM程序，kdSQL会自动转换为PSM序输入模式。在PSM程序输入模式当中SQL语句的输入结束时，SQL语句不会被个别运行。

为将kdSQL Utility转换为PSM程序输入模式而创建的语句当中有DECLARE，BEGIN等无名的块（anonymous block）和生成各存储过程函数、触发器的CREATE（OR　REPLACE）PROCEDURE、FUNCTION、TRIGGER。

PSM程序里的输入方法与一般SQL语句的情况类似。

下面是输入PSM程序的方法。

- 一般输入

  可以间隔多个行输入PSM程序。若要取消SQL语句，即使输入了空行还要再输入块结束字符(BLOCK TERMINATOR)。块结束字符的默认值为结束符（.）。块结束字符必须只有相关字符输入在一个行里，不能与其他字符串一起输入。

- 利用以前保存的语句输入

  一次输入的程序可以保存在SQL缓冲区里重复使用。

- 注释的插入

  用与SQL语句相同方式输入注释。

下面是在kdSQL Utility中输入无名块的示例。

```
SQL> DECLARE
    deptno NUMBER(2);
    BEGIN
        deptno := 5;
        UPDATE EMP SET SALARY = SALARY * 1.05
        WHERE DEPTNO = deptno;
        -- this is a comment.
    END;
    .
SQL>
```

在上面示例中，在块内插入了一行注释并且在END语句下面第9个行输入块结束字符（.），结束了PSM程序的输入。在最后行中可看到没有其他字符或字符串，只有一个块结束字符形成一个行。

参考

PSM使用的详细内容请参考"K-DB kdPSM说明书"。

---

kdSQL Utility命令语句的输入

kdSQL Utility命令语句中包括SQL运行的相关命令语句或为其他K-DB数据库管理命令语句。kdSQL Utility的命令语句的相关详细内容请参考「1.6. 命令」。

1.4.2. 命令语句的运行

输入运行kdSQL Utility命令提示符当中的方法有以下三种。

- 保存在SQL缓冲区的SQL语句或PSM程序的运行

  SQL缓冲区里只保存了最近输入的SQL语句或一个PSM程序。为了运行这种SQL语句或PSM程序，需要共同输入RUN或/命令语句。

- SQL语句的运行

  输入所有语句并用分号（;）结束时，SQL语句则马上运行。

- 与SQL缓冲区里的保存的同时运行

  SQL语句或PSM程序输入结束以后，若要与SQL缓冲区里的保存同时运行，则输入/命令。此时，与结束符（.）相同，其自身将成为一个行。

kdSQL　Utility的命令语句与SQL语句或PSM程序不同，没有专门用于运行的命令语句，并且也不保存在SQL缓冲区里。kdSQL Utility的命令语句将在输入结束的同时运行。

下面是运行保存在SQL缓冲区的SQL语句的示例。

```
SQL> SELECT ENAME, SALARY, ADDR
    FROM EMP
    -- this is a comment.
    WHERE DEPTNO = 5;
......运行结果 ①......
SQL> /
......运行结果 ②......
SQL>
```

上例当中在第一个SQL 提示符里输入SQL语句并用分号（;）结束使直接运行。在第二个SQL 提示符里输入/，运行了保存在SQL缓冲区里的SQL语句。在SQL缓冲区里保存了最新输入的SQL语句，与在第一个SQL 提示符里输入的SQL语句相同的语句被重新运行。因此运行结果①与运行结果②输出相同的结果。

下面是利用反斜杠（/）运行前面所显示的SQL语句的示例。此时，SQL语句的最后部分不输入分号（;）。

```
SQL> SELECT ENAME, SALARY, ADDR
    FROM EMP
    -- this is a comment.
    WHERE DEPTNO = 5
    /
......运行结果......
SQL>
```

1.4.3. 其他功能

本节将依次说明kdSQL Utility的基本功能中主要使用的注释插入、自动提交、操作系统命令运行、保存输出内容等功能。

注释的插入

注释主要由以下两种方式插入。

- 使用/* ... */

  使用/* ... */的方法与在C或C++编程语言里所使用的方法相同。在kdSQL Uriliry当中被/*和*/包含的部分不被识别为注释处理。

  该注释不会被重复使用。即，/* …… */注释里不会包含于另一个/* …… */注释里。

- 使用双减号（--）

  若利用双减号（--），则在各行当中将从双减号（--）开始到行的结束都会被视为注释而忽略掉。

  该注释与使用/* …… */的方法相同，会在任意位置出现，与为结束kdPSM程序输入的结束符（.）不能在同一个行里。

  因此下面编写的脚本文件在运行中会发生错误。

```
(PSM 程序)
.-- 错误的注释
RUN
```

自动提交

更新为SQL语句的内容在事务被提交之前不会在数据库当中永久反映。一个事务大部分由多个SQL语句构成，更新为SQL语句的内容不会直接反映在数据库里。

kdSQL Utility当中每次运行SQL语句时，可以设置自动运行COMMIT语句或停止该功能。默认值为不使用自动提交。

通过SET AUTOCOMMIT命令可以做这种设置，并且若要输出当前自动提交来确认时，使用SHOW AUTOCOMMIT命令语句。

操作系统命令语句的运行

若要在kdSQL Utility开始状态下运行操作系统命令，则需要输入HOST命令语句。

下面是展列扩展名为.sql的所有脚本文件的示例。使用!命令代替HOST命令也可以进行相同的操作。

```
SQL> HOST dir *.sql
..... 操作系统命令语句运行结果 .....
SQL>
```

运行操作系统命令之后，kdSQL Utility的提示符再次显示，可以输入kdSQL Utility的命令。

若要省略HOST命令语句或!命令语句后面的语句，则将被输出操作系统命令提示符。为了重新回到kdSQL Utility，则输入EXIT。

```
SQL> !
$ dir *.sql
..... 操作系统命令运行结果 .....
$ EXIT
SQL>
```

输出内容的保存

若要将kdSQL Utility中输入或输出的所有内容保存为文本文件，则使用SPOOL命令语句。若使用SPOOL命令语句，则用户输入的SQL语句或PSM程序、kdSQL Utility的命令，甚至插叙结果与程序运行结果、kdSQL Utility提示符也会保存在文件里。

若运行SPOOL命令语句，则从后面的行开始保存在文件里。若要停止该功能，则输入SPOOL OFF。若输入SPOOL OFF，则从下一个行开始不会保存在文件里。

下面是使用SPOOL命令语句的示例。若已有SPOOL命令语句当中使用的save.txt文件，则将会覆盖原来的文件之前的内容将会消失。

```
SQL> SPOOL save.txt
Spooling is started.
SQL> SELECT
    *
    FROM DUAL;

DUMMY
-----
X

1 row selected.

SQL> SPOOL OFF
Spooling is stopped: save.txt
```

下面是在之前示例当中保存在save.txt文件中的内容。

```
SQL> SELECT
    *
    FROM DUAL;

DUMMY
-----
X

1 row selected.

SQL> SPOOL OFF
```

用户输入的SQL语句、查询结果、最后的SPOOL OFF命令都保存在文件里。

1.5. 高级功能

本节将对相比kdSQL Utility的基本功能更高级的用户使用功能，利用脚本的成批操作运行与管理K-DB系统的DBA的功能进行说明。

1.5.1. 脚本功能

脚本是指使用一次命令运行成批操作的SQL语句与PSM程序、kdSQL Utility命令语句的集合。kdSQL Utility里若运行脚本，则其包含的所有命令语句将会依次运行。

脚本的生成

脚本文件可以从外部生成、编辑并在kdSQL Utility里运行，也可以在运行kdSQL Utility之后调出外部编辑器进行生成与编辑。调出外部编辑器时可以设置要使用何种编辑器。

下面是通过外部编辑器使用vi的示例。

```
$ export KD_EDITOR=vi
```

为了利用外部编辑器编辑脚本文件而使用EDIT命令语句。需要与EDIT命令语句一起提示文件名。若扩展名与SUFFIX系统变量相同，则可以省略。

下面是为了编辑脚本文件run.sql而调出外部编辑器的示例。

```
SQL> EDIT run
```

在脚本文件里输入SQL语句、PSM程序、kdSQL Utility命令的方法如下。

- 一般输入方法

  与kdSQL的命令提示符里输入的方法基本相同，可以间隔多个行输入。

- SQL语句与PSM程序的结束

  SQL语句必须在语句结束位置输入分号（;），PSM程序的最后行必须只用结束符（.）结束。

- 注释的插入

  可以在脚本文件里插入注释。

若运行脚本，SQL语句将会被直接运行，而PSM程序需要输入RUN或/命令语句来运行。

下面是对表EMP的相关几项操作运行的脚本文件示例。行之间可以有空白。

```
-- SQL 文章
SELECT ENAME, SALARY, ADDR FROM EMP
    WHERE DEPTNO = 5;
UPDATE EMP SET SALARY = SALARY * 1.05
    WHERE DEPTNO = 5;

-- PSM 程序
DECLARE
```

```
    deptno NUMBER(2);
BEGIN
    deptno := 20;
    UPDATE EMP SET SALARY = SALARY * 1.05
    WHERE DEPTNO = deptno;
END;

RUN -- 执行PSM程序
/* 反映最终所更新的内容。 */
COMMIT;
```

脚本的运行

若要运行脚本文件，则使用START或@命令语句。与命令语句一起指定文件名，扩展名与SUFFIX系统变量（sql）相同时省略也无妨。

下面两行是运行脚本文件run.sql的示例，结果相同。

```
SQL> START run
SQL> @run
```

脚本文件里可以运行一个以上的其他脚本文件。即，脚本文件里可以包含START或@命令语句。递归地运行脚本文件时，必须使它不能进行无限循环。

开始kdSQL Utility时，若使用@命令，在开始的同时可以运行脚本。这种方法在操作系统里运行部署程序时需要。

下面是开始 kdSQL Utility的同时运行脚本文件run.sql的示例。

```
$ kdsql dbuser/dbuserpassword @run
```

或利用如下命令来运行脚本。

```
$ kdsql dbuser/dbuserpassword < run.sql
```

1.5.2. 用于DBA的功能

通过kdSQL Utility可以运行DBA功能。若要通过kdSQL Utility运行DBA功能，首先具有DBA权限的用户登录K-DB。

下面是具有DBA权限的SYS用户登录的示例。

```
$ kdsql sys/syspassword
```

开始kdSQL Utility以后也可以通过DBA连接。此时使用CONNECT命令，像前面一样，具有DBA权限的用户连接便可。

下面是利用CONNECT命令连接DBA的示例。

```
SQL> CONNECT sys/syspassword
```

面是DBA通过kdSQL Utility可以运行的功能。

- K-DB的结束

  具有DBA权限的用户通过kdSQL命令，KDDOWN结束K-DB。

- 用户输入的字符替换

  在查询里由&开始的token接收相关token的用户的输入来进行替换。反复运行的查询只在特定部分改变时适用。

1.5.3. 访问信息加密功能

kdSQL Utility将数据库访问信息(connect_string)以加密文件(wallet)保存后使用的功能。在指定在环境变量ISQL_WALLET_PATH上的路径文件上可以创建连接kdSQL Utility的数据库信息创建为加密文件来在下次访问时使用。

加密文件的生成

利用kdSQL Utility访问特定数据库之后通过SAVE CREDENTIAL命令来生成加密文件。

下面是将ISQL_WALLET_PATH值设置为当前路径的wallet.dat文件后加密访问的数据库信息来生成的示例。

```
$ export ISQL_WALLET_PATH=./wallet.dat
$ kdsql

kdSQL 11

Inspur Corporation Copyright (c) 2008-. All rights reserved.

Can't login with the wallet file.
Login the database and SAVE CREDENTIAL again.

Enter Username: dbuser
Enter Password: dbuserpassword
Connected to K-DB.
```

```
SQL> SAVE CREDENTIAL
Complete to generate the wallet file.
```

运行kdSQL Utility之前设置ISQL_WALLET_PATH环境变量，因此虽然试图解密./wallet.dat文件内容，但相应文件不存在时会发生如上错误。

重新正常访问数据库之后通过SAVE CREDENTIAL命令生成./wallet.dat文件。

下面是不设置ISQL_WALLET_PATH环境变量加密当前路径的wallet.dat文件访问数据库信息来加密生成的示例。

```
$ kdsql

kdSQL 11

Inspur Corporation Copyright (c) 2008-. All rights reserved.

SQL> CONN dbuser/dbuserpassword
Connected to K-DB.

SQL> SAVE CREDENTIAL "./wallet.dat"
Complete to generate the wallet file.
```

作为SAVE CREDENTIAL的参数给予./wallet.dat文件路径时，加密访问数据库的信息来生成./wallet.dat文件。

加密文件的使用

运行kdSQL Utility之前作为环境变量ISQL_WALLET_PATH的值，指定在上面生成的加密文件(wallet)时可以重新利用加密文件生成前的访问信息。

下面是使用ISQL_WALLET_PATH值的文件来访问数据库的示例。

```
$ kdsql

kdSQL 11

Inspur Corporation Copyright (c) 2008-. All rights reserved.

SQL> CONN dbuser/dbuserpassword
Connected to K-DB.
```

参考

为了使用加密文件必须要设置ISQL_WALLET_PATH。

加密文件只能在生成的kdSQL Utility当中使用。如果需要将相应文件在其他kdSQL Utility当中使用时，需要通过此过程来重新生成加密文件。

Windows环境上无法使用上面加密文件的生成及使用功能。

1.6. 命令

本节将详细说明kdSQL Utility提供的命令。

下面是表示kdSQL Utility的命令语法的示例。

```
COM[MAND] param {choice1|choice2} [option] [arg]*
```

以上面的示例为基准，解释kdSQL Utility里所使用的命令的语法的方法如下。

项目	说明
大括号（[ ]）	大括号（[ ]）里所包含的内容没有输入，也可以运行命令。 在上面的示例中，COMMAND语句的后部分（MAND）与option，arg不包含于命令提示符里。
中括号（{ }）	包含于（{ }）的内容必须输入，才能运行命令。 上面示例中choice1与choice2位于中括号（{}）里，由竖直杠（|）分离，因此其中的一个必须包含于命令提示符里。
竖直杠（|）	由竖直杠（|）分离的内容中选择其中一个。
星号（*）	星号（*）指明的内容有可能不被包括，也可能被包含多次。 在上面的示例中arg、大括号（[]）后面有星号（*），因此arg可能不被包含也可能被包含一次以上。
斜体字	指明为斜体的内容按命令由适当的字符串代替。
大小字母	命令不区分大小写字母。

下面所列的三个命令按上面的命令语法表示全部有效。

```
COMMAND param choice1
COM param choice1 option
COM param choice2 arg1 arg2 arg3
```

kdSQL Utility命令中包括SQL语句的运行和数据库管理所需命令。各命令以字母顺序排列，按语法（syntax）、参数、示例的顺序说明。

kdSQL 中可以使用的命令如下。

命令	说明
!	是运行操作系统的命令。

命令	说明
	与HOST命令一致。
%	重新执行在历史缓存上保存的命令的命令。
@, @@	是运行脚本的命令。 与START命令一致。
/	是运行当前SQL缓冲区内的SQL语句和PSM程序的命令。 与RUN命令一致。
ACCEPT	是接收用户的输入设置置换变量的属性的命令。
ARCHIVE LOG	输出Redo日志文件信息的命令。
CHANGE	是在SQL缓冲区的当前行里找到模式符号，变换为所给字符的命令。
CLEAR	是初始化或取消所设置的选项的命令。
COLUMN	是列的输出属性的命令。
CONNECT	通过特定用户ID连接数据库的命令。
DEFINE	是定义或暑促置换变量的命令。
DEL	是取消SQL缓冲区里保存的行的命令。
DESCRIBE	是输出所指定对象的列信息的命令。
DISCONNECT	是取消连接当前数据库的命令。
EDIT	是利用外部编辑器编辑特定文件和SQL缓冲区的内容的命令。
EXECUTE	是运行单一PSM语句的命令。
EXIT	是结束kdSQL Utility的命令。 与QUIT命令一致。
HELP	是输出帮助的命令。
HISTORY	是输出所运行的命令的历史的命令。
HOST	是运行操作系统命令的命令。 与!命令一致。
INPUT	是在SQL缓冲区的当前行后面添加新行的命令。
LIST	是输出SQL缓冲区内的特定内容的命令。
LOADFILE	是以可以识别Oracle的SQL*Loader工具的形式保存K-DB的命令。
LOOP	无限重复执行单一指令的命令。
LS	是输出当前用户生成的数据库对象的命令。
PAUSE	是停止运行，直到用户点击<Enter>键的命令。

命令	说明
PING	输出对特定数据库是否可以访问的状态的命令。
PRINT	是输出用户定义的置换变量的值的命令。
PROMPT	是直接在界面里输出用户定义的SQL语句或空行的命令。
QUIT	是结束kdSQL Utility的命令。 与EXIT命令一致。
RUN	是运行当前SQL缓冲区里的SQL语句或PSM程序的命令。 与/命令一致。
SAVE CREDENTIAL	加密数据库访问信息来保存在文件上的命令。
SET	是设置kdSQL Utility的系统变量的命令。
SHOW	是输出kdSQL Utlity的系统变量的命令。
SPOOL	是开始或结束花面狸输出的内容保存在所有外部文件里的命令。
START	是运行脚本文件的命令。 与@命令一致。
KDDOWN	是结束K-DB的命令。
UNDEFINE	是删除一个以上的置换变量的命令。
VARIABLE	是说明用户所定义的置换变量的命令。
WHENEVER	定义发生错误是的运行。

1.6.1. !

在kdSQL Utility内运行操作系统。可以代替!命令使用HOST命令。

!命令的详细内容如下。

- 语法

```
! [command]
```

选项	说明
	若不用操作系统的命令而只输入!命令，则可以在操作系统的命令提示符的外部多次输入操作系统的命令。 此时，如要再回到kdSQL Utility，则输入EXIT命令。
command	是操作体系的命令。

- 示例

```
SQL> ! dir *.sql
SQL> !
```

1.6.2. %

在kdSQL内重新执行在历史缓存上保存的命令的命令。

%命令的详细内容如下。

- 语法

```
% number
```

选项	说明
number	保存在历史缓存上的命令编号。

- 示例

```
SQL> history
  1: set serveroutput on
  2: set pagesize 40
  3: select 1 from dual;
SQL> %3
```

```
          1
-----------
          1

1 row selected.
```

1.6.3. @, @@

运行脚本文件。脚本文件若有在SUFFIX系统变量登录的扩展名，则可以没有扩展名也可以制定脚本文件的名称。kdSQL Utility在当前目录内查找所指定的脚本文件。

脚本文件的运行以前，设置为SET命令的系统变量在运行脚本文件时也有效。若在脚本文件里运行EXIT或QUIT命令，则结束kdSQL Utility。

可以使用START命令代替@命令。

@命令的详细内容如下。

- 语法

```
@ {filename}
```

项目	说明
filename	是脚本文件的名称。

- 示例

```
SQL> @ run
SQL> @ run.sql
```

1.6.4. /

运行当前SQL缓冲区里的SQL语句或PSM程序。

可以使用RUN命令代替/命令。

/命令的详细内容如下。

- 语法

```
/
```

- 示例

```
SQL> SELECT * FROM DUAL;
..... SQL语句运行结果 .....
SQL> /
..... 与上面相同的结果 .....
```

1.6.5. ACCEPT

接收用户的输入设置变量的属性。此时所设定的置换变量值，在以后用户输入的SQL语句或PSM程序里与**&variable**一致的单词存在时，会被自动重置。

ACCEPT命令的详细内容如下。

- 语法

```
ACC[EPT] variable [FOR[MAT] format] [DEF[AULT] default] [PROMPT statement|NOPR[OMPT]]
```

项目	说明
variable	是要保存的置换变量的名称。若不存在，则新生成。

下面是ACCEPT命令的选项的有关说明。

选项	说明
FOR[MAT] format	置换变量的格式。如果置换变量的值和格式不一致时发生错误。
DEF[AULT] default	若没有从用户可接收的值，可以代替使用的值。
PROMPT statement	在从用户接收置换变量的值的输入之前，界面里输出提示符。
NOPR[OMPT]	不输出提示符，等待用户的输入。

- 示例

```
SQL> ACCEPT name PROMPT 'Enter name : '
Enter name : 'John'
SQL> SELECT &name FROM DUAL;
At line 1, column 8
old value : SELECT &name FROM DUAL
new value : SELECT 'John' FROM DUAL

'JOHN'
------
John

1 row selected.
```

```
SQL>
```

1.6.6. ARCHIVE LOG

输出Redo日志文件信息。

ARCHIVE LOG命令的详细内容如下。

- 语法

```
ARCHIVE LOG LIST
```

- 示例

```
SQL> ARCHIVE LOG LIST

NAME                           VALUE
------------------------------ ------------------------------------------------
Database log mode              Archive Mode
Archive destination            /home/kdb/database/kdb/archive
Oldest online log sequence     300
Next log sequence to archive   302
Current log sequence           302

SQL>
```

1.6.7. CHANGE

在SQL缓冲区里的语句的当前行里找到第一个old模式，变换为new模式。一般上，最后运行的SQL语句的当前行为最后行。若要变更当前行，则请参考示例。

CHANGE命令的详细内容如下。

- 语法

```
C[HANGE] delim old [delim [new [delim [option]]]]
```

项目	说明
delim	是数字除外的区分符。 必须使用没有old或new模式的字符。

项目	说明
old	是要改变的模式。 不区分大小写字母。 此时使用的单词除了一般性单词（如，dual，ksc911等）以外，还可以使用显示任意模式**...**。使用方法请参考示例。

下面是CHANGE命令的选项的相关说明。

选项	说明
delim	是数字除外的区分符。 必须使用没有old或new模式的字符。
new	新重置的模式。
option	– g：在当前行里改变所有模式。 – c：在当前行里随用户的选择改变。 – a：在所有语句里改变所有模式。

- 示例

当前行的默认值一直指最后行，因此在第二个行里的DUAL变换为T。

```
SQL> SELECT *
    FROM DUAL;
..... SQL 执行结果 .....
SQL> C/DUAL/T
    FROM T
SQL>
```

输入想要的行号码，改变当前行。

```
SQL> 5
    5 WHERE ROWNUM < 5 AND
```

可以使用...显示任意模式。此时，...可以在前、中、后插入。

```
SQL> CHANGE /RE...AND/RE ROWNUM >= 5 AND/
    5 WHERE ROWNUM >= 5 AND
SQL> CHANGE /...AND/WHERE ROWNUM < 3/
    5 WEHRE ROWNUM < 3
```

```
SQL> CHANGE /WHE.../WHERE ROWNUM < 5 AND/
    5 WHERE ROWNUM < 5 AND
```

要指定a选项时，在所有语句中找到所给的模式所有改变。因此，在第一个行里的*改变为字符串。

```
SQL> SELECT *
    FROM DUAL;
..... SQL 执行结果 .....
SQL> C/*/'replaced'/a
    SELECT 'replaced'
    FROM DUAL;
SQL>
```

1.6.8. CLEAR

初始化或消除所设置的选项。

CLEAR命令的详细内容如下。

- 语法

```
CL[EAR] [option]
```

选项	说明
option	– BUFF[ER]：删除SQL缓冲区里的所有内容。 – SCR[EEN]：删除界面里的所有内容。 – COL[UMNS]：初始化所登录的所有列的输出属性。

- 示例

```
SQL> CLEAR BUFFER
SQL buffer is cleared
SQL> CLEAR SCREEN
SQL> CLEAR COLUMNS
```

1.6.9. COLUMN

指定列的输出属性。如果只明示了列名称，则输出该列的属性；不明示列名时，输出所登录的所有列。

COLUMN命令的详细内容如下。

- 语法

```
COL[UMN] [name [option]]
```

项目	说明
name	要指定属性的列的名称。
option	– CLE[AR]：初始化列的输出属性。 – FOR[MAT] text：指定列的format。详细内容请参考「1.7. 列的格式化」。 – HEA[DING] text：列的页眉。 – NEW_V[ALUE] variable：设置保存列值的变量。 – WRA[PPED]：列数据的长度太长时，所超过的数据分成两个行。 – TRU[NCATED]：列数据的长度太长时，剪掉所超过的数据。 – ON：打开列的输出属性。 – OFF：关闭列的输出属性。

- 示例

```
SQL> COLUMN
SQL> COLUMN empno
SQL> COLUMN empno CLEAR
SQL> COLUMN empno FORMAT 999,999
SQL> COLUMN ename FORMAT A10
SQL> COLUMN sal HEADING the salary of this month
SQL> COLUMN sal OFF
SQL> COLUMN job WRAPPED
SQL> COLUMN job TRUNCATED
```

1.6.10. CONNECT

通过其他用户连接K-DB数据库。若没有输出用户的名称或密码，则要求在kdSQL Utility输入输出提示符。

若运行CONNECT，则提交以前所运行的事务，解除以前的连接之后，尝试新的连接。即使新的连接失败也不会复原以前的连接。

CONNECT命令的详细内容如下。

- 语法

```
CONN[ECT] {username[/password[@connect_identifier]]}
```

项目	说明
username	用户名

下面是可以指定CONNECT命令的选项的有关说明。

选项	说明
password	用户的密码。
connect_identifier	是为了连接数据库的连接信息。 可以指定$KD_HOME/client/config目录的 kddsn.tbr 文件，由HOST、PORT、DB_NAME信息构成。Windows上可以在数据原本(ODBC)指定，并且首先将搜索此。

- 示例

```
SQL> CONNECT dbuser/dbuserpassword@db_id
```

1.6.11. DEFINE

定义或输出置换变量。

DEFINE命令的详细内容如下。

- 语法

```
DEF[INE] [variable]|[variable = value]
```

项目	说明
	若不指定置换变量的名称，则输出所有置换变量。
variable	要定义的置换变量的名称。
variable = value	要定义的置换变量的名称和默认值。

- 示例

```
SQL> DEFINE NAME
..... 定义名为 NAME的绑定变量。 .....
SQL> DEFINE NAME = 'SMITH'
..... 定义名为NAME的绑定变量和 SMITH的默认值。 .....
SQL> DEFINE
..... 在界面里输出所有绑定变量。 .....
```

1.6.12. DEL

删除SQL缓冲区里所设置的行。若省略行号，则删除所有行。

DEL命令的详细内容如下。

- 语法

```
DEL [number|number number|number LAST|LAST number|LAST]
```

选项	说明
number	删除所指定的号码的行。
number number	删除从第一个所指定的号码的行到第二个所指定的号码的行。
number LAST	删除从第一个所指定的号码的行到所有一个行。
LAST number	删除从最后行到所指定号码的行。
LAST	删除最后行。

- 示例

```
SQL> DEL 1
..... 删除第一个行。 .....
SQL> DEL 1 3
..... 删除第一个行到第三个行。 .....
SQL> DEL 1 LAST
..... 删除从第一个行到最后行。 .....
SQL> DEL LAST
..... 删除最后行。 .....
```

1.6.13. DESCRIBE

输出所指定的对象的列信息。这里的对象可以是表、视图、函数、过程、数据包。

- 对象为表、视图时会输出列名、数据类型、约束条件与索引信息等，这将包含根据数据类型的最大长度、精度与刻度等。
- 对象为函数、过程时会输出参数的信息（名称、数据类型、IN/OUT），为数据包时会输出所属的所有函数与过程的信息。
- 也可以输出其他用户所拥有的对象的列信息。此时，指明所有者名称，所有者名称若没被指明，则默认为表示当前用户所有的对象。

DESCRIBE命令的详细内容如下。

- 语法

```
DESC[RIBE] [schema.]{object}
```

选项	说明
schema	包含对象的Schema（或所有者）。
object	要输出的列信息的对象。

- 示例

```
SQL> DESCRIBE emp
SQL> DESC scott.emp
```

1.6.14. DISCONNECT

结束当前数据库的连接。虽然提交进行中的事务，但是不结束kdSQL Utility。

在脚本文件里运行CONNECT命令并直接结束时，会继续维持连接数据库的状态。因此，脚本文件里包含CONNECT命令，则运行DISCONNECT命令很安全。

- 语法

```
DISC[ONNECT]
```

1.6.15. EDIT

利用外部编辑器编辑特定文件或SQL缓冲区的内容。可以在环境变量$KD_EDITOR里设置要使用何种外部编辑器。

若$KD_EDITOR没有登录，则参考环境变量$EDITOR；若$EDITOR也没有登录，则使用vi编辑器编辑。此时SQL缓冲器若为空，则返回错误。

若要编辑的文件的扩展名为指定为SUFFIX系统变量的默认扩展名，则没有扩展名也可以指定文件的名称。FILEEXT系统变量的默认值为sql，并且可以利用SET命令变更。kdSQL Utility在SUFFIX系统变量在当前目录下查找所指定的文件。

EDIT命令的详细内容如下。

- 语法

```
ED[IT] [filename]
```

选项	说明
	若不指定文件名而运行EDIT命令语句，则编辑当前SQL缓冲区里所保存的内容并使用默认文件.tbedit.sql。默认文件会在kdSQL Utility结束时被自动删除。
filename	要编辑的文件名称（大多是脚本文件的名称）。

- 示例

```
SQL> EDIT run.sql
SQL> EDIT run
SQL> ED
```

1.6.16. EXECUTE

运行单一PSM语句。可以使用的PSM语句只是没有CALL语句和没有名称的数据块。用户所输出的文章的最后必须有分号（;）。

EXECUTE命令的详细内容如下。

- 语法

```
EXEC[UTE] {statement}
```

项目	说明
statement	单一PSM程序的文章。

- 示例

```
SQL> EXECUTE begin dbms_output.put_line('success'); end;
success

PSM completed

SQL> EXECUTE call proc1();
```

另外，用户所定义的绑定变量里分配值时也很适用。

```
SQL> VAR x NUMBER;
SQL> EXEC :x := 5;

PSM completed

SQL>
```

1.6.17. EXIT

结束kdSQL Utility。提交当前进行中的所有事务，结束与数据库的所有连接。

- 语法

```
EXIT
```

1.6.18. HELP

在界面里输出包含了所指定的单词的所有项目的有关帮助。

HELP命令的详细内容如下。

- 语法

```
H[ELP] [topic]
```

选项	说明
	若不指定选项，则输出kdSQL Utility里可以使用的所有命令。
topic	指定要输出帮助的单词。

- 示例

```
SQL> HELP SET
```

1.6.19. HISTORY

在界面里输出保存在历史缓冲区里的命令。

HISTORY命令的详细内容如下。

- 语法

```
HIS[TORY]
```

- 示例

```
SQL> HISTORY
..... 输出整个命令. .....
```

1.6.20. HOST

与!命令相同。

HOST命令的详细内容如下。

- 语法

```
HO[ST] [command]
```

选项	说明
	若没有命令只输入HOST，则可以到操作系统的命令提示符多次输入操作系统命令。此时，若要重新回到kdSQL Utility，则输入EXIT。
command	操作系统的命令。

1.6.21. INPUT

在SQL缓冲区里所保存的最后行的后面添加用户输入的语句。

INPUT命令的详细内容如下。

- 语法

```
I[NPUT] [statement]
```

选项	说明
	若省略statement，则间隔多个行添加语句。
statement	要添加的SQL语句。

- 示例

```
SQL> select * from dual;
..... 输出结果 .....
SQL> LIST
     select * from dual
SQL> INPUT where rownum < 2
     select * from dual
     where rownum < 2
SQL>
```

下面省略了statement选项。但是与上面不同的是在输入结束的同时，SQL语句被运行。

```
SQL> select * from dual;
..... 输出结果 .....
SQL> INPUT
     select * from dual
     ... 在此输入即可 ...
```

1.6.22. LIST

在界面里输出SQL缓冲区里的特定内容。

LIST命令的详细内容如下。

- 语法

```
L[IST] [number|number number|number LAST|LAST number|LAST]
```

选项	说明
	在省略行的号码时，输入所有行。
number	输出所指定号码的行。
number number	输出第一个指定的号码到第二个指定的号码的行。
number LAST	输出指定的号码的行到最后的行。
LAST number	输出从最后行到所指定的号码的行。
LAST	输出最后行。

- 示例

```
SQL> LIST 1
..... 输出第一个行。 .....
SQL> LIST 2 3
..... 输出第二个到第三个行。 .....
SQL> LIST 2 LAST
..... 输出第二个到最后的行。 ......
SQL> LIST LAST
..... 输出最后的行。 .....
```

1.6.23. LOADFILE

将K-DB表保存为Oracle的SQL*Loader可以识别的形式。

LOADFILE命令的详细内容如下。

- 语法

```
LOAD[FILE] {filename}
```

项目	说明
filename	除了扩展名以外的文件的名称。

- 示例

若要将名为EMP的表保存为Oracle的SQL*Loader可以识别的文件，则输入下面的内容。若运行本命令，则会生成emp.ctl与emp.dat两个文件。

```
SQL> LOADFILE emp
SQL> select * from emp;
```

1.6.24. LOOP

无限重复执行任意语句的指令。

LOOP指令的详细内容如下。

- 语法

```
LOOP stmt
```

项目	说明
stmt	反复执行的语句。

- 示例

```
SQL> LOOP select count(*) from v$session
..... 将SQL以一秒间隔反复执行。 .....
SQL> SET INTERVAL 10
SQL> LOOP ls
..... 将LS以10秒间隔反复执行。 .....
```

1.6.25. LS

输出当前用户生成的特定类型或名称的数据库对象的信息。

LS命令的详细内容如下。

- 语法

```
LS [object_type|object_name]
```

项目	说明
	若省略object_type或object_name，则输出用户所拥有的所有对象。
object_type	FUNCTION、INDEX、PACKAGE、PROCEDURE、SEQUENCE、SYNONYM、TABLE、TABLESPACE、TRIGGER、TYPE、USER、VIEW。
object_name	是要输出的对象的名称。 可以使用显示任意模式的型号（*）字符。

- 示例

```
SQL> LS
NAME                           SUBNAME       OBJECT_TYPE
------------------------------ ------------- --------------------
SYS_CON100                                   INDEX
SYS_CON400                                   INDEX
SYS_CON700                                   INDEX
_DD_CCOL_IDX1                                INDEX
......中间省略.....
UTL_RAW                                      PACKAGE
DBMS_STATS                                   PACKGE BODY
KD_HIDDEN2                                   PACKGE BODY

SQL>
..... 输出所有对象。 .....

SQL> LS TABLESPACE
TABLESPACE_NAME
------------------------------
SYSTEM
UNDO
TEMP
USR
..... 输出类型为TABLESPACE的所有对象。.....

SQL> LS USER
USERNAME
------------------------------
SYS
..... 查询正在连接当前系统的用户。 .....
```

1.6.26. PAUSE

暂时停止运行，直到用户点击<Enter>。点击消息时，相关消息显示在界面里。

PAUSE命令的详细内容如下。

- 语法

```
PAU[SE] [message]
```

选项	说明
message	用户点击<Enter>时界面显示的消息。

- 示例

```
SQL> PAUSE please enter...
please enter...
..... 点击<Enter> .....
SQL>
```

1.6.27. PING

输出对特定数据库是否可以访问的状态。

PING命令的详细内容如下。

- 语法

```
PING connect_identifier
```

项目	说明
connect_identifier	要访问的数据库名。

- 示例

```
SQL> PING kdb11
Server is alive.

SQL>
```

1.6.28. PRINT

输出用户定义的绑定变量的名称和值。

PRINT命令的详细内容如下。

- 语法

```
PRI[NT] [variable...]
```

选项	说明
	省略variable时，输出所有绑定变量。
variable	要输出的绑定变量的名称。

- 示例

```
SQL> VARIABLE x NUMBER
SQL> EXECUTE :x := 5;
SQL> PRINT x

         x
----------
         5

SQL>
```

1.6.29. PROMPT

在界面里输出特定消息或空的行。

PROMPT命令的详细内容如下。

- 语法

```
PRO[MPT] [message]
```

项目	说明
	要省略message时输出空的行。
message	是界面里要显示的消息。

- 示例

下面是外部编辑的SQL文件的示例，名为PromptUsage.sql。

```
PROMPT >>> Test is started.
CREATE TABLE T (c1 NUMBER);
INSERT INTO T VALUES (1);
PROMPT Value 1 is inserted.
COMMIT;
PROMPT <<< Test is ended.
```

下面显示了运行上面的PromptUsage.sql的结果。

```
SQL> @PromptUsage
>>> Test is started.
Table 'T' created.
1 row inserted.
Value 1 is inserted.
Commit succeeded.
<<< Test is ended.
File finished.

SQL>
```

1.6.30. QUIT

与EXIT命令相同。

QUIT命令的详细内容如下。

- 语法

```
Q[UIT]
```

1.6.31. RUN

虽然与/命令相同，使用该命令时，在界面里输出当前运行的SQL语法。

RUN命令的详细内容如下。

- 语法

```
R[UN]
```

- 示例

```
SQL> select 1 from dual;

             1
----------
             1

1 row selected.

SQL> RUN
      1 select 1 from dual

             1
----------
             1

1 row selected.
```

1.6.32. SAVE CREDENTIAL

在kdSQL加密数据库访问信息来保存在文件上的命令。具体使用在「1.5.3. 访问信息加密功能」当中有具体说明。

SAVE CREDENTIAL命令的详细内容如下。

- 语法

```
SAVE CREDENTIAL [filename]
```

项目	说明
	只要输入SAVE CREDENTIAL命令，就可以在环境变量ISQL_WALLET_PATH上设置的路径文件上加密当前数据库访问信息来保存。
filename	在所输入的filename文件上加密数据库访问信息来保存。

- 示例

```
SQL> SAVE CREDENTIAL
SQL> SAVE CREDENTIAL "./wallet.dat"
```

1.6.33. SET

设置kdSQL Utility系统变量。由SET命令所设置的系统变量使用SHOW命令输出。但是，所变更的系统变量只在当前会话里有效。

各系统变量的相关内容在「1.3. 系统变量」具体说明。

SET命令的详细内容如下。

- 语法

```
SET {parameter} {value}
```

项目	说明
parameter	kdSQL Utility系统变量的名称。
value	kdSQL Utility系统变量的值。

- 示例

```
SQL> SET AUTOCOMMIT ON
```

1.6.34. SHOW

输出kdSQL Utility的系统变量。可以使用参数选择要输出的信息，可以输出所有信息。

SHOW命令的详细内容如下。

- 语法

```
SHO[W] {option}
```

下面是在option可以输入的项目的有关说明。

项目	说明
system_parameter	输出所给名称的kdSQL Utility的系统变量。
ALL	输出所有kdSQL Utility系统变量。
ERROR	输出前面发生的PSM程序的错误。
PARAM[ETERS] [name]	输出所给名称的数据库系统变量。 但是名称被省略时输出所有系统变量。
RELEASE	输出kdSQL Utility的Release信息。

- 示例

```
SQL> SHOW autocommit
SQL> SHOW all
SQL> SHOW error
```

```
SQL> SHOW param db_name
SQL> SHOW release
```

1.6.35. SPOOL

开始或结束界面里所输出的内容保存在所有外部文件里的过程。输出文件会在当前目录里生成。

SPOOL命令的详细内容如下。

- 语法

```
SPO[OL] [filename [APP[END]]|OFF]
```

项目	说明
	若只输入SPOOL命令，则输出SPOOL命令的当前运行状态。
filename	要保存的输出文件的名称。
APP[END]	选择是否粘贴在文件尾端。
OFF	停止输出文件的保存。

- 示例

```
SQL> SPOOL report.txt
SQL> SPOOL OFF
```

1.6.36. START

与@命令相同。

START命令的详细内容如下。

- 语法

```
STA[RT] {filename}
```

项目	说明
filename	脚本文件的名称。

1.6.37. KDDOWN

结束K-DB。可以按紧急性选择结束选项，并且按选项重启数据库时，会需要复原过程。

若要运行此命令，需要通过SYSDBA或SYSOPER连接数据库。

KDDOWN命令的详细内容如下。

- 语法

```
KDDOWN [NORMAL|POST_TX|IMMEDIATE|ABORT]
```

项目	说明
NORMAL	等待，直到当前连接中的所有用户结束连接之后结束。（默认值）
POST_TX	等待，直到结束当前进行中的事务后结束。
IMMEDIATE	回滚后强制结束当前进行中的事务。
ABORT	不进行回滚当前进行中的事务，立即结束。

- 示例

```
SQL> KDDOWN
SQL> KDDOWN ABORT
```

1.6.38. UNDEFINE

删除由ACCEPT命令等定义的绑定变量。

UNDEFINE命令的详细内容如下。

- 语法

```
UNDEF[INE] [variable...]
```

项目	说明
	要省略variable...时，删除所有绑定变量。
variable...	绑定变量名称的目录。

- 示例

```
SQL> UNDEFINE x
SQL> UNDEFINE x y z
```

1.6.39. VARIABLE

说明PSM程序或SQL语句里可以使用的用户所定义的绑定变量。

VARIABLE命令的详细内容如下。

- 语法

```
VAR[IABLE] [variable [datatype]]
```

项目	说明
	只使用VARIABLE命令时，在界面里输出所有绑定变量。
variable	置换变量的名称。
datatype	数据类型。 目前支持如下类型。 – NUMBER, CHAR(n), VARCHAR(n), VARCHAR2(n), NCHAR(n), NVARCHAR2(n), RAW(n), BLOB, CLOB, NCLOB, DATE, TIMESTAMP, REFCURSOR

- 示例

```
SQL> VARIABLE x NUMBER
SQL> EXEC :x := 1;

PSM completed.

SQL> SELECT :x FROM DUAL;
        :x
----------
         1
1 row selected.
SQL>
```

1.6.40. WHENEVER

发生错误时定义kdSQL的操作。

WHENEVER命令的详细内容如下。

- 语法

```
WHENEVER {error_type} {EXIT [exit_option]|CONTINUE} [tx_option]
```

下面是可以输入到error_type上的项目说明。

项目	说明
	决定对何种类型的错误运行定义的选项。在未使用WHENEVER命令的基本状态下与OSERROR和SQLERROR全部给予CONTINUE NONE选项的命令执行相同。
OSERROR	kdSQL为运行中环境的操作系统错误。
SQLERROR	执行SQL语句当中发生的错误。

下面是输入在exit_option上的项目说明。

项目	说明
	给予EXIT关键字时若发生错误，则会终止程序，在这时指定终止代码值（整数）的选项。
SUCCESS	返回正常终止代码0。
FAILURE	返回失败终止代码1。
WARNING	返回警告终止代码2。
SQL.SQLCODE	返回发生错误时的错误代码。但可以使用终止代码的范围将根据值有所改变。
n	可以指定使用终止代码的整数型数字，使用值的范围根据操作系统各异。
:variable	用VARIABLE 命令定义的绑定变量来指定终止代码。但使用的绑定变量得是数字型类型。

下面是可以输入在tx_option上的项目说明。

项目	说明
	定义操作当中的事务处理。
COMMIT	发生错误时执行COMMIT。
ROLLBACK	发生错误时执行ROLLBACK。
NONE	发生错误时也对事务不做任何处理的选项。只有CONTINUE时才可使用。

- 示例

```
SQL> whenever sqlerror exit failure rollback
SQL> select 1 from no_such_table;
TBR-8033: Specified schema object was not found.
at line 1, column 16:
select 1 from no_such_table
             ^

$ echo exit code: $?
exit code: 1
$
```

1.7. 列的格式化

本节将说明随数据类型设置kdSQL Utility的列格式化的方法。

kdSQL Utility的列格式化通过前一节说明的COLUMN命令设置，利用COLUMN命令输出。

1.7.1. 字符型

为CHAR、NCHAR、VARCHAR、NVARCHAR类型时，数据库列的长度为默认长度。数据值比列的长度大时，数据会在下一个行里被记录或删除，使用字符型格式化时，便可以简单处理。

设置字符型的列格式化的详细内容如下。

- 语法

```
COL[UMN] {name} FOR[MAT] A{n}
```

A也可以使用为小写字母a，n指字符串数据的长度。

- 示例

```
SQL> SELECT 'K-DB is the best choice' test FROM DUAL;

TEST
-------------------------
K-DB is the best choice

1 row selected.

SQL> COL test FORMAT a10
SQL> SELECT 'K-DB is the best choice' test FROM DUAL;

TEST
----------
K-DB is
the best c
hoice

1 row selected.

SQL>
```

1.7.2. 数字型

设置数字型的列格式化的详细内容如下。

- 语法

```
COL[UMN] {col_name} FOR[MAT] {fmt_str}
```

项目	说明
col_name	指定列名。
fmt_str	在下一个表里指定说明的列格式化。

下面是可以在fmt_str指定的格式化。

格式	设置示例	说明
逗号（,）	9,999	在所给的位置里输出逗号（,）。
点（.）	9.999	在分离整数部分与小数部分里输出点（.）。
$	$9999	在最前面输出$。
0	0999, 9990	在最前面或最后面输出0。
9	9999	输出所给位数的数字。
B	B9999	整数部分为0时，用空白置换。
C	C9999	在所给的位置里输出ISO currency symbol。
D	9D999	为了分离实数的整数和小数，输出decimal字符。
EEEE	9.99EEEE	通过科学的计数法输出。
G	9G999	在整数部分所给位置里输出组分离符。
L	L9999	在所给位置里输出local currency symbol。
MI	9999MI	在负数后面输出负号，在整数后面输出空白。
PR	9999PR	负数时，用<和>包括起来输出；正数时，在两面输出空白。
RN	RN	以大写字母输出。
rn	rn	以小写字母输出。
S	S9999, 9999S	在最前面或最后面输出正数或负数符号。
TM	TM	尽可能输出小的数。
U	U9999	在所给位置里输出dual currency symbol。
V	99V999	输出10n倍大的值。 这里的n是V后面出现的9的个数。
X	XXXX, xxxx	以16进数的形式输出。

- 示例

```
SQL> COLUMN x FORMAT 999,999
SQL> SELECT 123456 x FROM DUAL;

           X
 --------
   123,456

1 row selected.
```

第2章 kdMigrator 2.0

本章将主要介绍kdMigrator 2.0 Utility并说明其使用方法。

2.1. 概要

kdMigrator 2.0是K-DB DB中提供的Migration Utility。

该Utility帮助Oracle DBMS组成的整个或部分数据库移动到K-DB迁移操作。也就是说将保存在Source DBMS的表、索引、视图等Schema Object与PSM程序等移动到K-DB数据库，使数据库实现与之前数据库相同的功能。

kdMigrator 2.0 Utility的功能如下。

- 选择用户所需表来迁移到K-DB。
- 迁移表、索引、视图、同义词等Schema Object和在表中定义的各种约束条件。
- 迁移用户特权（privilege）及角色（role）。
- 提供迁移目标数据库的相关信息。
- 使用**[Option]**按钮以多种方法迁移。
- 通过进程界面，可确认迁移的进行事项。

参考

kdMigrator 2.0由Java语言实现，可以在Java 6以上版本使用。并且要将运行之前需要访问的DB的JDBC Driver文件路径添加在运行脚本内的classpath设置。

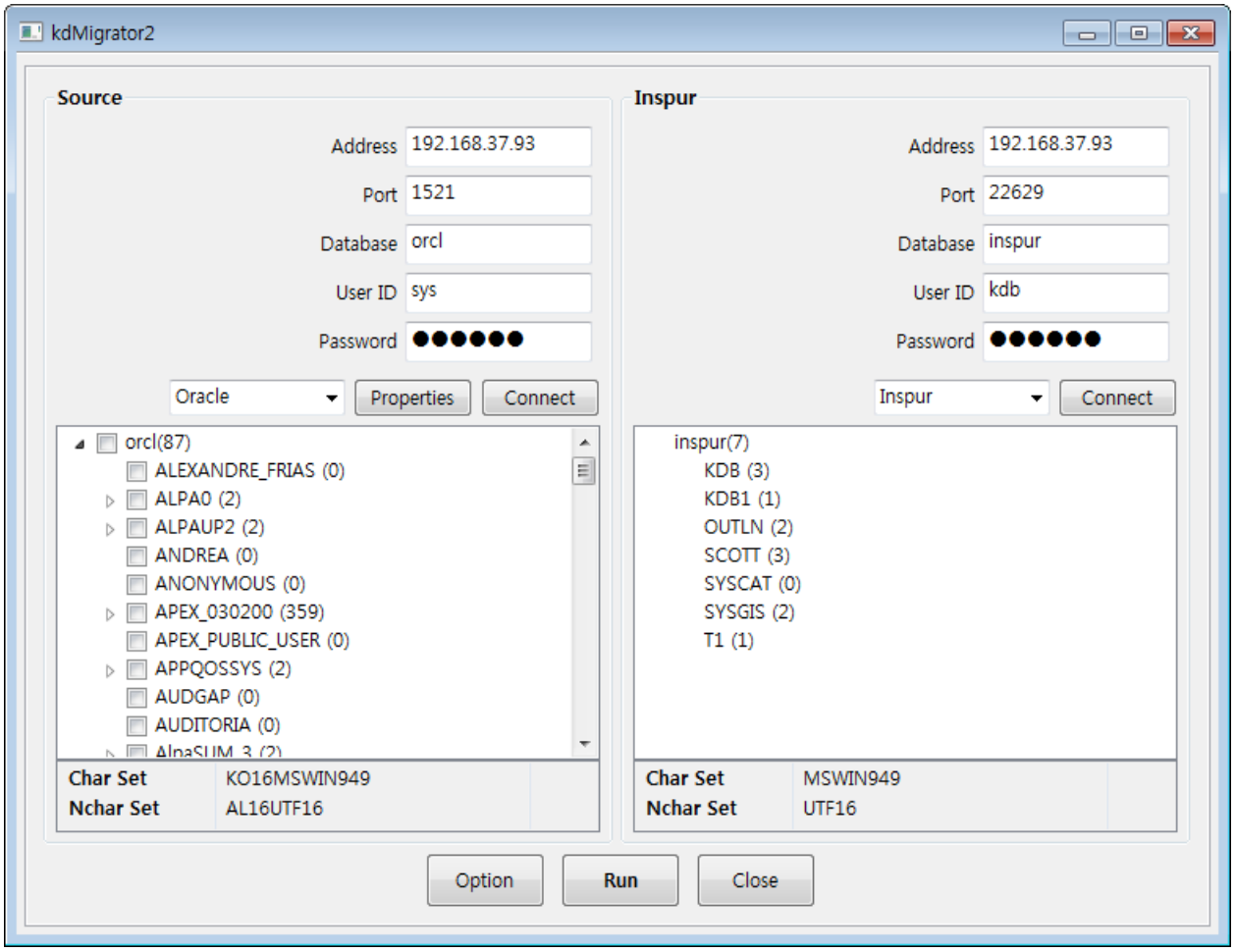
2.2. 界面说明

kdMigrator 2.0由Main界面、Option界面、Progress界面Report界面构成。

2.2.1. Main界面

下面是Main界面的说明。

[图 **2.1] Main**界面



- **Source**

  – **Source**连接信息

  下面是Source连接信息各项目的相关说明。

项目	说明
Address	Source数据库的IP地址名。
Port	Source数据库的端口号。
Database	Source数据库的SID。
User ID	Source数据库的用户ID。
Password	访问Source数据库的用户密码。
DB Type	Source数据库的种类。各数据库需要考虑的事项请参阅「2.3.1. 所支持的Objects」。
Properties	Source数据库的附加连接信息。

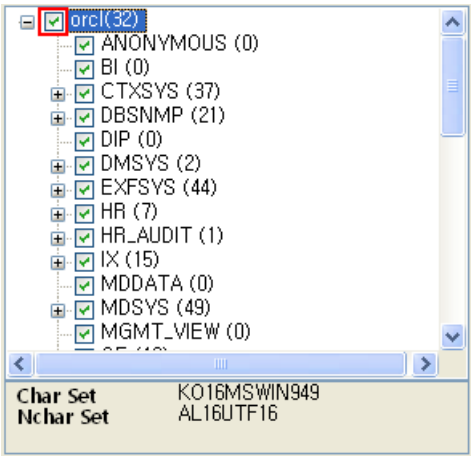
  – **Source**数据库视图

Source数据库视图显示用户所需数据的选择功能和数据库的字符集设置。数据选择方式有三种。

- Full Mode

  选择数据库名时，所有Schema都将被选择。哪怕只释放所属Schema的一个要素，也会从整体模式转换为Schema模式。

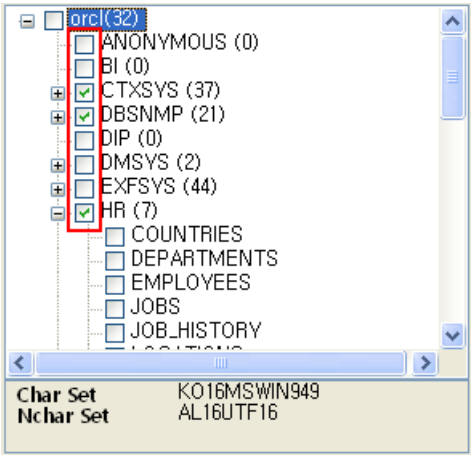
  [图 2.2] Full Mode选择方式



- Schema Mode

  选择特定Schema名时，将指定所属Schema的所有表。哪怕只释放所属表的一个要素，也会从Schema模式转换为表模式。

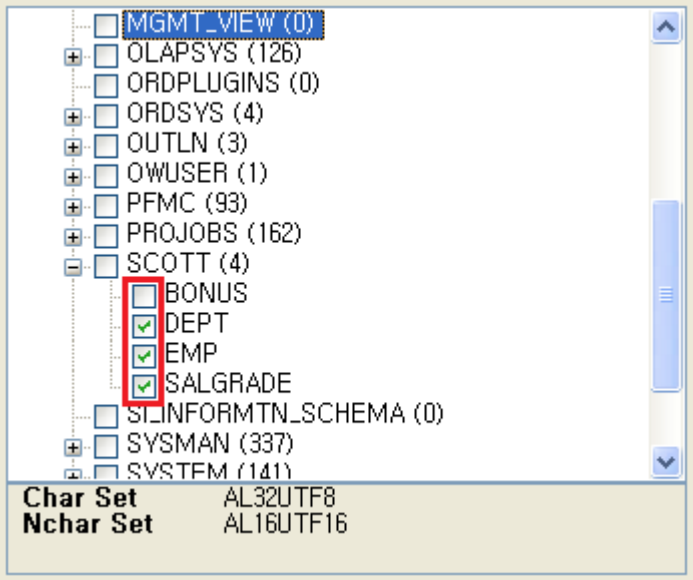
  [图 2.3] Schema Mode选择方式



- Table Mode

  选择表要素为kdMigrator的最小迁移单位。

[图 2.4] Table Mode选择方式



Source的字符集设置显示如下两种信息。

- Char Set
- NChar Set

- **K-DB**

  – **K-DB**连接信息

  下面是K-DB访问信息各项说明。

项目	说明
Address	K-DB数据库的IP地址名。
Port	K-DB数据库的端口号。
Database	K-DB数据库的SID。
User ID	K-DB数据库的用户ID。
Password	访问K-DB数据库的用户密码。
DB Version	K-DB数据库的版本。

  – **K-DB** 数据库视图

  显示K-DB数据库视图显示在K-DB上查询数据的功能和字符集设置。

- 按钮

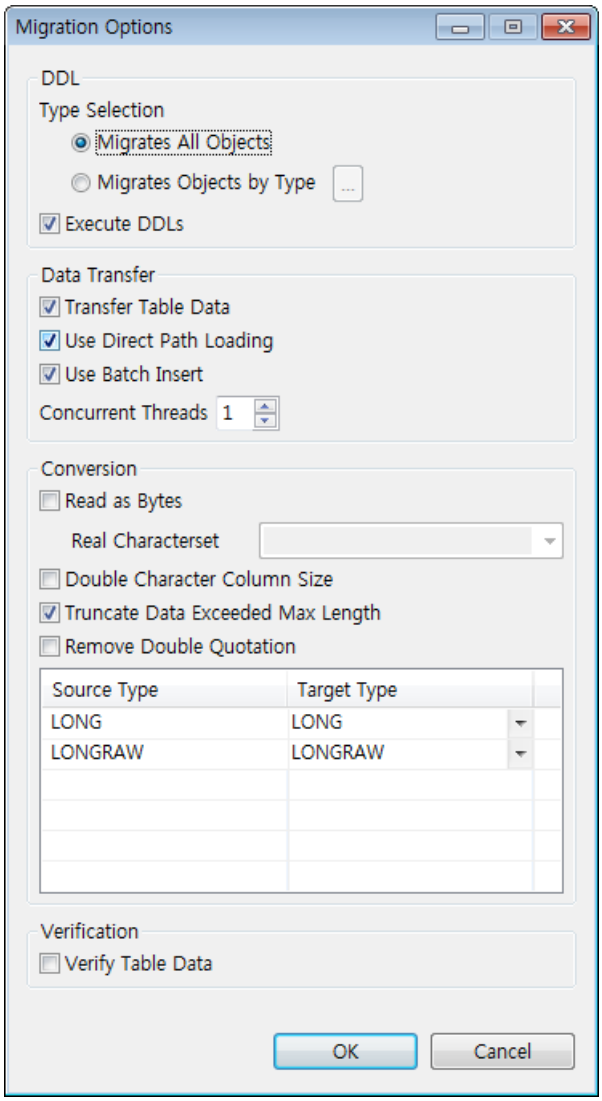
  迁移主窗口的按钮如下。

按钮	说明
[Connect]	连接目标数据库。
[Option]	出现选择对话框。
[Run]	开始迁移。
[Close]	结束kdMigrator。

2.2.2. Option界面

下面是Option界面的相关说明。

[图 2.5] Option界面



- **DDL**

  DDL是在迁移时作为第一步创建K-DB数据库对象时使用的语法。在这里决定提取任意类型的Schema Object的DDL选择是否执行所生成的DDL语句等有关DDL的选项。

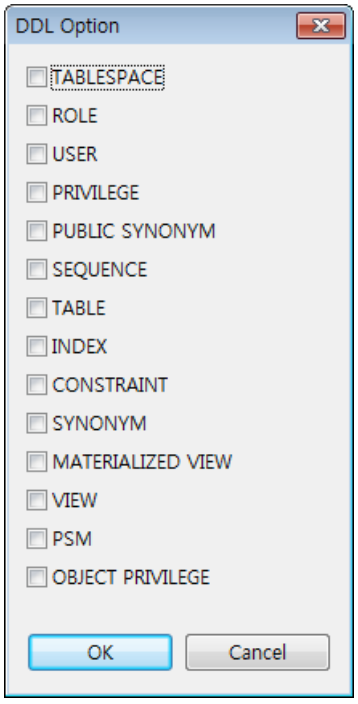
  – Type Selection

  指定成为迁移对象的Schema Object类型。

项目	说明
Migrates All Objects	提取所支持的所有类型的DDL语句。
Migrates Objects by Type	提取根据所选的Object种类所属的DDL语句。

点击详细选择按钮(**[...]**)时会如下出现Object种类选项框。

[图 **2.6] Object**种类选项框



– Execute DDLs

决定是否在目标数据库运行提取的DDL语句。若迁移的表等对象已生成时，可以跳到释放选择来跳到运行生成SQL语句的步骤。

- **Data Transfer**

数据传送为DDL下一步骤，将数据从Source数据库迁移到K-DB。

分类	说明
Transfer Table Data	选择是否传送表数据。
Use Direct Path Load	将表数据以Direct Path Load方式迁移。
Use Batch Insert	将表数据以Batch Insert方式迁移。
Concurrent Thread	指定为了同时迁移多个表数据而使用的线程个数。

- **Conversion**

下面是数据转换选项的相关说明。

分类	说明
Read as Bytes	在保存Oracle的char、varchar这种字符串的列上，使用与数据库不同设置的其他字符集来保存实际字符串时，若以字符串格式取回数据，字符串会产生乱码。为防止这一点，不用字符串格式而是用二进制格式取回数据，并且用二进制格式迁移到K-DB时使用的选项。
Real Characterset	表数据以外的部分若有与实际数据库的字符集输入不同的部分时，迁移后相应内容可能会乱码。为了防止此要指定实际使用的字符集来移动到正确的字符串形态的选项。 激活Read as Bytes设置时才有效，接收影响的项目为PSM DDL、表的comment、表的列comment。
Double Character Column Size	Source数据库和K-DB的字符集相互不同时，转换的字符串数据的实际byte长度会有所不同。 因为此超出列的长度时会发生限制迁移失败的情况。为防止此，生成基于字符串的列时，改为Source数据库指定的两倍的长度。
Truncate Data Exceeded Max Length	列长度超过K-DB当中支持的最大长度时，决定切断后迁移或是发起错误的选项。
Remove Double Quotation	迁移时为忽略Object名大小写而提供的选项。 未使用选项时，迁移时为了将source的Object名直接迁移默认添加双引号来迁移。 支持Table, Column object，不应用在如PSM Source以文本形态保存的部分。
Type Conversion Table	Source数据库和K-DB的列类型不完全一致，因此这是为兼容而设置的U型安详。该选项内容根据Source数据库种类各异。

- **Verification**

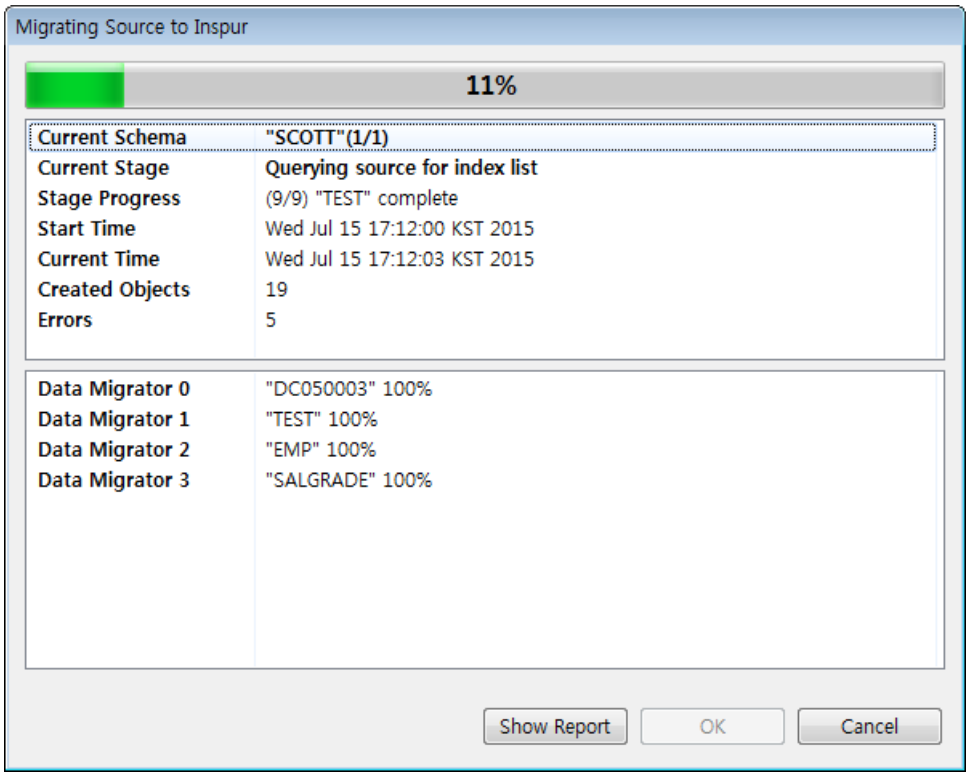
下面是数据验证选项的说明。

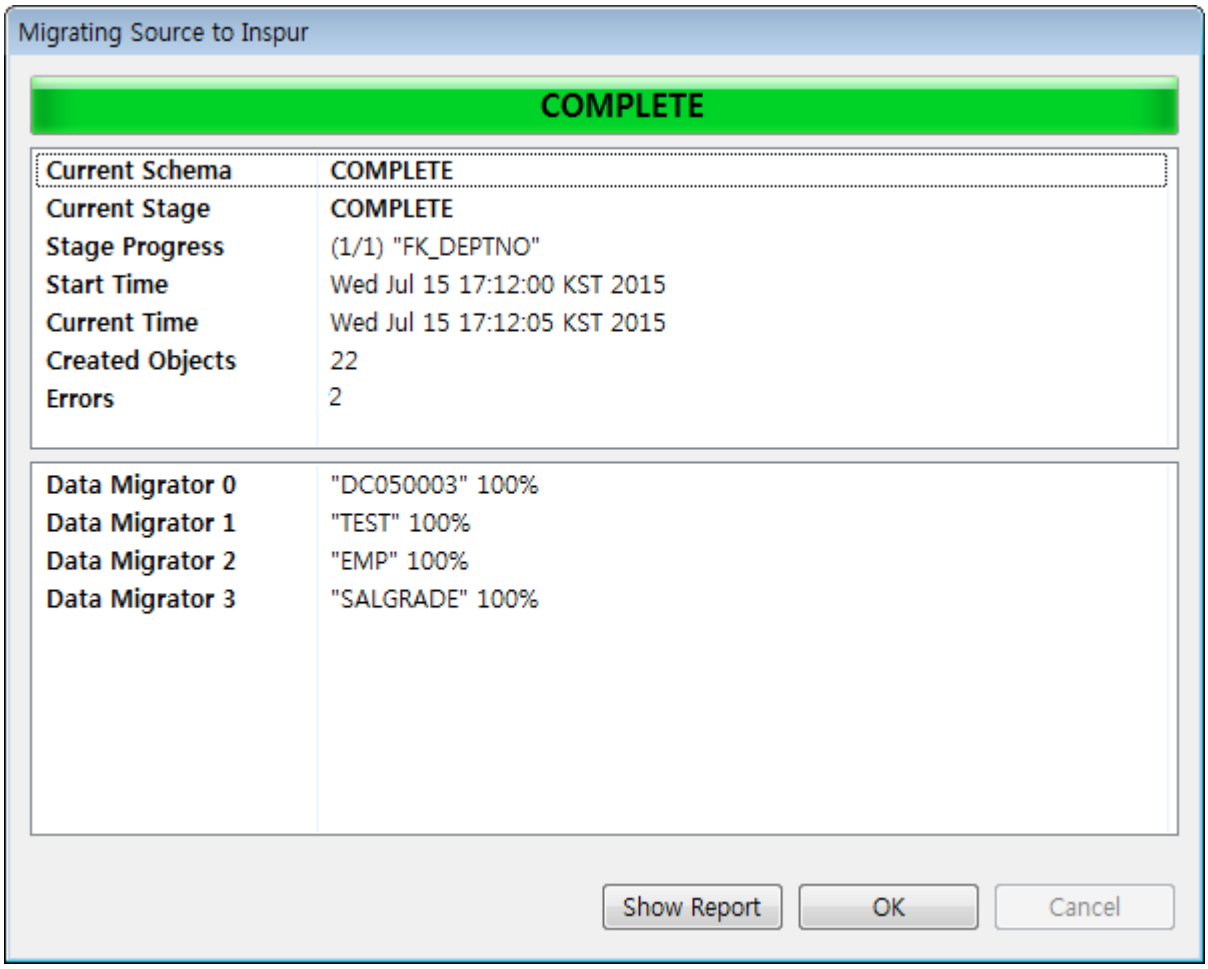
分类	说明
Verify Table Data	确认表数据是否正确迁移的功能。 读取Source数据库和K-DB两边迁移的所有表数据1:1比较。数据量较多时消耗时间较大。

2.2.3. Progress界面

用户可通过Progress窗口确认迁移的进行事项。

[图 2.7] Progress界面





- 查询项目

项目	说明
Current Schema	当前正进行的架构（Schema ）信息。 当前架构显示需要迁移的架构个数和已迁移的架构个数。迁移结束后将显示COMPLETE。
Current Progress	当前正进行的架构阶段信息。 阶段信息显示架构数据类型信息，迁移结束后将显示COMPLETE。
Strage Progress	显示进行迁移的各阶段进行状态。 阶段进行信息是指正进行迁移的数据名，并且显示需要迁移的数据个数与已迁移的数据个数。
Created Objects	显示当前成功生成的Object个数。
Errors	显示当前发生的Error个数。

项目	说明
Data Migrator #	显示处理表数据的线程，显示当前个处理的表名称和进行率。总个数将根据Option界面的数据传送选项中Concurrent Threads项目中指定的值不同。

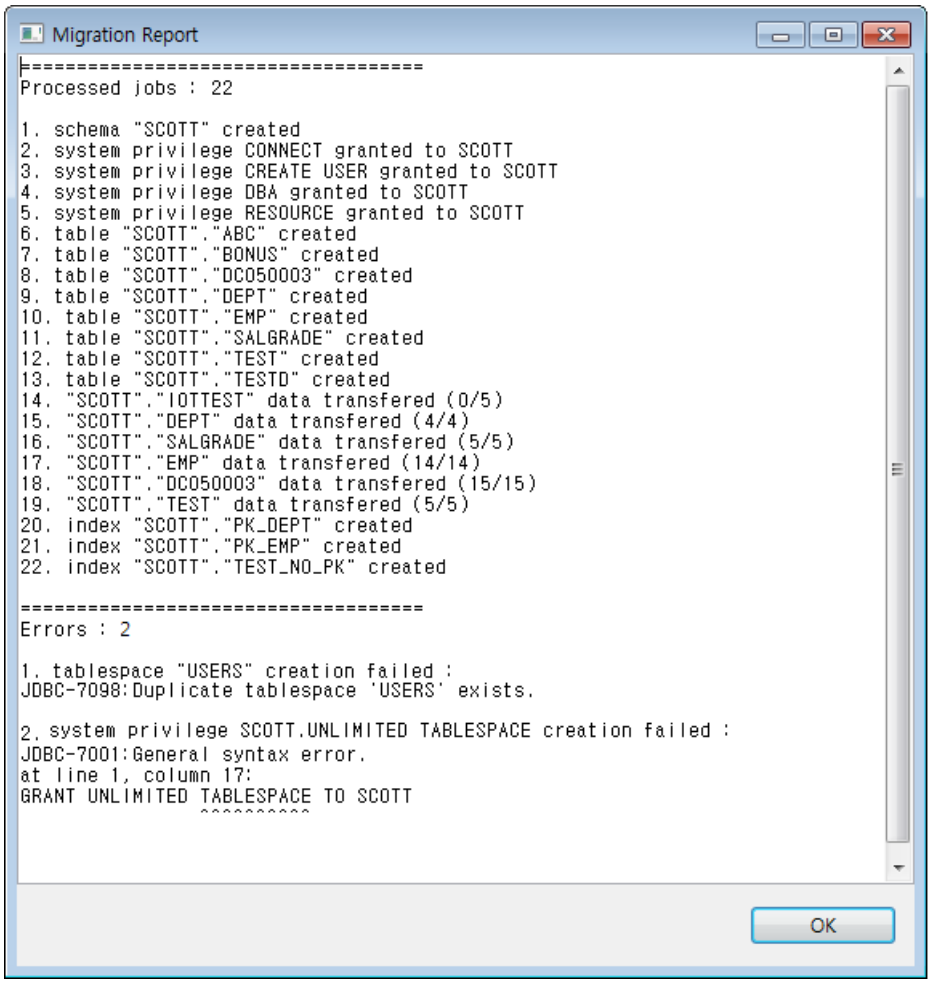
- 按钮

项目	说明
[Show Report]	可以确认迁移结果的Report界面。详细内容请参阅「2.2.4. Report 界面」。
[OK]	进行迁移当中被禁用。操作过程全部结束时按钮将被激活，点击时全部过程结束。
[Cancel]	迁移进行中断。操作过程全部结束时禁用该按钮。

2.2.4. Report 界面

Report界面显示迁移的进行结果。

[图 2.8] Report界面



2.3. 迁移目标

kdMigrator 2.0 Utility支持Full Mode、Schema Mode、Table Mode三种。

各模式支持不同的迁移范围。

- Full Mode

  选择Full Mode时，数据库内的所有对象将成为迁移目标。

- Schema Mode

  选择特定Schema时，用Schema Mode运行所选Schema和属于此的对象或相关对象将成为迁移目标。

- Table Mode

  不仅是选择的表，所属相应表的Schema关联对象也将被迁移。

根据各模式迁移的对象如下表。

项目	**Full Mode**	**Schema Mode**	**Table Mode**
TABLESPACE	●	●	✕
ROLE	●	✕	✕
SCHEMA	●	●	●
SYSTEM PRIVILEGE	●	✕	✕
PUBLIC SYNONYM	●	●	●
SEQUENCE	●	●	●
TABLE	●	●	●
INDEX	●	●	●
CONSTRAINT	●	●	●
SYNONYM	●	●	●
MATERIALIZED VIEW	●	●	●
VIEW	●	●	●
REFERENTIAL CONSTRAINT	●	●	●
PSM	●	●	●
OBJECT PRIVILEGE	●	●	●

这时目标数据库上新生成的用户密码全部被初始化，默认值为'kdb'。

Index Organized Table(IOT)时，Source数据库为Oracle时支持迁移。

下面是简单示例。

```
# DBA权限用户的登录
create user owuser identified by kdb;
grant resource, connect to owuser;
create user gtuser1 identified by kdb;
grant resource, connect to gtuser1;
create user gtuser2 identified by kdb;

# owuser用户的登录
create table grantest1 ( c1 varchar2(20) );
grant select on grantest1 to gtuser1  with grant option;

# gtuser1用户的登录
grant select on owuser.grantest1 to gtuser2;
```

如上顺序赋予特权时，将生成grantor不同的object privilege。这种属于grantor的用户访问信息不可能在Migrator全部确认，因此将一次性执行迁移，grantor不能直接移动。若要生成用户A为grantor，用户B为grantee的特权时需要参考上例，首先向A赋予特权后用A来登录重新向B赋予权限。

2.3.1. 所支持的Objects

Main界面的Source访问信息上可以选择Source数据库。要按照各数据库来考虑的项目如下。

- default value的表达式在K-DB当中不支持时，迁移将失败。
- character的size和number的precision等超出K-DB的支持范围时迁移将失败。

Oracle

下面是Source数据库为Oracle时的考虑事项。

- **Main**界面**Source**访问信息

  要指定Connect As设置。点击**[Properties]**按钮时出现可选择选项的对话框。只能在NORMAL、SYSDBA、SYSOPER中选择一项。（默认值：NORMAL）

参考

选择SYSDBA或SYSOPER来访问时，根据Oracle的设置会进制远程上的登录。此情况可以通过设置REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE来解决，相应设置的详细内容请参阅Oracle的文档。

- **Option**界面的数据转换选项

  出现Option界面时可以利用Type Conversion Table来设置列类型转换类型。

LONG和LONG RAW列在Oracle 8x以后不推荐使用，作为推荐使用的列类型只是为于7x之前版本的兼容性而支持。利用该选项来迁移时可以指定迁移时是按照上述列来转换为各自对应的CLOB和BLOB，或是持续相应类型。

下面是从Oracle迁移到K-DB时根据各支持object来按类型整理的内容。

Oracle	**K-DB**	迁移	备注
Constraint	Constraint	●	对Primary Key, Foreign Key, Check, Ref Constraint支持迁移。Primary Key index/constraint全部被处理为constraint。Check constraint的表达式是利用保存在Oracle的DD上的语句来生成DDL。
Index	Index	●	支持：Bitmap, Function-based Index 未支持：R-TREE, Domain Index
Materialized View	Materialized View	●	
Materialized View Log	Materialized View Log	●	
Privilege	Privilege	●	
PSM	PSM	●	
Role	Role	●	
Schema	Schema	●	
Sequence	Sequence	●	
Synonym	Synonym	●	
Table	Table	●	支持IOT和Partitioned Table
Tablespace	Tablespace	●	
View	View	●	

从Oracle迁移到K-DB时输入类型如下转换。

Oracle	迁移**Data Type**	备注
blob	BLOB	
binary_float	BINARY_FLOAT	
binary_double	BINARY_DOUBLE	
character	CHAR	
clob	CLOB	
date	DATE	
interval day to second	INTERVAL DAY(2) TO SECOND(6)	

Oracle	迁移Data Type	备注
interval year to month	INTERVAL YEAR(2) TO MONTH	
long	LONG	
long raw	LONG RAW	
nchar	NCHAR	
nclob	NCLOB	
number	NUMBER	
nvarchar2	NVARCHAR2	
rowid	ROWID	
time	TIME	
timestamp	TIMESTAMP	
timestamp with time zone	TIMESTAMP WITH TIME ZONE	
timestamp with local time zone	TIMESTAMP(6) WITH LOCAL TIME ZONE	
varchar	VARCHAR	
varchar2	VARCHAR2	
xmltype	XMLTYPE	

参考

对BFILE和spatial data不支持迁移。

K-DB

在相同的K-DB间执行迁移时，与选择其他数据库不同的是Source数据库和目标数据库K-DB连接时会使用相同的JDBC驱动器。因此包含在kdMigrator的JDBC要与两边数据库全部兼容，最好使用最新的JDBC。

Sybase Adaptive Server Enterprise

下面是以Sybase Adaptive Server Enterprice (ASE) 15为标准整理的与K-DB不同部分的内容。

Sybase ASE	K-DB	迁移	备注
User	Schema	●	K-DB的schema为包含DB schema和DB user的概念。
Segment	Segment	●	ASE中没有Tablespace概念，各对象将直接保存在Segment。迁移时创建属于Segment名的Tablespace来将各对象分配给此。
✕	Tablespace		

Sybase ASE	**K-DB**	迁移	备注
Role	Role	●	
Table	Table	●	ASE的表中迁移分类于USER TABLE的表。
View	View	●（部分）	可以利用ASE中提供的sp_helptext来获取的生成DDL来迁移。但语法不能完全兼容。
Index	Index	●（部分）	迁移除了Function based Index的Table Index。
Rule	Constraint	●	可以迁移Primary Key、Unique、Not Null、Check、Referential constraint。
System Protect	System Privilege	●（部分）	System Protect和Privilege的各项名只有在ASE和K-DB两处全部相同时才可迁移。
	Object Privilege		
Transaction SQL	PSM	●（部分）	利用ASE中提供的sp_helptext来获取的生成DDL来迁移。但语法不能完全兼容。
SQLJ Procedure			
Scalar Function			

Informix

下面是Source数据库为Informix时的考虑事项。

- **Main**界面**Source**访问信息的**Properties**

  要输入Informix服务器名。点击**[Properties]**按钮时会出现输入Informix服务器名的对话框。

SQL Server

下面是将PostgreSQL迁移到K-DB是的Object的各类型内容。各Object的Storage选项等具体选项大部分不支持。

SQL Server	**K-DB**	迁移	备注
Schema	Schema	●	
Table	Table	●	
View	View	✕	
Index	Index	●	Unique index/constraint全部处理为index。
Privilege	Privilege	✕	
Role	Role	✕	
Sequence	Sequence	✕	
Tablespace	Tablespace	✕	

SQL Server	**K-DB**	迁移	备注
Index	Index	●	
Constraint	Constraint	●	对Primary Key、Foreign Key、Check支持迁移。Primary Key index/constraint全部处理为constraint。 不支持Foreign Key的update规则。
Synonym	Synonym	✕	
✕	Public Synonym	✕	
Materialized View	Materialized View	✕	
PSM	PSM	✕	

下面是将PostgreSQL迁移到K-DB是的数据各类型。

SQL Server	迁移**Data Type**	备注
bigint	NUMBER(19)	
binary	RAW	长度超过2000的数据将被truncate。
bit	NUMBER(1)	
char	CHAR	最大列长为2000(MAX_CHAR_SIZE)，超过此长度的数据将被trancate。
cursor	未支持	
date	DATE	
datetime	TIMESTAMP(3)	
datetime2	TIMESTAMP(7)	
datetimeoffset	TIMESTAMP WITH TIMEZONE	
decimal	NUMBER	
double precision	BINARY_DOUBLE	
float	BINARY_FLOAT 或 BINARY_DOUBLE	1 ~ 24 precision迁移到BINARY_FLOAT，25 ~ 53被迁移到BINARY_DOUBLE。
hyierachyid	未支持	
image	BLOB	
int	NUMBER(10)	
money	NUMBER(19, 4)	
nchar	NCHAR	长度超过2000的数据将被truncate。
numeric	NUMBER	

SQL Server	迁移**Data Type**	备注
nvarchar	NVARCHAR	长度超过4000的数据将被truncate。
ntext	NCLOB	
real	BINARY_FLOAT	
smalldatetime	DATE	
smallint	NUMBER(5)	
smallmoney	NUMBER(10, 4)	
sql_variant	LONG RAW	
table	未支持	
text	CLOB	
time	TIME(7)	
timestamp	RAW(8)	
tinyint	NUMBER(3)	
uniqueidentifier	VARCHAR(36)	
varbinary	RAW	长度超过2000的数据将被truncate，RAW(max)将被迁移到BLOB。
varchar	VARCHAR	长度超过8000的数据将被truncate。 根据Target DB版本所支持的varchar max size有所差异。
xml	XMLTYPE	

参考

不支持spartial类型列。

PostgreSQL

下面是将PostgreSQL迁移到K-DB时，将object根据类型整理的内容。不支持各object的storage选项等具体选项。

PostgreSQL	**K-DB**	迁移	备注
Tablespace	Tablespace	✕	
Role	Role	✕	
Schema	Schema(=User)	●	K-DB中Schema和User是相同的Object。但是PostgreSQL中Schema(namespace)和User是区分的。
User			

PostgreSQL	**K-DB**	迁移	备注
			kdMigrator 2.0中PostgreSQL的Schema(namespace)将生成于K-DB Schema(=K-DB的User)。 但是PostgreSQL的User不被迁移到K-DB User(=K-DB Schema)。
Privilege	Privilege	✕	
✕	Public Synonym	✕	
Sequence	Sequence	✕	
Table	Table	●	
Index	Index	●	Unique index/constraint全部处理为index。
Constraint	Constraint	●	对Primary Key、Foreign Key、Check支持迁移。 – Primary Key index/constraint全部处理为constraint。 – constraint的表达式是利用保存在PostgreSQL的DD的语句来生成DDL，即使K-DB和由大小写处理方式差异会迁移失败。 – 不支持Foreign Key的update规则，delete规则仅支持CASCADE和SET NULL两种。.
Synonym	Synonym	✕	
Materialized View	Materialized View	✕	
View	View	✕	
✕	PSM	✕	

将PostgreSQL迁移到K-DB时数据类型将会如下转换。

PostgreSQL	迁移**Data Type**	备注
smallint	NUMBER(5)	
integer	NUMBER(10)	
bigint	NUMBER(19)	
decimal	NUMBER	
numeric	NUMBER	
real	BINARY_FLOAT	
double precision	BINARY_DOUBLE	

PostgreSQL	迁移**Data Type**	备注
float	BINARY_FLOAT或 BINARY_DOUBLE	1 ~ 24 precision迁移到BINARY_FLOAT，25 ~ 53将迁移到 BINARY_DOUBLE。
money	NUMBER	
character	CHAR	
character varying	VARCHAR	
text	VARCHAR(65532)	
date	DATE	
time	TIME(6)	
time with time zone	TIME(6)	迁移到timezone反映的time值。
timestamp	TIMESTAMP(6)	
timestamp with time zone	TIMESTAMP(6) WITH TIME ZONE	反映查看时接收的time zone。
interval	未支持	
boolean	CHAR(1)	
cidr	VARCHAR(43)	
inet	VARCHAR(43)	
macaddr	VARCHAR(17)	
BIT	VARCHAR	
BIT VARYING	VARCHAR	
bytea	RAW(2000)	
uuid	VARCHAR(40)	
xml	XMLTYPE	

参考

1. 不支持Array类型列。

2. 列的添加仅支持nullable和default value设置。

2.4. 访问用户的权限

访问Source和Target数据库时用户要被赋予迁移操作所需的权限。为赋予正确的权限时迁移将失败。例如没有select any table权限时，会发生无法查找迁移表的情况。主要对赋予DBA权限的用户推荐使用，实际所需的详细权限目录会根据数据库的种类或Option界面上所选的需要迁移的对象种类有所不同。

例如，从Oracle向Full Mode迁移时，访问Source数据库时用户要被赋予如下权限。

- CONNECT
- SELECT ANY TABLE
- SELECT ANY DICTIONARY
- ALTER SESSION

Target数据库为K-DB时在访问时用户会被赋予如下权限。

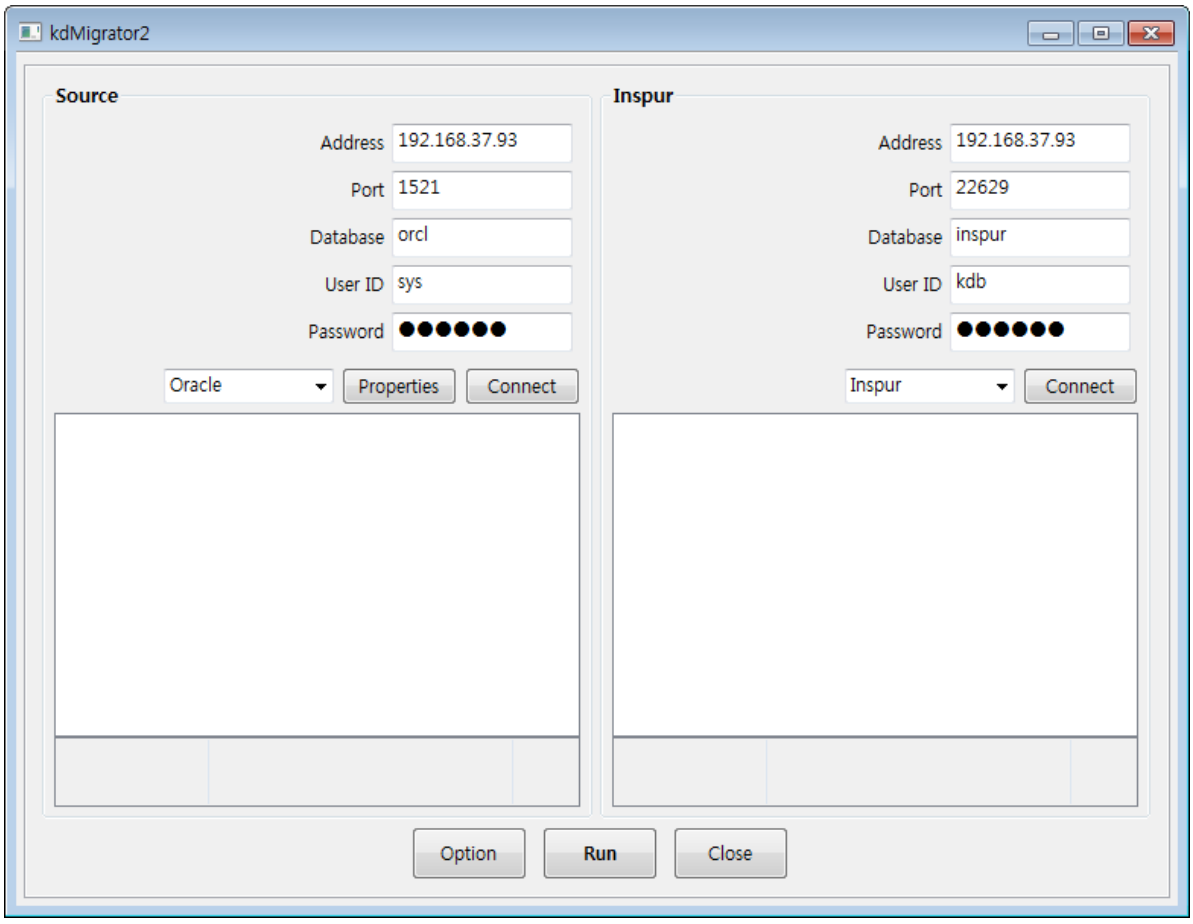
- CONNECT
- SELECT ANY TABLE
- SELECT ANY DICTIONARY
- RESOURCE
- ALTER SESSION

2.5. 运行示例

下面是使用kdMigrator 2.0过程的相关说明。

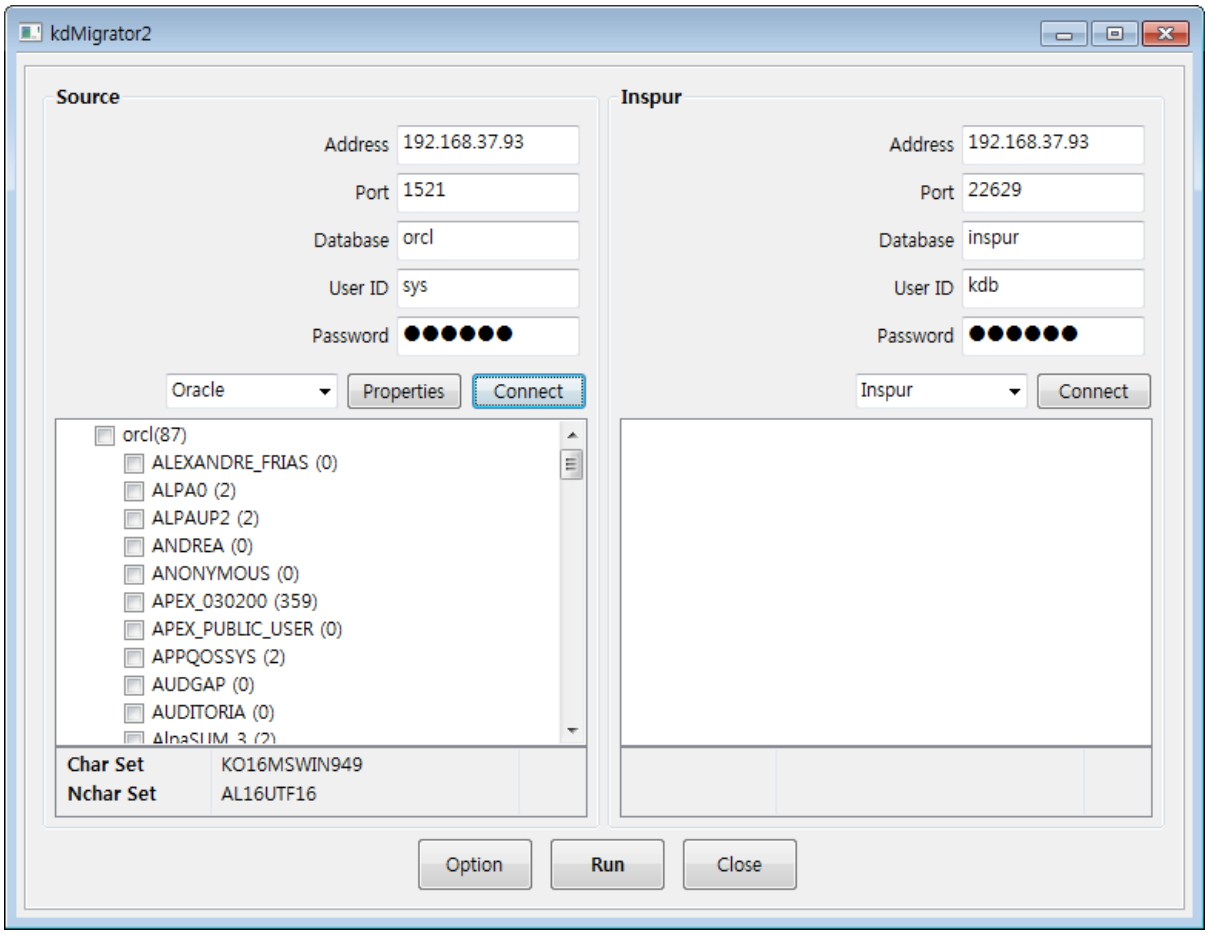
1. 运行kdMigrator 2.0 Utility时，出现如下初始界面。

[图 **2.9]** 迁移 **-** 初始界面



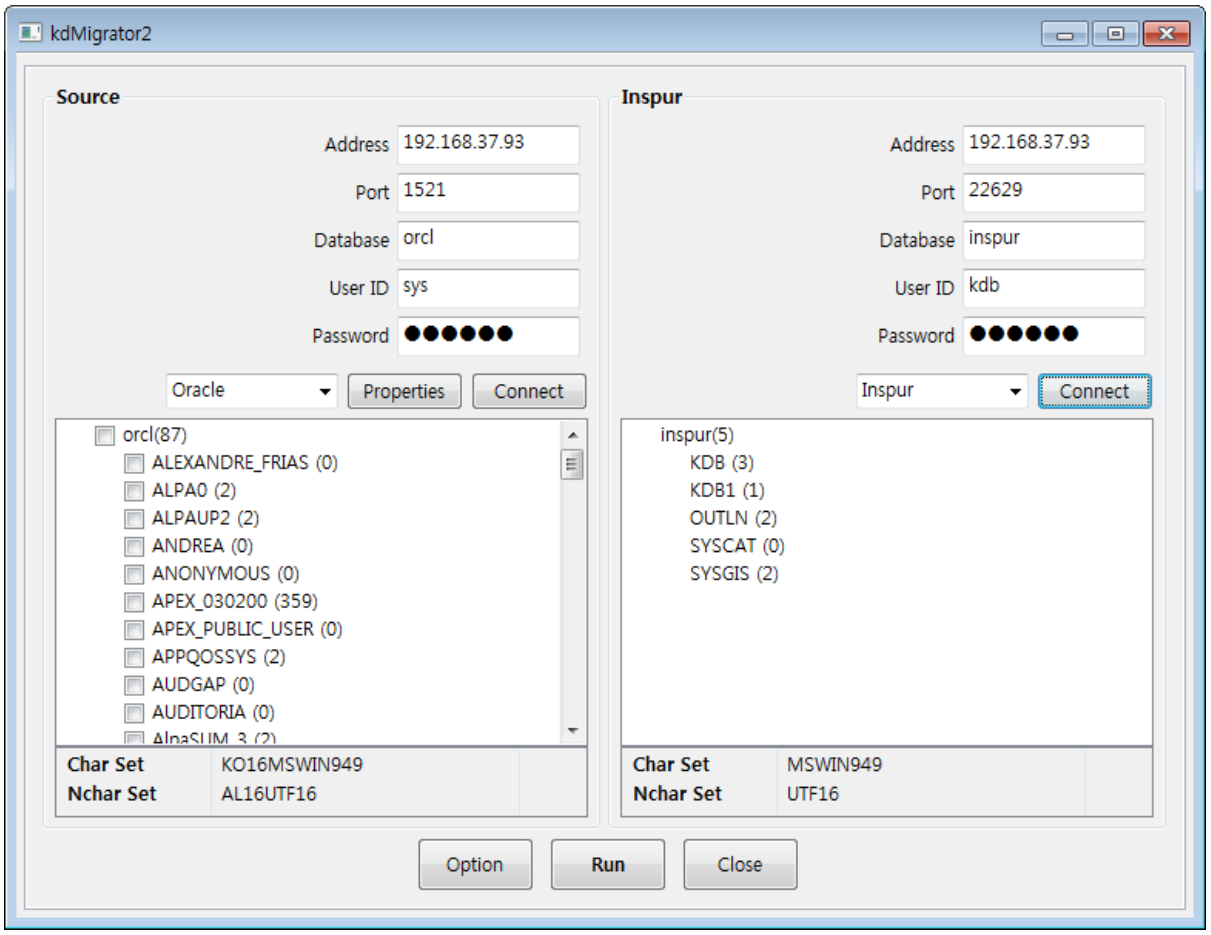
2. 要连接的Source数据库用户ID、密码等输入完毕后点击**[Connect]**按钮。

[图 **2.10]** 迁移 **-** 输入源数据库连接信息



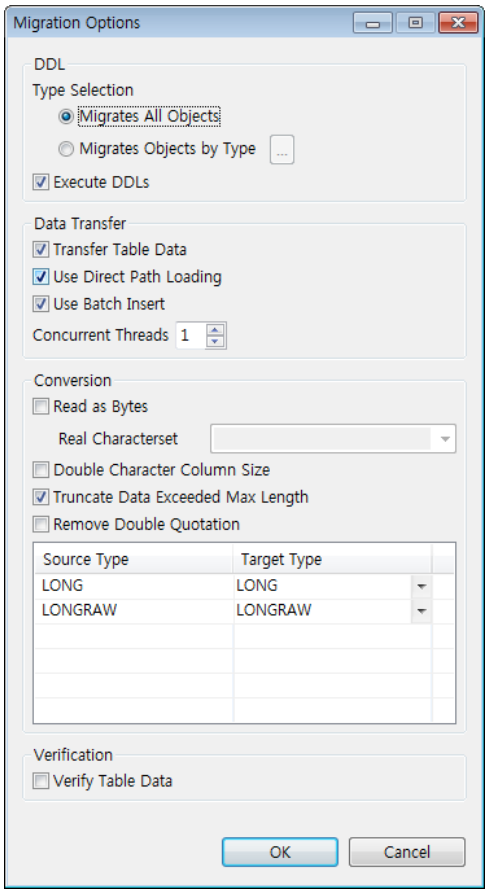
3. 要连接的K-DB数据库用户ID, 密码等输入完毕后点击**[Connect]**按钮。

[图 **2.11]** 迁移 **-** 输入目标数据库连接信息



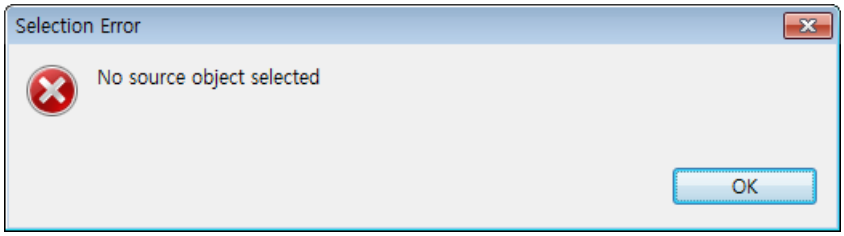
4. 点击**[OPTION]**按钮。选项信息设置结束后点击**[OK]**按钮。

[图 **2.12]** 迁移 **- Migration Options**输入界面



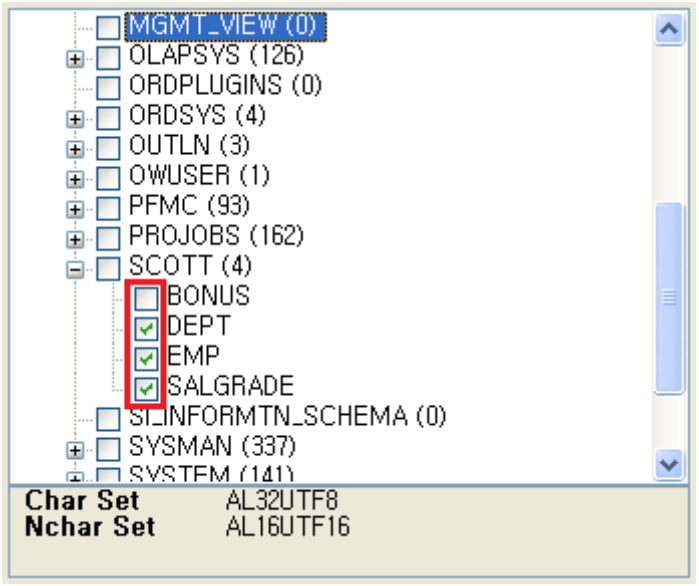
5. 在Source数据库视图中不做任何选择点击**[Run]**按钮时，将弹出如下警告窗。

[图 **2.13]** 迁移 **-** 不做选择运行时的警告窗



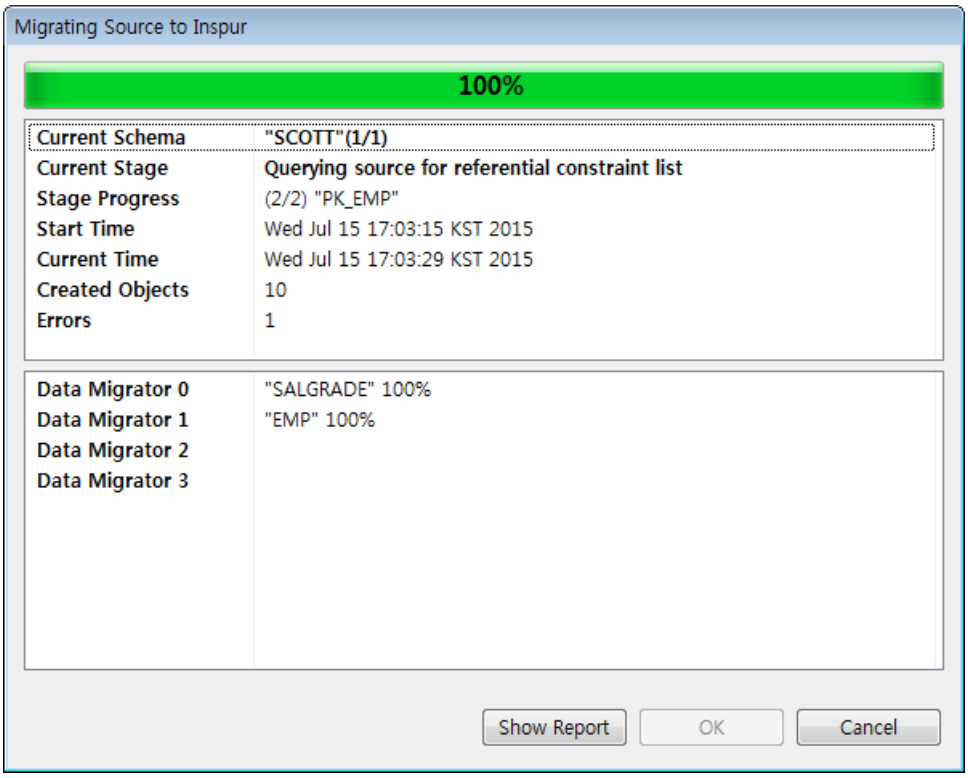
6. 在Source数据库视图中选择要迁移的目标点击**[Run]**按钮，进行迁移。

[图 **2.14]** 迁移 **-** 选择后运行



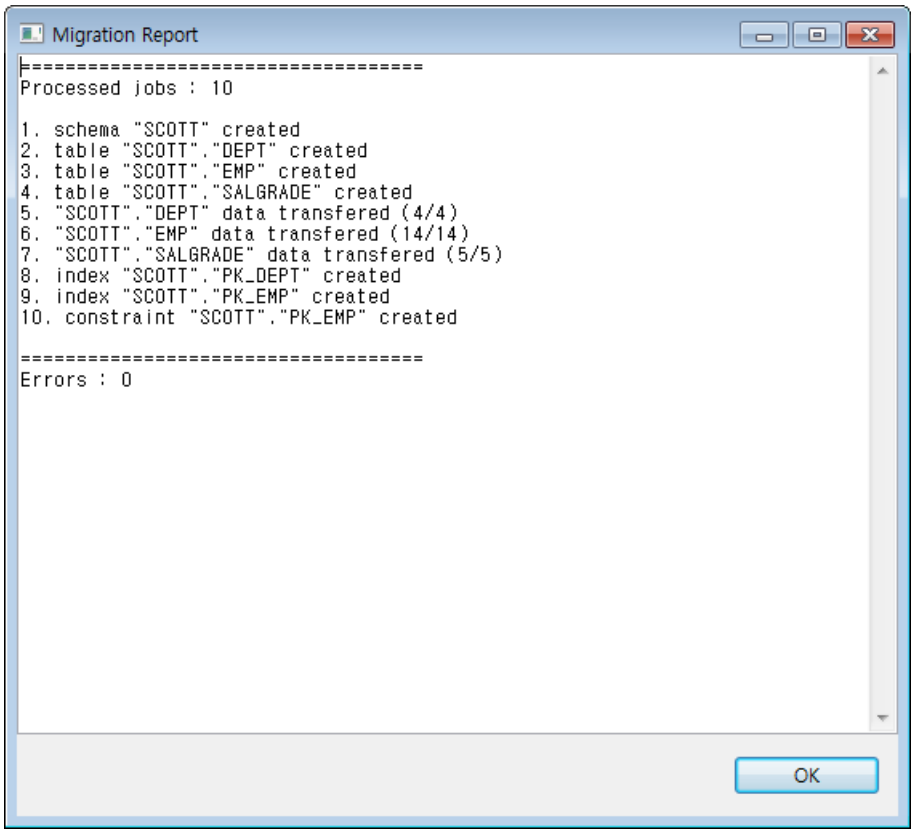
7. 进行迁移时，如下出现显示进行情况的**Progress**对话窗。并且在界面下端视图中可以确认进行情况日志。

[图 **2.15]** 迁移 **-** 进行迁移



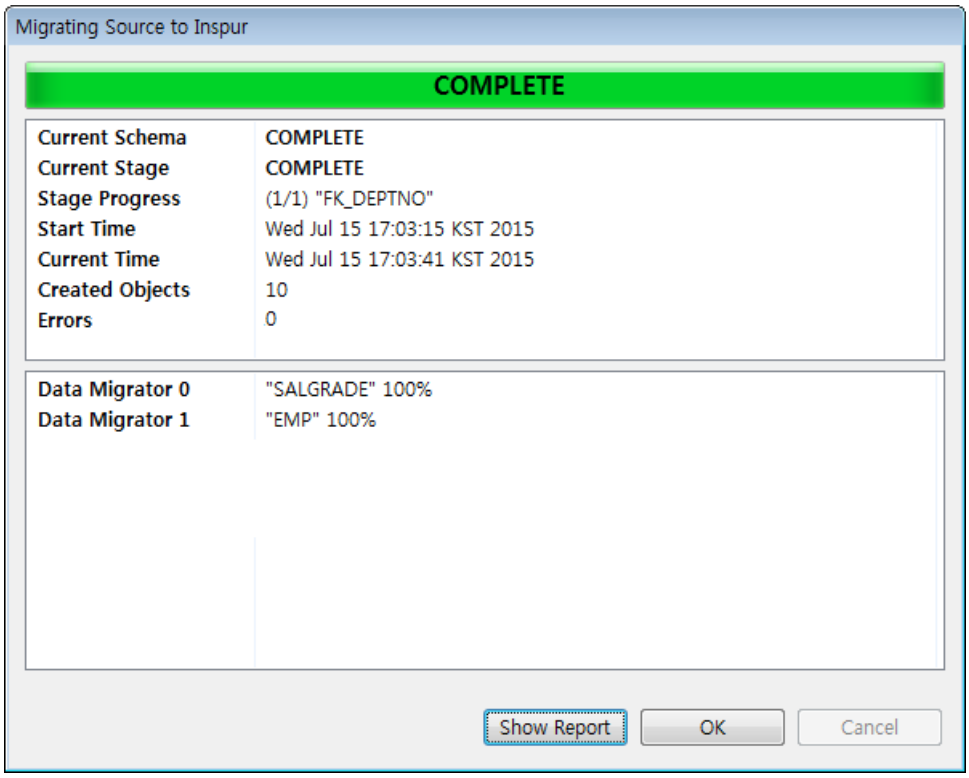
8. 进行当中或终止后点击Show Report时将出现如下Report界面，在此可以确认迁移进行记录。

[图 **2.16]** 迁移 **- Report**界面



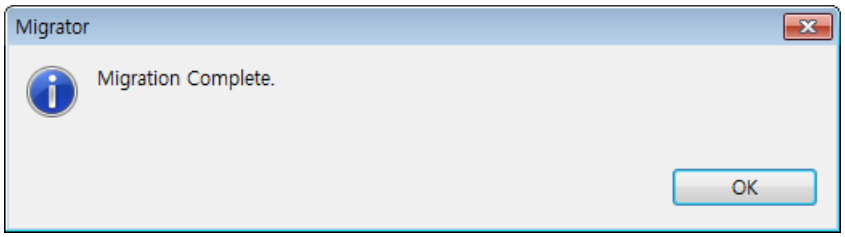
9. 所有迁移步骤结束时，如下在Progress对话框的最上端显示COMPLETE。点击**[OK]**按钮。

[图 **2.17]** 迁移 **-** 迁移步骤结束



10. 所有步骤结束后关闭Progress对话框时，将出现如下提醒迁移结束的对话框。点击**[OK]**按钮。

[图 **2.18]** 迁移 **-** 结束界面



第3章 kdExport

本章将介绍kdExport Utility并说明使用方法。

3.1. 概要

kdExport是K-DB中提供的Export Utility。通过该Utility，抽取保存在K-DB数据库当中的所有或部分Schema来保存为固有格式的文件，因此有利于数据库的备份与其他机器间的传递。

若在kdExport Utility当中抽取一个Schema Object，则与其相关的Schema Object将会被一起抽出。例如，若抽出一个表，则该表里所生成的索引和约束条件等都会被一起抽出。为此可根据需要指定不抽取的一部分相关Schema Object。

Export模式里有所有数据库模式、使用模式、表模式。其中所有数据库模式只可以使用DBA。

作为运行kdExport Utility的结果生成的文件为操作系统文件。因此与K-DB数据库文件不同，可以运行一般文件等操作。例如，可以利用FTP传送文件或保存在CD-ROM来移动到远程K-DB数据库当中。

运行Export过程当中所发生的日志使用LOG参数指定。

下面是运行kdExport实用程序的结果，即对完成、警告、错误消息的说明。

项目	说明
完成消息	Export成功结束后被输出。
警告消息	Export结束后发生警告时输出。 试图输出不存在的表时，kdExport Utlity输出警告消息后，跳过该表，继续输出下一个对象。
错误消息	运行Export的过程当中因发生错误而不能继续Export时输出。 系统内存不足或没有生成Export所需的视图等情况一样不能继续进行Export时被输出，输出错误消息以后结束Export会话。

3.2. 特征

kdExport实用程序的特征如下。

- 逻辑备份

  K-DB的内部Schema与数据以SQL语句输出。

- 不同时间的数据

  将多个表Export时抽出的各表数据不是同一时刻的数据，而是进行Export操作时的顺次数据。

- 保存表定义

  与是否存在数据无关保存表定义（表的DDL脚本）。

- 表的重新构成

  生成表后，删除由大量DML操作发生移植的行（migrated row）或碎片（fragmentation）。

3.3. 快速开始

kdExport Utility在安装K-DB过程当中会被一起安装，若删除K-DB，则该实用程序会被一起卸载。并且它由Java语言实现，在安装了JVM（Java Virtual Machine）的任意平台当中都可以直接操作。

3.3.1. 前期准备事项

运行kdExport Utility之前需要准备如下事项。

- 安装JRE1.4.2以上的版本。
- 安装在K-DB数据库服务器等平台当中或用网络连接。
- 运行所需的类库（默认位置：$KD_HOME/client/lib/jar目录）
  - kdExportexpimp.jar
  - 实用程序共享库：toolcom.jar
  - 实用程序共享Logger库：msllogger-14.jar
  - JDBC Driver：internal-jdbc-14.jar

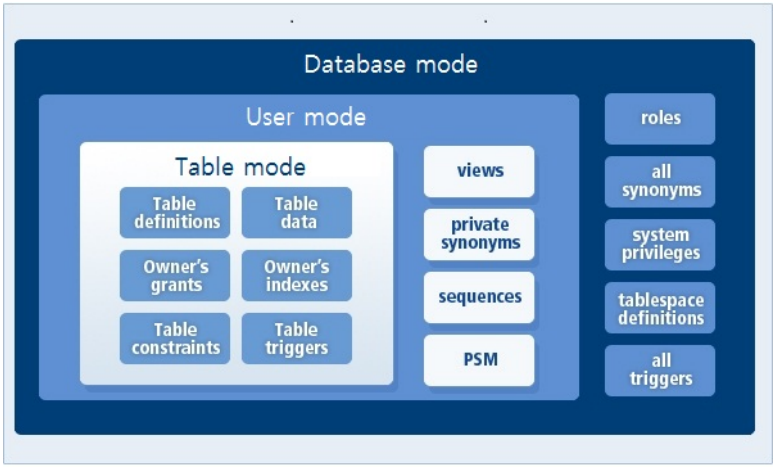
运行所需的类库在安装K-DB的过程当中被一起安装，因此无需添加操作。

3.3.2. Export模式

Export模式中有所有数据库模式、用户模式、表模式。各模式可以使用并指定参数。

下图显示按各模式Export的Schema Object所包含的关系。

[图 **3.1] Export**模式



整个数据库模式

整个数据库模式作为输出的文件当中Import整个K-DB数据库的模式，只可以使用DBA的模式。用于输出除了SYS用户之外的所有用户对象。

若要使用整个数据库模式，需要如下将FULL参数设置为Y。

```
FULL=Y
```

用户模式

用户模式是Export指定对象用户和指定的用户所拥有的所有Schema的模式。指定的用户为了Export使用所拥有的对象，DBA是只有一个以上的用户才可以使用此模式。

若要使用用户模式，以USER=userlist格式设置下面的USER参数。

```
USER=SCOTT, USER1, …
```

Table模式

表模式指定一个以上的表，是与该表相关的索引等Schema Object一起输出的模式。

若要使用表模式，如下以TABLE=tablelist格式设置TABLE参数。需要注意的是必须指明像SCOTT.EMP一样持有表的用户。

```
TABLE=SCOTT.EMP, USER1.TABLE1, …
```

3.3.3. 运行

若要运行kdExport Utility，在**$KD_HOME/client/bin**目录里输入**kdexport**命令语句。

下面是以所有数据库模式运行的示例。

[例 **3.1] kdExport Utility**的运行

```
$ kdexport username=kdb password=inspur file=export.dat full=y
$ kdexport cfgfile=export.cfg
```

3.4. 命令提示符下的参数指定

用户不指定参数值运行kdExport时，会如下显示命令提示符当中可以指定的参数目录和使用方法。

```
kdExport 11.0 103119 Inspur Corporation Copyright (c) 2008-. All rights reserved.
Usage: kdexport [PARAMETER1=VALUE] [PARAMETER2=VALUE] ...
Parameters:
  CFGFILE      Config file name
  CONSTRAINT   Export Constraint: Y/N, default: Y
  CONSISTENT   Consistent Mode: Y/N, default: N
  EXCLUDE      Limit the export to specific objects
  FILE         Export dump file name, default: default.dat
  FULL         Full Mode: Y/N, default: N
  GEOM_ASBYTES Export the geometry columns as bytes, default: Y
  GRANT        Export Grant: Y/N, default: Y
  INDEX        Export Index: Y/N, default: Y
  INLINE_CONSTRAINT Use the Inline Constraint: Y/N, default: N (this option
               is only supported for the not null)
  IP           IP address, default: localhost
  LOG          Export log file name
  LOGDIR       Export log directory
  NO_PACK_DIR  Export unpacked dump files to specified directory.
               If this option is specified, FILE parameter will be ignored.
  OVERWRITE    Overwrite datafile if same file name exists
  PASSWORD     User password
  PORT         PORT number, default: 8629
```

```
QUERY        Where predicate: (Optional) to filter data to be exported
             (must be used with TABLE parameter.)
REMAP_TABLESPACE Remaps the objects from the source tablespace
             to the target tablespace.
ROWS         Export Table Rows: Y/N, default: Y
SAVE_CREDENTIAL    Save your username and password to specified file
SCRIPT       LOG THE DDL SCRIPT: Y/N, default: N
SID          Database name
TABLE        Table Mode: table name list
TARGETDB     Target Server, default: K-DB
TEMP_DIR         Directory for the temporary raw dump files.
THREAD_CNT   Thread Count, default: 4
USER         User Mode: user name list
USERNAME     Database user name
```

参数值不用指定顺序输入。虽然参数值中CFGFILE只可以在命令提示符当中指定，但其余的参数值在环境设置文件当中也可指定。

在环境设置文件里保存并管理命令提示符当中所使用的参数的方法有下面两种。第二种是指定一个以上参数值的情况。

```
PARAMETER=value
PARAMETER=value1, ...
```

下面是指定参数的示例。

```
FULL=Y
FILE=EXPORT.DAT
GRANT=Y
INDEX=Y
CONSTRAINT=Y
```

3.4.1. 参数目录

下面是在命令提示符当中可以指定的kdExport Utility的参数

项目	说明
CFGFILE	环境配置文件的名称。该文件中使用的所有参数名必须大写。
CONSISTENT	以执行Export的时间点为标准将数据Export的功能。 – Y: 以Consistent模式进行Export。 – N: 不以Consistent模式进行Export。（默认值） 不支持flashback query中不支持的对象。

项目	说明
CONSTRAINT	执行Export时指定是否使用约束条件的Export。 – Y: 将约束条件Export。（默认值） – N: 不将约束条件Export。

EXCLUDE	可以设置执行Export时需要排除的特定user和table。
	为了排除可以设置如下方法来设置。
	– schema和schema.table除外
	`exclude=schema:\"=\'K-DB\'/table:LIKE \'T%\'\"`
	`exclude=schema:\"IN\'USER1\'\"`
	– 多个schema除外
	`exclude=schema:\"=\'K-DB\'\"`
	`exclude=schema:\"=\'USER1\'\"`
	– 多个table除外
	`exclude=table:\"LIKE\'E%\'\"`
	`exclude=table:\"LIKE\'E%\'\"`

FILE	执行Export时生成的文件名。（默认值：default.dat） 以二进制文件的形态在操作系统当中生成，未指定名时以默认值生成。
FULL	指定是否以所有数据库模式进行Export。 – Y：以所有数据库模式进行Export。 – N：以用户或表模式进行Export。（其中必须要有一个模式）
GEOM_ASBYTES	对geometry列决定是否用WKB或bytes来获取。 （默认值：N） – K-DB 11以上服务器中将Geometry列数据以WKB格式保存，因此export和import时使用该选项则在性能方面会有所提高。export的情况，若将geom_asbytes选项设置为'Y'，则无需使用如st_asbinary的函数可以直接处理LOB。 – 在K-DB 5 SP1以下版本的K-DB当中export时，为了用WKB格式来export，则要注意不能将geom_asbytes设置为'Y'。

项目	说明
	要将geom_asbytes设置为'N'使用，这样可以从内部使用st_asbinary()来以WKB格式接收数据。 从下级版本的K-DB服务器运行export时，会发生生成temp LOB来处理的性能问题和import时使用DPL时无法处理size较大的geometry数据。
GRANT	在Export时，指定是否输出权限。 – Y：输出权限。（默认值） – N：不输出权限。
INDEX	运行Export时，指定是否输出索引信息。 – Y：输出索引。（默认值） – N：不输出索引。
INLINE_CONSTRAINT	执行Export时是否将脚本以Inline Constraint输出。 – Y: 以Inline Constraint输出(仅在Not Null当中支持)。 – N: 以Out-of-line Constraint输出。（默认值）
IP	输入Export对象K-DB服务器IP地址。（默认值：localhost）
LOG	输入记录Export日志的文件名。
LOGDIR	输入保存记录Export日志的文件目录名。
NO_PACK_DIR	解压缩存储文件的保存目录。指定该选项时设置在FILE参数上的值将被忽略。
OVERWRITE	指定存在与执行Export时生成的文件名相同名称的文件时，是否覆盖文件。 – Y: 覆盖文件。 – N: 不覆盖文件。（默认值）
PASSWORD	输入执行Export的用户密码。
PORT	输入Export对象K-DB服务器的端口号。（默认值：8629）
QUERY	指定Export数据的过滤条件。 – 忽略模式运行，但要注意有可能会应用在不必要的表当中。 – 要将Where条件前后用扩上"\。但要向条件子句的内容进行String处理时，需要扩上"\"。 – 由指定的条件在SQL语句上发生语法(Syntax)错误时，在不应用条件的情况下重试。

项目	说明
REMAP_TABLESPACE	提供修改Tablespace名的功能。 可以用如下方法设置。 在Tablespace USR1修改为USR3 – REMAP_TABLESPACE=usr1:usr3 修改多个tablespace的设置 – REMAP_TABLESPACE=usr1:usr3,usr2:usr4 区分大小写设置 – REMAP_TABLESPACE=\"Usr1\":usr3
ROWS	在Export时，指定是否输出表数据。 – Y：输出表数据。（默认值） – N：不输出表数据。

SAVE_CREDENTIAL	使用加密的USERNAME和PASSWORD时设置。
	● 使用方法
	– 使用SAVE_CREDENTIAL选项来生成EXPIMP_WALLET加密文件。
	– SAVE_CREDENTIAL=[EXPIMP_WALLET_FILE_NAME]
	例）
	`SAVE_CREDENTIAL=/tmp/.expimp`
	`USERNAME=username PASSWORD=password`
	● 设置EXPIMP_WALLET文件环境变量
	`export SAVE_CREDENTIAL=/tmp/.expimp`
	● 识别优先级
	– 首先确认输入在command line的username和password参数。
	– cfgfile内USERNAME和PASSWORD文件
	– 读取EXPIMP_WALLET文件的USERNAME和PASSWORD。
	– 未指定如上设置时发生错误。

项目	说明
SCRIPT	在Export时，指定是否指明生成Schema Object的DDL脚本。 – Y：指明生成Schema Object的DDL脚本。 – N：不指明生成Schema Object的DDL脚本。（默认值）
SID	输入Export对象K-DB服务器的SID。
TABLE	以表模式执行Export时，指定要Export的对象表名称。 以TABLE=tablelist的形态使用。
TEMP_DIR	指定执行Expor时生成临时dump file的目录。
THREAD_CNT	输入为输出表数据而使用的线程个数。 （默认值：4）
USER	以用户模式执行Export时，指定Export的对象所有者。 以USER=userlist形态使用。
USERNAME	输入执行Export的用户账户。

3.5. 执行示例

下面是利用kdExport Utility运行Export的示例。

[例 **3.2]** 使用**kdExport Utility**的**Export**的运行

```
kdExport 11.0 97435 Inspur Corporation Copyright (c) 2008-. All rights reserved.
the entire database: Fri Feb 06 10:46:17 KST 2015
Export character set: MS949
    exporting tablespaces
    exporting roles
    exporting schema: "K-DB"
        exporting tables
          [0] exporting table BONUS     no rows exported.
          [1] exporting table DEPT      4 rows exported.
          [2] exporting table EMP       10 rows exported.
          [3] exporting table SALGRADE  5 rows exported.
        exporting object privileges
        exporting indexes
        exporting sequences
        exporting views
        exporting synonyms
Packing the file...
Export completed successfully: Fri Feb 06 10:47:17 KST 2015
```

第4章 kdImport

本章将介绍kdImport Utility说明书的使用方法。

4.1. 概要

kdImport是K-DB中提供的Import Utility。通过该Utility将保存在外部文件当中的Schema Object重新保存在K-DB数据库当中，因此有利于与kdExport Utility一起备份数据库，以及在其他machine之间的数据库传送是有用。kdImport Utility在功能方面与kdExport Utility对称或类似的情况较多。

若保存一个Schema Object，与其相关的Schema Object会被自动保存。按需要可以指定不保存相关的一部分Schema Object。

Import模式当中与kdExport里的一样有整个数据库模式、用户模式、表模式。其中整个数据库模式只可以使用DBA。

进行Import的过程当中所发生的日志使用LOG参数指定。

下面是对运行kdImport实用程序的结果，即完成、警告、错误消息进行说明。

项目	说明
完成消息	Import成功结束后输出。
警告消息	Import结束后发生警告时输出。
错误消息	运行Import的过程当中因发生错误而不能继续Import时输出。

4.2. 快速开始

kdImport Utility在安装K-DB的过程当中将被一起安装，卸载K-DB也将一起卸载实用程序。并且它由Java语言实现，在安装了JVM（Java Virtual Machine）的任意平台当中都可以直接操作。

4.2.1. 前期准备事项

在运行kdImport Utility之前，需要准备的事项如下。

- 安装JRE1.4.2以上的版本。
- 安装在K-DB数据库服务器等平台当中或用网络连接。
- 运行时所需的类库（默认位置：$KD_HOME/client/lib/jar目录）。

– kdExport class：expimp.jar

– 实用程序共享库：toolcom.jar

– 实用程序共享Logger库：msllogger-14.jar

– JDBC Driver：internal-jdbc-14.jar

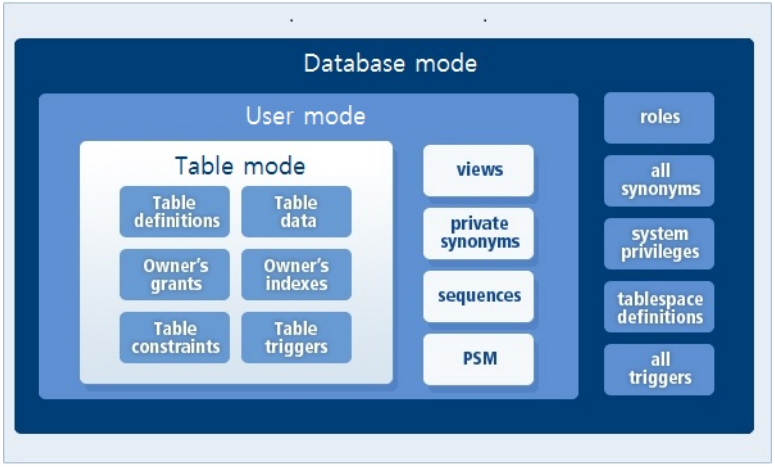
运行所需的类库在安装K-DB的过程当中被一起安装，因此无需添加操作。

4.2.2. Import模式

Import模式与Export模式相同有整个数据库模式、用户模式、表模式。Export的文件通过source按各模式的特性import数据。各模式可以使用参数指定。

下图显示按各模式Import的Schema Object所包含的关系。

[图 4.1] Import模式



整个数据库模式

整个数据库模式作为输出的文件当中Import整个K-DB数据库的模式，只可以使用DBA的模式。用于输出除了SYS用户之外的所有用户对象。

若要使用整个数据库模式，需要如下将FULL参数设置为Y。

```
FULL=Y
```

用户模式

用户模式是import对Export的文件指定的用户所持有的整个Schema Object的模式。DBA是只有一个以上的用户才可以使用此模式。

为了使用用户模式，将USER参数设置为USER=userlist格式。

```
USER=SCOTT, USER1, …
```

Table模式

Table模式是从Export的文件指定一个以上的表来将该表所相关的索引等的Schema Object一起Import的模式。

若要使用用户模式，使用TABLE=tablelist格式设置TABLE参数。需要注意的是必须指明像SCOTT.EMP一样持有表的用户。

```
TABLE=SCOTT.EMP, USER1.TABLE1, …
```

From User To User模式

From User To User模式是更改并import从Export的文件指定为FROMUSER参数的用户到指定为TOUSER参数的用户的相关对象的所有者。DBA是由一个以上的用户才可以使用此模式。

为了使用From User To User模式，需要通过下面的格式设置参数。

```
FROMUSER=SCOTT,USER1 TOUSER=USER2,USER3…
```

4.2.3. 运行

若要运行kdImport Utility，则在**$KD_HOME/client/bin**目录里输入**kdimport**命令。

下面是运行所有数据库模式的示例。

[例 **4.1] kdImport**实用程序的运行

```
$ kdimport username=kdb password=inspur file=export.dat full=y
$ kdimport cfgfile=import.cfg
```

4.3. 运行方法

4.3.1. 存在约束条件的表的Import

在运行kdImport Utility当中，若表已设置约束条件，则违反该约束条件的行不会被保存。

若按照kdImport Utility的运行顺序，比起表数据线先保存表约束条件，则会因违反约束条件而产生未保存的行。例如，在设置好参照完整性约束条件的两个表当中，若子表的数据比父表先被保存，子表中一个行将不被保存。

因此Import有约束条件的表时，先输入所有表数据，最后设置表约束条件。

4.3.2. 支持兼容的表的Import

在运行kdImport Utility之前，在数据库里用户可以直接定义相同名称的表。这时候新定义的表要在与要运行kdImport Utility运行的表可维持兼容的范围内分别定义。

若要维持表之间的兼容性，需要注意以下事项。

- 运行kdImport实用程序的表

  必须包括该表所包括的所有列。

- 数据类型

  必须要有兼容性，不能改变默认值。

- 添加新的列时

  不能设置NOT NULL或基本关键约束条件。

4.3.3. 在已存在的表中Import数据

在数据库里已有的同名表当中也可运行kdImport。此时，与前面所说明的一样，需要维持两个表之间的兼容性。

在运行kdImport Utility之前，用户定义同名表或数据库内已有同名表时，根据表中已设好的约束条件，会产生没有保存在kdImport Utility的行。

在已存在的表当中Import数据时，可以使用以下两种方法。

- 约束条件的停止

  运行kdImport Utility时，暂时停止约束条件。

- 调节运行顺序

运行kdImport Utility时调节运行顺序。例如，在设置好参照完整性约束条件的两个表当中先保存父表，最后保存子表。

表过大时，利用调节顺序的第二个解决方法在性能方面更有利一些。而暂时停止约束条件的方法，在重新允许约束条件情况，会检查表内所有行的约束条件是否成立，因此会给性能带来不利的影响。

4.4. 命令提示符中的参数指定

用户不指定参数值运行kdImport Utility时，会如下面显示命令提示符当中可以指定的参数目录和使用方法。

```
kdImport 11.0 102146 Inspur Corporation Copyright (c) 2008-. All rights reserved.
Usage: kdimport [PARAMETER1=VALUE] [PARAMETER2=VALUE] ...
Parameters:
    BIND_BUF_SIZE     Specify the buffer size of DPL stream, default: 1M(1048576)
    CFGFILE           Config file name
    COMMIT            Commit after the insertion, default: N
    CONSTRAINT        Import Constraint: Y/N, default: Y
    DBLINK            Import DB Link: Y/N, default: Y
    DPL               Use Direct Path Load: Y/N, default: N
    EXP_SERVER_VER    Specify the exported server version, default: 8
    FILE              Import dump file name, default: default.dat
    FROMUSER          FromUser toUser Mode: user name list
                      (must be used with TOUSER parameter)
    FULL              Full Mode: Y/N, default: N
    GRANT             Import Grant: Y/N, default: Y
    GEOM_ASBYTES      Import the data to the geometry columns as bytes, default: Y
    IGNORE            Ignore create error due to object existence: Y/N, default: N
    INDEX             Import Index: Y/N, default: Y
    IO_BUF_SIZE       Specify the buffer size of file I/O, default: 16M(16777216)
    IP                IP address, default: localhost
    LOG               Import log file name
    LOGDIR            Import log directory
    NATIONAL_CHARSET  Specify the exported national character set,
                      default is the exported character set
    NO_PACK_DIR       Import unpacked dump files from specified directory.
                      If this option is specified, FILE parameter will be ignored.
    PASSWORD          User password
    ROLE              Import Role: Y/N, default: Y
    PORT              PORT number, default: 8629
    PSM               Import PSM: Y/N, default: Y
    P_DPL             Use Parallel DPL: Y/N, default: N
    ROWS              Import Table Rows: Y/N, default: Y
    SAVE_CREDENTIAL   Save your username and password to specified file
    SCRIPT            LOG THE DDL SCRIPT: Y/N, default: N
    SEQUENCE          Import Sequence: Y/N, default: Y
    SID               Database name
```

```
    SYNONYM           Import Synonym: Y/N, default: Y
    TABLE             Table Mode: table name list
    TEMP_DIR          Directory for the raw dump files.
    THREAD_CNT        Thread Count, default: 4
    TOUSER            FromUser toUser Mode: user name list
                      (must be used with FROMUSER parameter)
    TRIGGER           Import Trigger: Y/N, default: Y
    USER              User Mode: user name list
    USERNAME          Database user name
```

由于已经指定了顺序，无需输入参数值。虽然参数值中CFGFILE只可以在命令提示符当中指定，但其余的参数值在环境设置文件当中也可指定。

在环境设置文件里保存并管理命令提示符当中所使用的参数的方法有下面两种。第二种是指定一个以上参数值的情况。

```
PARAMETER=value
PARAMETER=value1, ...
```

下面是指定参数的示例。

```
FULL=Y
FILE=KDEXPORT.DAT
GRANT=Y
INDEX=Y
CONSTRAINT=Y
```

4.4.1. 参数目录

下面是在命令提示符里可以指定的kdImport Utility的参数。

项目	说明
BIND_BUF_SIZE	调整将Import以DPL模式运行时在stream当中使用的bind buffer的大小。（默认值：1MB(1048576)）
CFGFILE	环境配置文件名。
COMMIT	insert任务之后执行commit。（默认值：N） insert任务单位如下。 – 向CPL进行import时超过bind insert buffer size即1M时默认执行commit。 如果有LONG, LONG RAW列时，以row单位执行commit。 – 以DPL执行import时，超过以BIND_BUF_SIZE指定的大小时执行commit。

项目	说明
CONSTRAINT	指定在运行Import时，是否Import约束条件。 – Y：Import约束条件。（约束条件） – N：不Import约束条件。
DPL	指定是否以DPL方法执行Import。 – Y: 使用DPL方法。 – N: 不使用DPL方法。（默认值）
DBLINK	执行Import时是否指定DBLink Import。 – Y：指定DBLink Import。（默认值） – N：不指定DBLink Import。
EXP_SERVER_VER	设置Export的服务器版本。（默认值：8） – K-DB 11: 8 – K-DB 1: 7 – K-DB 5: 6
FILE	执行Importt时生成的文件名。（默认值：default.dat） 以二进制文件的形态在操作系统当中生成，未指定名时以默认值Import。
FROMUSER	指定在From to User模式中使用，并在Export时使用的对象原来持有者。 以FROMUSER=userlist形态使用。
FULL	指定是否以所有数据库模式进行Import。 – Y：以所有数据库模式进行Import。 – N：以用户或表模式进行Import。（其中必须要有一个模式）（默认值）
GRANT	指定在执行Import时是否将权限Import。 – Y: 将权限Import。（默认值） – N: 不将权限Import。
GEOM_ASBYTES	对geometry列决定是否用WKB或bytes来获取。

项目	说明
	（默认值：Y） – K-DB 11以上服务器中将Geometry列数据以WKB格式保存，因此无需使用该选项。该选项只能在K-DB 5SP1以下版本的K-DB当中将Export的数据进行Import来设置时使用。 – 在K-DB 5 SP1以下版本的K-DB当中Export时将geom_asbytes设置为'N'来将Geometry列以WKB格式接收时可以将geom_asbytes选项设置为'N'。内部进行st_geomfromwkb。 如果用DPL时，要将服务器的'_DP_IMPORT_GEOM_FROM_OLD_FORMAT'（默认值：N）iparam设置为'Y'。
IGNORE	忽略运行Import时由已存在的Schema Object而引起的生成错误。 – Y：忽略由已存在的Schema Object引起的生成错误。 – N：不忽略由已存在的Schema Object引起的生成错误。 （默认值）
INDEX	运行Import时，指定是否输入索引信息。 – Y：输入索引。（默认值） – N：不输入索引。
IO_BUF_SIZE	调节在运行Import时，用于文件的输入/输出的buffer的大小。 （默认值：16MB(16777216)）
IP	输入Import对象K-DB服务器IP地址。（默认值：localhost）
LOG	输入记录Import日志的文件名。
LOGDIR	输入保存记录Import日志的文件目录名。
NATIONAL_CHARSET	设置Import的语言设置。（默认值：Import的字符集）
NO_PACK_DIR	解压缩存储文件的保存目录。指定该选项时设置在FILE参数上的值将被忽略。
PASSWORD	输入执行Import的用户密码。
ROLE	指定执行Import时是否输入ROLE。 – Y: 输入ROLE。（默认值） – N: 输入ROLE。
PORT	输入Import对象K-DB服务器的端口号。（默认值：8629）

项目	说明
PSM	指定执行Import时是否输入PSM对象。 – Y: 输入PSM对象。（默认值） – N: 不输入PSM对象。
P_DPL	指定是否用并列DPL方法输入。 – Y：使用并列DPL方法。 – N：不使用并列DPL方法。（默认值）
ROWS	在Export时，指定是否输入表数据。 – Y：输入表数据。（默认值） – N：不输入表数据。

SAVE_CREDENTIAL	使用加密的USERNAME和PASSWORD时设置。
	● 使用方法
	– 使用SAVE_CREDENTIAL选项来生成EXPIMP_WALLET加密文件。
	– SAVE_CREDENTIAL=[EXPIMP_WALLET_FILE_NAME]
	例）
	`SAVE_CREDENTIAL=/tmp/.expimp`
	`USERNAME=username PASSWORD=password`
	● 设置EXPIMP_WALLET文件环境变量
	`export SAVE_CREDENTIAL=/tmp/.expimp`
	● 识别优先级
	– 首先确认输入在command line的username和password参数。
	– cfgfile内USERNAME和PASSWORD文件
	– 读取EXPIMP_WALLET文件的USERNAME和PASSWORD。
	– 未指定如上设置时发生错误。

SCRIPT	在Import时，指定是否指明生成Schema Object的DDL脚本。 – Y：指明生成Schema Object的DDL脚本。

项目	说明
	– N：不指明生成Schema Object的DDL脚本。（默认值）
SEQUENCE	指定Import时是否输入Sequence。 – Y：输入Sequence。（默认值） – N：不输入Sequence。
SID	输入Import对象K-DB服务器的SID。
SYNONYM	指定执行Import时是否输入Synonym的Import。 – Y: 输入Synonym。（默认值） – N：不输入Synonym。
TABLE	以表模式执行Import时，指定要Export的对象表名称。 以TABLE=tablelist的形态使用。
TEMP_DIR	指定执行Impor时生成临时dump file的目录。
THREAD_CNT	输入为输入表数据而使用的线程个数。 （默认值：4）
TOUSER	指定在From to User模式当中使用并执行Import时需要输入的所有者。 以TOUSER=userlist的形态使用。
TRIGGER	指定执行Import时是否指定Trigger的Import。 – Y: 输入Trigger。（默认值） – N: 不输入Trigger。
USER	以用户模式执行Import时，指定Import的对象所有者。 以USER=userlist形态使用。
USERNAME	输入执行Import的用户账户。

4.5. 执行示例

kdImport Utility执行Import的顺序如下。

1. 表定义

2. 表数据

3. 表索引

4. 表的约束条件、视图、存储过程等

下面是利用kdImport实用程序执行Import的示例。

[例 4.2] 利用**kdImport Utility**执行**Import**

```
kdImport 11.0 97819 Inspur Corporation Copyright (c) 2008-. All rights reserved.
Unpacking the file...
the entire database: Mon Jul 14 01:07:43 KST 2014
Import character set: MS949
The version of this kdExport dump file is 5.0.
  importing schema: "K-DB"
    importing tables
      [M] importing table BONUS     no rows imported.
      [0] importing table DEPT      4 rows imported.
      [0] importing table SALGRADE  5 rows imported.
      [1] importing table EMP       10 rows imported.
    importing index
    importing sequences
    importing views
    importing synonyms
Import completed successfully:  Mon Jul 14 01:07:55 KST 2014
```

第5章 kdLoader

本章将介绍kdLoader Utility及其使用方法。

5.1. 概要

kdLoader是将大容量的数据一次保存在K-DB 数据库里的Utility。通过该Utility，不用挨个编辑SQL语句，在数据库里输入要输入的数据，只要创 建一般文本文件一次性加载列数据便可。因此该Utility有利于将大量数据一次性保存在K-DB的数据库。

若使用kdLoader，用户可以方便编辑数据文件，还可以缩短加载数据的时间。

5.2. 快速开始

kdLoader Utility在安装K-DB的过程中一起被安装，删除K-DB，则被一起删除。

kdLoader Utility在命令提示符里以下面的格式运行。

```
$ kdloader [options]
```

在命令提示符里可以指定的选项详细内容请参考「5.7. 指定命令提示符的参数」。

下面是运行kdLoader Utility的示例。

[例 5.1] kdLoader Utility的运行

```
$ kdloader userid=db_user/db_password@default
control=sample.ctl data=sample.data direct=Y
```

5.3. 输入输出文件

kdLoader Utility通过接收控制文件（Control file）和数据文件（Data file），输出日志文件（Log file）和错误文件（Bad file）。控制文件与数据文件作为输入文件由用户编写，日志文件与错误文件是在kdLoader Utility里自动生成的输出文件。kdLoader Utility的输入输出文件都是一般文本文件，因此用户生成输入文件很方便。

本节将说明使用kdLoader Utility时所需的输入文件，控制文件、数据文件和由加载结果输出的日志文件、错误文件。

5.3.1. 控制文件

控制文件是为运行kdLoader Utility的指定参数的文件。用户指明从控制文件里读取的数据位置和读取数据的具体方法，以及保存实际数据的位置。控制文件的参数在「5.9. 指定控制文件的选项」里有详细说明。

5.3.2. 数据文件

数据文件是要保存在数据所存在数据库表当中的文本文件。数据文件利用kdSQL的SPOOL命令语句保存SQL查询结果或用一般文本编辑器编辑。

用户指定的数据文件以固定记录格式（Fixed Record Format）、分隔记录格式（Separated Record Format）两种格式被保存。

5.3.2.1. 固定记录格式

是用户在控制文件里对所有列指明POSITION信息时使用的格式。此类型在字段之间不用分隔字符，而是在用户指明的列位置读取实际数据。但是列的位置由字节的长度决定。与分割记录格式相比，虽然灵活性不大，但性能方面比较突出。

用户为了分隔记录，指明固定长度的记录大小或使用系统的行尾（End of Line，以下EOL）字符。此外，在系统里，为了提高速度，可在控制文件里所指明的列位置上直接读取数据，因此忽略ESCAPED BY和LINE STARTOR参数。

因此，无法使用「5.9. 指定控制文件的选项」中的所有FIELDS语句和LINE STARTED BY语句，只能使用列的POSITION语句。

- 使用LINES FIX语句的示例

  是数据文件所包含的记录长度为固定长度（字节数）时使用的方式。用户在控制文件里必须指明LINES FIX 12一样的记录大小。

  使用LINES TERMINATED BY语句时则忽略。

  下面是固定的长度记录的示例。

```
example.ctl:
     LOAD DATA
     INFILE  'example.dat'
     LOGFILE 'example.log'
     BADFILE 'example.bad'
     APPEND
     INTO TABLE EMP
     LINES FIX 12
     (
         empno position (01:04),
         ename position (06:12)
     )
```

```
example.dat:
     7922 MILLER 7777 KKS    7839 KING   7934 MILLER 7566 JONES
```

- 不使用LINES FIX语句的示例

  用户若在控制文件里不指明记录大小，使用EOL字符（'\n'）分隔记录。

  下面是记录分隔符为EOL字符的示例。

```
example.ctl:
     LOAD DATA
     INFILE  'example.dat'
     LOGFILE 'example.log'
     BADFILE 'example.bad'
     APPEND
     INTO TABLE EMP
     TRAILING NULLCOLS
     (
         empno position (01:04),
         ename position (06:15),
         job   position (17:25),
         mgr   position (27:30),
         sal   position (32:39)
     )
```

```
example.dat:
     7922 MILLER     CLERK     7782 920.00
     7777 KKS        CHAIRMAN
     7839 KING       PRESIDENT      5500.0
     7934 MILLER     CLERK     7782 920.00
     7566 JONES      MANAGER   7839 3123.75
     7658 CHAN       ANALYST   7566 3450.00
     7654 MARTIN     SALESMAN  7698 1312.50
```

example.dat数据文件的第二行的最后两个列对应的值不存在，因此如果没有「5.9.15. TRAILING NULLCOLS语句」将会发生错误。

5.3.2.2. 分隔记录格式

是用户在控制文件里没有指明所有列的POSITION信息时所应用的格式。各字段被FIELD TERMINATOR分隔，各记录被LINE TERMINATOR分隔。与固定记录格式相比，在性能方面虽有不足，但用户可以指定所需格式，因此具有优越的灵活性。

下面是被分隔的记录格式示例。

```
example.ctl:
   LOAD DATA
```

```
   INFILE  'example.dat'
   LOGFILE 'example.log'
   BADFILE 'example.bad'
   APPEND
   INTO TABLE emp
   FIELDS TERMINATED BY ','
          OPTIONALLY ENCLOSED BY '"'
          ESCAPED BY '\\'
   LINES TERMINATED BY '\n'
   (
       empno,
       ename,
       job,
       mgr,
       hiredate,
       sal,
       comm,
       deptno
   )
```

```
example.dat:
   7654, "Martin", "Sales",7698,1981/10/28,1312.50,3,10
   7782, "\,Clark","Manager" ,7839, 1981/01/11 ,2572.50,10,20
   7839, "King",President,,1981/11/17,5500.00,,10
   7934,"Miller","Clerk",7782 ,1977/10/12,920.00,,10
   7566, "Jones",Manager" ,7839, 1981/04/02,3123.75,,20
   7658, "Chan", Ana lyst, 7566,1982/05/03,3450,,20
```

5.3.3. 日志文件

是记录了kdLoader Utility运行过程的文件。向用户与要输入的基本源数据一起提供实际输入成功的记录和失败记录的统计。此外，关于失败的记录，提供kdLoader Utility判断失败的理由。

5.3.4. 错误文件

是运行kdLoader Utility时，记录加载失败的数据文件。用户修改并重新加载包含失败记录的错误文件。

5.4. 加载方法

在kdLoader Utility中加载数据的方法有以下两种。

- Conventional Path Load

kdLoader Utility里提供基本的数据加载方法。通过在用户指定的数据文件中读取列数据放入列数组里成批处理（BATCH UPDATE）的方式将数据加载在数据库服务器里。虽然性能比Direct Path Load低，但具有没有其他制约事项的优点。

- Direct Path Load

  读取用户指定的数据文件，根据特定咧数据类型将数据制成列数组（column array）形态。列数组格式的数据通过数据块格式编辑器（block formatter）按K-DB数据库块形态创建，该数据块直接用于K-DB的数据库当中。

  Direct Path Load虽然比Conventional Path Load快很多，但有如下约束事项。

  – 不检查CHECK约束条件和参照约束（Referential Constraint）。

  – 检查主键约束（Primary Key Constraint）、唯一键约束（Unique Key Constraint）、NOT NULL约束。

  – 在加载中不会运行插入触发器（Insert trigger）。

5.5. 约束条件

本节中将说明kdLoader Utility的约束条件

5.5.1. 使用相同的分隔符

若FIELD TERMINATOR、ENCLOSED BY、ESCAPED BY、LINE TERMINATOR参数值中任意一个被指定为相同的值，就不能从数据文件里正常读取。.

例如，若指定下面的FIELD TERMINATOR和ENCLOSED BY参数，就会发生错误。

```
FIELDS TERMINATED BY '"'
       OPTIONALLY ENCLOSED BY '"'
```

5.5.2. 不指定 ESCAPED BY 参数值的情况

数据文件的输入字段值与指定为ENCLOSED BY、FIELD TERMINATOR、LINE TERMINATOR的字符相同时，则不能正确解释所输入字段。详细事项请参照「5.9.11. FIELDS语句的ESCAPED BY语句」。

5.6. 空白政策

在本节中将说明kdLoader Utility里处理空白（Whitespace）的方法。

5.6.1. 字段值全部为空白时

在一个记录的列值所输入相应文件字段值为空白时，kdLoader Utility将根据输入表的列数据类型加载0或NULL值。

5.6.2. 部分字段值为空白时

kdLoader Utility将空字符（''），tab字符（'\t'）与EOL字符（'\n'）视为空白。但是，相关字符声明为FIELD TERMINATOR或LINE TERMINATOR时不会被视为为空白。空白可以存在于字段的开始与结束。但是在字段中间的空白需处理为数据的一部分。kdLoader Utility对于空白将根据数据文件格式做不同的处理。

固定记录格式时

存在于字段值前面的空白，处理为实际数据，后面的空白被视为无必要的空白而被删除。因为用户虽然可以临时删除字段值前面的空白，但是为了调整后面的空白，需要再次插入的情况会很多。

用户在字段值的前后留下空白时，后边的空白会删除，如下。

```
" aaa \t" -> " aaa"
```

分隔记录格式时

存在于字段值前后的空白将视为多余而被删除。但是，用户如果要在字段的前后插入空白，则用ENCLOSED BY字符串括起来后插入空白，就可将空白视为字段的数据。

用户在字段值的前后留下空白时，前后的空白都会如下被删除。

```
" aaa \t" -> "aaa"
```

5.6.3. 将字段值的空白视为数据时

虽然用前面所提及的方式删除了数据，但kdLoader Utility可以利用「5.9.6. PRESERVE BLANKS语句」强行将字段前后存在的空白用数据值插入。

5.7. 指定命令提示符的参数

本节中将说明在命令提示符当中可以指定的kdLoader Utility参数。

在运行kdLoader Utility时，若不指定参数，则会如下显示命令提示符当中可以指定的参数目录和使用方法。

```
$ kdloader
kdLoader 11
Inspur Corporation Copyright (c) 2008-. All rights reserved.

Usage: kdloader [options] [controls]

Options:
  -h|--help        Displays the more detailed information.
  -c|--charset     Display usable character sets in this version.
  -v|--version     Displays the version information.

Controls:
  bad              Bad file name
  bindsize         Bind buffer size
  control          Control file name
  data             Data file name
  direct           Direct Path Load [default:N]
  dpl_log          Enable Direct Path Load logging [default:N]
  dpl_parallel     Use Parallel Direct Path Loading [default:N]
  errors           Errors to allow [default:50]
  log              Log file name
  message          Loading progress message to stdout
  multithread      Use multi-thread for Direct Path Loading [default:Y]
  readsize         Read buffer size
  rows             Rows per commit
  skip             Skip lines in data file [default:0]
  skip_idx_errors  Skip index errors [default:N]
  userid           userid/passwd@dbname

Example:
  kdloader userid=userid/passwd@dbname control=sample.ctl bindsize=1000000
```

用户在命令提示符里可以输入参数或控制文件里所需的源数据。但是USERID、CONTROL、DIRECT、MESSAG、READSIZE、BINDSIZE、ERRORS、ROWS参数值能在命令提示符中指定。

Options的-c表示用于下级兼容性的当前client版本当中可使用的character sets信息。

5.7.1. 参数目录

下面是命令提示符当中可以指定的kdLoader Utility参数的示例。

[例 5.2] 命令提示符当中可以指定的**kdLoader Utility**参数

```
userid=userid/passwd@dbname

control=/home/test/control.ctl
control=../control.ctl
```

```
log=/home/test/control.log
log=../control.log

data=/home/test/data.dat
data=../data.dat

bad=/home/test/data.bad
bad=../data.bad

skip=1
direct=Y
dpl_log=Y
message=10000
readsize=65536
bindsize=65536
errors=100
rows=100
multithread=N
dpl_parallel=Y
```

项目	说明
Options	– -c：显示可兼容下级的当前客户端版本中可以使用的字符集信息。
userid	是指定K-DB的数据库用户名和密码及数据库名的参数。指定为userid=userid/passwd@databasename格式。
control	是指定包含参数信息的控制文件路径和名称的参数。 可以使用绝对路径与当前目录的相对路径方式。
log	是指定记录数据加载过程当中发生的日志路径与名称的参数。（默认值：控制文件名.log） 可以使用绝对路径与当前路径的相对路径方式。 用户若在命令提示符与控制文件当中指定了所有路径，则首在命令提示符里指定的值优先。
data	是指定包含实际数据的文件路径与名称的参数。 可以使用决定路径和当前目录的相对路径方式。 用户若在命令提示符与控制文件里指定了所有路径，则在命令提示符里指定的值优先。
bad	对于数据加载失败记录文件指定路径和名称的参数。（默认值：控制文件名.log）

项目	说明
	可以使用绝对路径和当前目录的相对路径方式。 用户若在命令提示符与控制文件当中指定了所有路径，则在命令提示符当中指定的值优先。
skip	以该数据文件开始指定的行数来从加载的目标中剔除的参数。（默认值：0） 功能与控制文件选项中的「5.9.16. IGNORE LINES语句」相同。
direct	是用户加载数据时，指定Conventional Path Load与Direct Path Load方法中的一个参数。 – Y：指定为Direct Path Load。输入其他值或空值，则以Conventional Path Load方式加载。 – N：以Conventional Path Load方式加载数据。（默认值）
dpl_log	以Direct Path Load方式加载数据时，指定是否在服务器的日志文件里保留日志的参数。 – Y：上传数据时，在服务器的日志文件当中保留日志。发生障碍时可以还原，但是加载时会性能低下。 – N：上传数据时，在服务器的日志文件当中不保留日志。发生障碍时，不能还原。（默认值）
message	是在界面里输出kdLoader Utility当前处理的逻辑性记录个数的参数。 不另外指明时，不在界面里输出进行状态。但是若指明太小的值，则不会影响到性能。
readsize	kdLoader使用缓冲区读取数据文件的内容。是此时使用的读取缓冲区大小的参数。 （单位是byte，最大值为2,097,152（2M byte）。）
bindsize	是以Direct Path Load方式加载数据时，指定客户使用的Direct Path Stream的大小的参数。 K-DB客户以数据指定的大小被绑定以前，不会加载到服务器里。因此加载大容量数据时可以有效使用。（单位是byte，最大值为15,728,640（15M byte）。）
errors	指定加载数据时允许的最大错误个数的参数。 （默认值：50） kdLoader Utility在不超过用户指定的错误个数的范围里加载数据。如果超出指定个数，则停止数据的加载。

项目	说明
	– ERRORS是从-1和0开始的最大整数值（2147483647）中的某个值。 – ERRORS设为0表示不允许任何一个错误。 – ERRORS设为正整数N时，表示发生N个错误后，停止加载数据。即，只允许N-1个错误。 – ERRORS设为-1时，表示跳过所有错误，指加载没有错误的内容。
rows	是用户在加载大容量数据时，指定要进行提交的记录个数的参数。但是考虑到kdLoader的性能，不会正确按所指定的记录个数向服务器发送数据。
multithread	是用Direct Path Load方法加载数据时，指定是否指定多线程的参数。 kdLoader Utility使用两个线程，一个线程从用户的数据文件里读取数据，加载在Direct Path Load的stream缓冲区，其余的线程将所加载的Stream缓冲区加载到服务器里。 持有多个CPU的机器或客户与服务器存在于其他机器时，可以期待性能的提高。 – Y：另外存在从文件里读取数据的线程和向服务器加载相应数据的线程。（默认值） – N：一个线程从文件里读取数据，向服务器里加载该数据。
dpl_parallel	使用Direct Path Load方法加载数据时，决定是否使用Parallel Loading传输数据的参数。 – Y：使用Parallel Direct Path Load。 – N：用默认Direct Path Load加载。

5.8. 高级功能

本章中将通过kdLoader使用程序对附加的高级功能进行说明。

5.8.1. Parallel DPL

使用Direct Path Load方式传输时，当对象表被锁定，则无法使用DPL同时加载相同的表。为了解决这个问题开发了新的方法，即Parallel DPL。

使用Parallel DPL可以在想要加载的数据中设置Parallel DPL flag，服务器并发接收数据，并在执行合并操作后进行存储。在使用Parallel DPL之前，必须要将kdloader中的dpl和dpl_parallel设为'Y'。

下面是使用Parallel DPL的示例。

[例 5.3] 使用**Parallel DPL**运行**kdLoader** 的示例

```
$ kdloader userid=db_user/db_password@default
control=sample.ctl data=sample.data direct=Y dpl_parallel=Y
```

5.8.2. 加密连接信息的功能

在kdLoader实用程序中还提供了利用加密文件（wallet）加密数据库连接信息（connect_string）的功能。

在使用加密连接信息功能之前，首先应创建存储连接信息的加密文件（wallet）。有关加密文件的详细说明请参考「1.6.32. SAVE CREDENTIAL」。

下面是利用加密文件进行连接的示例。

[例 5.4] 利用加密文件（**wallet**）运行**kdLoader** 的示例

```
$ export LR_WALLET_PATH=./wallet_file.dat

$ kdloader control=sample.ctl data=sample.data
```

参考

该功能只在UNIX系统中支持，并且需要安装Open SSL。

5.9. 指定控制文件的选项

本节将说明指定控制文件选项的方法。

用户在文件里可以指定下面的信息。

- 数据文件里包含的字符集合
- 包含数据的数据文件
- 记录加载中发生日志的日志文件
- 记录加载失败的数据错误文件
- 存在于表中的原有数据的处理方法（APPEND|REPLACE|TRUNCATE|MERGE）
- 字段结束符与其他选项（TERMINATOR、ENCLOSED BY STRING、ESCAPED BY STRING）
- 行的开始字符串和结束字符串
- 数据文件里要被忽略的行的个数
- 表的特定列的相关选项

控制文件的格式如下。必须按控制文件里所指定的下面格式的顺序使用，大括号（[]）里包含的内容可以省略。

```
LOAD DATA
  [CHARACTERSET characterset_name]
  [INFILE data_file_name]
  [LOGFILE log_file_name]
  [BADFILE bad_file_name]
  [APPEND|REPLACE|TRUNCATE|MERGE(column_name, .....)]
  [PRESERVE BLANKS]
  INTO TABLE table_name
  [MULTI INSERT INDEXES|FAST BUILD INDEXES]
  [FIELDS [TERMINATED BY field_terminator]
          [OPTIONALLY ENCLOSED BY enclosed_by_start_string
                              [AND enclosed_by_end_string]]
          [ESCAPED BY escaped_by_string]]
  [LINES  [FIX number]
          [STARTED BY line_start_string]
          [TERMINATED BY line_terminator_string]]
  [TRAILING NULLCOLS]
  [IGNORE number LINES]
  (column_name [POSITION(from:to)]
               [INTEGER EXTERNAL(size)|FLOAT EXTERNAL(size)|DOUBLE EXTERNAL(size)|
                   CHAR(SIZE)|RAW(SIZE)|DATE(size) date_fmt_string|
                   TIMESTAMP(size) timestamp_fmt_string|TIME(size) time_fmt_string]
               [OUTFILE]
               [CONSTANT constant_value]
               [NULL TERMINATED]
               [PRESERVE BLANKS]
               [sql_expression], ..........)
  -- This line is comment.
```

5.9.1. CHARACTERSET 语句

可以在K-DB服务器中加载多种字符集的数据。指定的值对控制文件和数据文件的字符集产生影响。不做指定时，将指定客户端的字符集，即客户端的环境变量或dsn文件的KD_NLS_LANG

指定字符集的详细内容如下。

- 语法

```
CHARACTERSET characterset_name
```

参数值	说明
characterset_name	指定为可以作为KD_NLS_LANG （默认值：环境变量或dsn文件的KD_NLS_LANG值）

- 示例

  当前客户端的字符集为KSC5601(KD_NLS_LANG =EUCKR)，欲将MSWIN949数据文件加载到服务器时，在控制文件中指定如下内容。

```
CHARACTERSET MSWIN949
```

5.9.2. INFILE语句

指定实际包含数据的文本文件即数据文件的路径和名称。用户可以使用绝对路径与当前目录的相对路径。

用户若在命令提示符和控制文件当中指定了所有路径时，则在命令提示符里指定的值优先。

指定数据文件

- 语法

```
INFILE data_file_name
```

参数值	说明
data_file_name	指定数据文件的路径和名称。

- 示例

```
INFILE '/home/test/data.dat'
INFILE '../data.dat'
```

5.9.3. LOGFILE语句

指定记录加载数据的过程中所发生日志的日志文件的路径和名称。用户可以使用绝对路径和当前目录的相对路径方式。

用户若在命令提示符和控制文件里指定了所有路径，则在命令提示符里指定的值优先。

指定日志文件的详细内容如下。

- 语法

```
LOGFILE log_file_name
```

参数值	说明
log_file_name	指定日志文件的路径和名称。（默认值：控制文件名.log）

- 示例

```
LOGFILE '/home/test/control.log'
LOGFILE '../control.log'
```

5.9.4. BADFILE语句

指定记录数据加载失败记录的错误文件的路径和名称。用户可以使用绝对路径和当前目录的相对路径方式。

用户若在命令提示符和控制文件里指定了所有路径，则在命令提示符里指定的值优先。

欲使用DPL进行加载时，如果服务器中发生错误，那么不记录失败的记录。原因是性能有限的DPL方式无法掌握失败的row。

指定错误文件的详细内容如下。

- 语法

```
BADFILE bad_file_name
```

参数值	说明
bad_file_name	指定错误文件的路径和名称。（默认值：数据文件名.bad）

- 示例

```
BADFILE '/home/test/data.bad'
BADFILE '../data.bad'
```

5.9.5. 现有数据的处理方法

在用户指定的表里存在现有数据时，可以指定该数据的方法。

为REPLACE和TRUNCATE选项时，表的记录被删除后会自动提交，因此kdLoader Utility运行以后不能恢复现有的数据。

指定现有数据的处理方法的详细内容如下。

- 语法

```
APPEND|REPLACE|TRUNCATE|MERGE(column_name, .....)
```

参数值	说明
APPEND	保留现有数据，并添加新的数据。（默认值）
REPLACE	用DELETE命令删除当前数据后添加新的数据。 用户必须持有DELETE权限。存在有关DELETE的触发器时，该触发器会被运行。
TRUNCATE	用TRUNCATE语句删除现有的数据，添加新的数据。 但是，该表里若有参考完整性约束条件时，必须暂停条件之后运行TRUNCATE语句。
MERGE(column_name, ...)	指明要MERGE的列目录。 Utility通过关键值使用用户指定的列的目录。新数据与现有数据的关键值相同时，现有数据UPDATE为新数据，否则输入相关数据。

- 示例

```
control.ctl:
  LOAD DATA
  ...
  REPLACE
  ...
  (...)
```

```
control.ctl:
  LOAD DATA
  ...
  MERGE(EMPNO, EMPNM)
  ...
  (...)
```

5.9.6. PRESERVE BLANKS语句

不删除字段的数据所包含的空白，直接在数据库里输入时使用。

- 示例

```
control.ctl:
  LOAD DATA
```

```
...
PRESERVE BLANKS
...
(...)
```

5.9.7. 表的指定方法

指定要输入数据文件内容的对象表的名称。

指定表的详细内容如下。

- 语法

```
INTO TABLE table_name
```

参数值	说明
table_name	指定对象表的名称。可设为PUBLIC synonym的表名。

- 示例

```
control.ctl:
  LOAD DATA
  ...
  INTO TABLE EMP
  ...
  (...)
```

5.9.8. 索引生成方法

以Direct Path Load方式在表里加载数据时，指定表里存在的索引的生成方法。可以在MULTI INSERT方式与FAST BUILD方式中任选一个。

- **MULTI INSERT INDEXES**

  是将索引以多个记录单位优化后一次性生成的方式，

- **FAST BUILD INDEXES**

  是忽略现有索引加载数据文件的数据后，重新生成的方式。

在使用kdLoader Utility以前若在该表里存在很多现有数据，则用MULTI INSERT方式生成索引比较有利，其余的情况，用FAST BUILD方式更有利。

使用 kdLoader实用程序执行DPL时，数据重复等问题可能会造成索引变成Unusable状态。

指定索引创建方法的详细内容如下。

- 语法

```
[MULTI INSERT INDEXES|FAST BUILD INDEXES]
```

参数值	说明
MULTI INSERT INDEXES	用MULTI INSERT方式生成索引。
FAST BUILD INDEXES	用FAST BUILD方式生成索引。

- 示例

```
control.ctl:
  LOAD DATA
  ...
  MULTI INSERT INDEXES
  ...
  (...)
```

5.9.9. FIELDS语句的TERMINATED BY语句

从数据文件里读取记录时，读取后用一个字段处理，直到发现指定为FIELD TERMINATOR的字符。

指定字段结束符的详细内容如下。

- 语法

```
FIELDS TERMINATED BY field_terminator
```

参数值	说明
field_terminator	指定为ASCII字符串

- 示例

```
control.ctl:
  LOAD DATA
  ...
  FIELDS TERMINATED BY ','
  ...
  (...)
```

5.9.10. FIELDS语句的OPTIONALLY ENCLOSED BY语句

从数据文件里读取记录时，指定包含字段的值开端和尾端的字符串。

在字段值里若包含空字符（' '、'\t'、'\r'、'\n'）或字段结束符、行结束符等元字符串时，kdLoader Utility会将该字符串视为数据值。但是ESCAPED BY字符串即使通过ENCLOSED BY字符串被包含，也会被视为元字符串。

需要注意的是若指定为ENCLOSED BY的字符串等字符串被字段值括起时，指定为ESCAPED BY的字符串必须使用前缀，才能以字段值解释。若被指定为ESCAPED BY的字符串没有使用为前缀，则该字符串即使用ENCLOSED BY字符解释，也不能识别字段值的剩余部分。

指定ENCLOSED BY字符串的详细内容如下。

- 语法

```
FIELDS OPTIONALLY ENCLOSED BY enclosed_by_start_string
```

参数值	说明
enclosed_by_start_string	指定为ASCII字符串。

- 示例

包含字段的开始字符串与结束字符串相同时，用下面的格式指定。

```
control.ctl:
  LOAD DATA
  ...
  FIELDS OPTIONALLY ENCLOSED BY '"'
  ...
  (...)
```

包含字段的开始字符串与结束字符串不同时，用下面的格式指定。

```
control.ctl:
  LOAD DATA
  ...
  FIELDS OPTIONALLY ENCLOSED BY '{$' AND '$}'
  ...
  (...)
```

最后若下面的参数被指定时，则用下面的格式指定。

```
control.ctl:
  LOAD DATA
  ...
```

```
FIELDS OPTIONALLY ENCLOSED BY '"'
       ESCAPED BY '\\'
...
(...)
```

作为字段值指定为ENCLOSED BY字符串等的字符串被括起时，若指定为ESCAPED BY的字符串不使用为前缀，就不能正确识别字段的值。

```
”quotation mark[\”]”,0001→(quotation mark["],0001)
”quotation mark[”]”,0001→发生错误
```

5.9.11. FIELDS语句的ESCAPED BY语句

若遇见由ESCAPED BY指定的字符串，则扩展后面的字符的意义，可以读取特殊字符或字符串。需要注意的是，为了指定'\'，必须使用2个'\\'。

指定ESCAPED BY字符串的详细内容如下。

- 语法

```
ESCAPED BY escaped_by_string
```

参数值	说明
escaped_by_string	指定为ASCII字符串。

- 示例

```
ESCAPED BY '\\'
ESCAPED BY '$$!'
```

5.9.12. LINES语句的FIX语句

可以在数据文件中指定一个行的长度。不能使用FIELDS语句、LINE TERMINATED BY和LINE STARTED BY语句。

指定一个行的长度的详细内容如下。

- 语法

```
LINES FIX number
```

参数值	说明
number	设置为整数值。单位为byte。

- 示例

```
control.ctl:
  LOAD DATA
  ...
  LINES FIX 5
  ...
  (...)
```

如上指定参数，假设读取保存下面内容的数据文件。

```
data.dat:
abcdefghijklmnopqrst
```

kdLoader Utility读取上面的数据文件，解释如下。

```
LINE #1: abcde
LINE #2: fghij
LINE #3: klmno
LINE #4: pqrst
```

5.9.13. LINES语句的STARTED BY语句

从数据文件中以行为单位读取时，只加载前缀后面的数据。当遇到不包含前缀的行时，将从加载对象中排除该行。为提高性能通常反复使用相同字符作为前缀。

指定行的开始字符串的详细内容如下。

- 语法

```
LINES STARTED BY line_start_string
```

参数值	说明
line_start_string	指定为ASCII字符串（最大30bytes）。

- 示例

```
control.ctl:
  LOAD DATA
  ...
```

```
  LINES STARTED BY '$$$'
  ...
  (...)
```

如上指定参数，假设读取保存下面内容的数据文件。

```
data.dat:
$$$0001,”SMITH” … something … $$$0002,”DAVID”
```

kdLoader Utility读取上面的数据文件，解释如下。

```
(0001, “SMITH”), (0002, “DAVID”)
```

5.9.14. LINES语句的TERMINATED BY语句

从数据文件里读取数据时，若发现了用LINE TERMINATOR指定的字符串，则被处理为完成一个记录。

指定行结束字符的详细内容如下。

- 语法

```
LINES TERMINATED BY line_terminator_string
```

参数值	说明
line_terminator_string	指定为ASCII字符串。（默认值：'\n'）

- 示例

下面是将LINE TERMINATOR字符串指定为由2个ASCII字符构成的'|\n'的示例。

windows环境下的数据文件中应输入'|\r\n'。

```
control.ctl:
  LOAD DATA
  ...
  LINES TERMINATED BY '|\n'
  ...
  (...)
```

5.9.15. TRAILING NULLCOLS语句

数据文件的记录里没有的最后一个列的数据将被视为NULL。若不添加该子句，则在记录里没有的NULL列的数据将视为错误。

TRAILING NULLCOLS语句的详细内容如下。

- 语法

```
TRAILING NULLCOLS
```

- 示例

下面是将第一个记录的job 列值设置为NULL的示例。

```
control.ctl:
  LOAD DATA
  ...
  TRAILING NULLCOLS
  ...
  (
      id,
      name,
      job
  )
```

```
data.dat:
    10 SMITH
```

5.9.16. IGNORE LINES语句

从加载对象中删除从数据文件的开始到所指定的数的行。

指定加载对象的详细内容如下。

- 语法

```
IGNORE number LINES
```

参数值	说明
number	指定为整数值（默认值：0）

- 示例

数据文件的第一个行里若显示列的名称，则指定参数如下，并只删除第一个行。

```
control.ctl:
  LOAD DATA
  ...
  IGNORE 1 LINES
```

```
...
(...)
```

5.9.17. 对象列与属性

用户将指明要输入数据表的列目录。但是必须要与数据文件的列顺序一样编辑。

列属性请参考各节的说明。

- 「5.9.17.1. 列的POSITION语句」
- 「5.9.17.2. 数据类型」
- 「5.9.17.3. 数据缓冲区大小」
- 「5.9.17.4. 列的OUTFILE语句」
- 「5.9.17.5. 列的CONSTANT语句」
- 「5.9.17.6. 保留列的空白」
- 「5.9.17.7. 列的SQL表达」

指定对象列和属性的详细内容如下。

- 语法

```
(column_name [POSITION(from:to)]
             [INTEGER EXTERNAL(size)|FLOAT EXTERNAL(size)|
                 DOUBLE EXTERNAL(size)|
                 CHAR(SIZE)|RAW(SIZE)|DATE(size) date_fmt_string|
                 TIMESTAMP(size) timestamp_fmt_string|
                 TIME(size) time_fmt_string]
             [OUTFILE]
             [CONSTANT constant_value]
             [NULL TERMINATED]
             [PRESERVE BLANKS]
             [sql_expression], ..........)
```

参数值	说明
column_name	指明对象表的列名称。

5.9.17.1. 列的POSITION语句

在数据文件的一个行中指定与该列对应的值的开始位置和结束位置。

POSITION语句中进行指定的详细内容如下。

- 语法

```
POSITION(from:to)
```

参数值	说明
from	在数据文件的一个行中指定该列的开始位置。行的位置从1开始。
to	在数据文件的一个行中指定该列的结束位置。

- 示例

若是固定的记录格式，则如下与列的目录一起指明正确的位置。

```
control.ctl:
  LOAD DATA
  ...
  (
      empno   position(01:04),
      ename   position(06:15),
      job     position(17:25),
      mgr     position(27:30),
      sal     position(32:39),
      comm    position(41:48),
      deptno  position(50:51)
  )
```

若是分隔的记录格式，则不必必需使用POSITION语句。

```
control.ctl:
  LOAD DATA
  ...
  (
      empno,
      ename,
      job,
      mgr,
      sal,
      comm,
      deptno
  )
```

5.9.17.2. 数据类型

kdLoader Utility提供特定数据类型。 按数据类型，默认值的不同，或以不同的方式绑定列的数据。

Utility提供的数据类型如下。

- 数字数据类型
- 字符串数据类型
- 二进制数据类型
- 日期数据类型

数字数据类型

为了在K-DB服务器的数字型列里加载字符串数据而使用。在列里若指明NULL，则kdLoader Utility用默认值绑定0，加载在服务器里。相反，若不绑定0而绑定NULL，则只要声明字符串数据类型来使用即可。

指定数字数据类型（NUMERIC EXTERNAL）的列格式的详细内容如下。

- 语法

```
INTEGER EXTERNAL(size)|FLOAT EXTERNAL(size)|DOUBLE EXTERNAL(size)
```

参数值	说明
INTEGER EXTERNAL(size)	字符串数据为整数时使用。
FLOAT EXTERNAL(size)	字符串数据为实数时使用。
DOUBLE EXTERNAL(size)	字符串数据为两倍精度（double precision）实数形式时使用。

参考

size指定「5.9.17.3. 数据缓冲区大小」。

字符串数据类型

为了在K-DB服务器的字符型（CHAR、VARCHAR、CLOB、NCLOB）列里加载字符串数据而使用。若在列里指明NULL，则kdLoader Utility以默认值绑定NULL并在服务器里加载。相反，若不绑定为NULL而是绑定NULL，则只要声明字符串数据类型来使用即可。

指定字符串数据类型的列格式的详细内容如下。

- 语法

```
CHAR(size)
```

参数值	说明
CHAR(size)	在字符串数据里使用。size是「5.9.17.3. 数据缓冲区大小」。

二进制数据类型

为了在K-DB服务器的大对象型（RAW、BLOB、LONGRAW）列里加载二进制数据而使用。若在列指明NULL，则kdLoader Utility以默认值绑定NULL并在服务器里加载。

指定二进制数据类型的列格式的详细内容如下。

- 语法

```
RAW(size)
```

参数值	说明
RAW(size)	在二进制数据里使用。size是「5.9.17.3. 数据缓冲区大小」。

日期数据类型

为了在K-DB服务器的日期型（DATE、TIME、TIMESTAMP）列里加载字符串数据而使用。在列里若指明NULL，则kdLoader Utility用默认值绑定NULL并在服务器里加载。

K-DB的客户用KD_NLS_DATE_FORMAT, KD_NLS_TIMESTAMP_FORMAT指定DATE和TIMESTAMP类型的列格式。但是，用户在kdLoader Utility的控制文件里也可以直接用DATE、TIMESTAMP、TIME类型的列格式指定。但是必须用双引号（""）括起。

指定日期数据类型的列格式的详细内容如下。

- 语法

```
DATE date_fmt_string|TIMESTAMP timestamp_fmt_string|TIME time_fmt_string
```

参数值	说明
date_fmt_string	指定DATE类型的列格式。
timestamp_fmt_string	指定TIMESTAMP类型的列格式。.
time_fmt_string	指定TIME类型的列格式。

- 示例

下面是kdLoader实用程序的控制文件。本示例中按照data.dat文件内部的列hiredate的'YYYYMMDD'格式来设置日志。

```
control.ctl:
  LOAD DATA
  .....
  INFILE 'data.dat'
  .....
```

```
  (
      empno,
      ename,
      hiredate DATE "YYYYMMDD"
  )
```

```
data.dat:
1111, JOHN, 19981112
2222, TOM, 20070802
```

5.9.17.3. 数据缓冲区大小

kdLoader Utility度与数据文件的数据保存在数据缓冲器里，将该数据转换为K-DB服务器的数据类型后加载。

kdLoader Utility为了读取基本分隔的记录格式的数据，利用数据库里的Schema信息分配适当大小的缓冲器。相反为了读取固定记录格式的数据，利用POSITION的开始与末端的信息，分配缓冲器。

若用户准确知道数据的长度信息，则可以指明数据缓冲器的大小。此外，为了读取BLOB、CLOB、LONG、LONG RAW等大容量数据，可以指明本人所希望的数据缓冲器的大小（单位是byte）。

- 示例

  控制文件的格式如下，empno和hiredate列中分别分配5bytes、10bytes的数据缓冲区。

```
control.ctl:
  LOAD DATA
  ....
  (
      empno INTEGER EXTERNAL(5),
      hiredate DATE(10) "YYYY/MM/DD"
  )
```

5.9.17.4. 列的OUTFILE语句

用户在读取BLOB、CLOB、LONG、LONG RAW类型大容量数据或RAW等二进制数据时，可以指定**OUTFILE**属性，使用户从不是数据文件的其他文件里读取。但是用户必须得在数据文件里指明该文件的路径。

用户若指定OUTFILE属性，kdLoader Utility缓冲内部文件的数据，多次加载到服务器里。相反，若不指定OUTFILE属性，则kdLoader Utility可以直接从数据文件里读取大容量数据，为此基本分配32Kbyte的数据缓冲器。因此若要从数据文件里读取比这更大的数据，直接指定「5.9.17.3. 数据缓冲区大小」即可。

- 示例

  若控制文件的信息如下，

```
control.ctl:
  LOAD DATA
  .....
  INFILE 'data.dat'
  .....
  (
      empno,
      ename,
      resume OUTFILE
  )
```

在data.dat内部resumelieder位置上输入如下文件路径，就可以从该文件中读取数。

```
data.dat
1111, JOHN, ./clobdata.txt
2222, TOM, /home/test/clobdata.txt
```

5.9.17.5. 列的CONSTANT语句

利用**CONSTANT**属性可以将常数值设置为指定列的值。由于常量中可能包含空白字符，因此必须要用双引号括住该值(" ")。

- 示例

  控制文件格式如下时，不受data.dat文件内部数据值的影响，列empno值设置为数字1234。

```
control.ctl:
  LOAD DATA
  .....
  INFILE 'data.dat'
  .....
  (
      empno constant "1234",
      ename
  )
```

5.9.17.6. 保留列的空白

虽然功能与「5.9.6. PRESERVE BLANKS语句」相同，但只能在对应列中应用该列的属性。

- 示例

  控制文件格式如下时，列empno对应的第一个字段的所有空白字符都将保留为数据值。

```
control.ctl:
  LOAD DATA
  ...
  (
```

```
    empno PRESERVE BLANKS,
    ename
)
```

5.9.17.7. 列的SQL表达

可使用SQL的表达来表现特定列的值。SQL表达必需要使用双引号(" ")。在SQL表达中使用**:**号和列名称可以绑定列的数据值。

- 示例

  控制文件格式如下时，利用列empno中的TO_CHAR函数，把SYSDATE值设置为'YYYYMMDD'格式的字符串。在列ename中将数据文件的列empno字段值绑定为数据值。

```
control.ctl:
  LOAD DATA
  ...
  (
      empno "TO_CHAR(SYSDATE, 'YYYYMMDD')",
      ename ":empno"
  )
```

5.9.18. 插入注释

数据文件的相关行用注释处理。

- 示例

```
-- This is comment for control file.
```

5.10. 运行示例

本节将对被分割的记录格式、固定的记录格式、BLOB与CLOB类型等大容量数据存在的三种情况按下面的顺序加载数据的示例进行介绍。

1. 生成表。(在所有示例中的共同事项。）
2. 编辑控制文件。
3. 编辑数据文件。
4. 运行kdLoader。
5. 确认日志文件与错误文件。

下面是生成要共同使用的表的示例。在本示例中假设数据库名称为default，表的持有者为loader/loader_pw。

```
CREATE TABLE MEMBER
(
    ID        NUMBER(4) NOT NULL,
    NAME      VARCHAR2(10),
    JOB       VARCHAR2(9),
    BIRTHDATE DATE,
    CITY      VARCHAR2(10),
    PICTURE   BLOB,
    AGE       NUMBER(3),
    RESUME    CLOB
);

CREATE TABLE CLUB
(
    ID       NUMBER(6) NOT NULL,
    NAME     VARCHAR2(10),
    MASTERID NUMBER(4)
);
```

5.10.1. 分隔记录格式

利用Conventional Path Load方法将被分隔的记录格式的数据加载到K-DB服务器里。

控制文件的编辑

控制文件sample1.ctl的内容如下。

```
LOAD DATA
INFILE './data.dat'
APPEND
INTO TABLE club
FIELDS TERMINATED BY ','
       OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '|\n'
IGNORE 1 LINES
(
 id integer external,
 name,
 masterid integer external
)
```

下面是在控制文件里指定的信息。

- 数据文件

  是位于当前目录里的data.dat文件。

- 日志文件

  用户若在命令提示符里不指明日志文件名，则默认的控制文件名生成为其他control.log文件。

- 错误文件

  用户若在命令提示符里不指明错误文件名，则基本的数据文件名生成为其他data.bad文件。但是，若错误记录不存在，则不生成错误文件。

- 对象表

  加载用户club 表的id、name、masterid列的数据。

- FIELDS TERMINATED BY字符串、FIELDS OPTIONALLY ENCLOSED BY字符串、FIELDS ESCAPED BY字符串、LINES TERMINATED BY字符串

  FIELDS TERMINATED BY字符使用逗号(,)，FIELDS OPTIONALLY ENCLOSED BY字符使用双引号(")，LINES TERMINATED BY字符串使用'|\n'。FIELDS ESCAPED BY字符未作指定，因此默认使用'\\'。

- 加载对象

  由于存在IGNORE LINES语句，因此忽略数据文件的第一行记录。

数据文件的编辑

数据文件data.dat的内容如下。

```
id     name      masterid|
111111,FC-SNIFER,2345|
dkkkkkkkkkk|
111112,"DOCTOR CLUBE ZZANG",2222|
111113,"ARTLOVE",3333|
111114,FINANCE,1235|
111115,"DANCE MANIA",2456|
111116,"MUHANZILZU",2378|
111117,"INT'L",5555
```

kdLoader实用程序的运行

如下运行kdLoader实用程序。

```
kdloader userid=loader/loader_pw@default control=./control.ctl
```

确认日志文件与错误文件

确认运行kdLoader Utility以后生成的日志文件与错误文件。

记录kdLoader Utility的运行过程的日志文件的内容如下。

```
kdLoader 11
Inspur Corporation Copyright (c) 2008-. All rights reserved.

Data File : ./data.dat
Bad File : ./data.bad

Table 'CLUB' was loaded from the data file.

Column Name                     Position DataType
------------------------------ ---------- ------------
ID                                    1 NUMERIC EXTERNAL
NAME                                  2 CHARACTER
MASTERID                              3 NUMERIC EXTERNAL

Record 2 was rejected.
  TBR-80053 : Some relatively positioned columns are not present in the record.

Record 3 was rejected.
  TBR-80025 : Column data exceeds data buffer size.

Record 6 was rejected.
  TBR-80025 : Column data exceeds data buffer size.

Table 'CLUB'
----------------
8 Rows were requested to load.
5 Rows were loaded successfully.
3 Rows were failed to load because of some errors

Elapsed time was: 00:00:00.407089
```

kdLoader Utility记录加载失败的错误数据的错误文件如下。用户修改错误发生的数据，重新在K-DB的数据库里上载。

```
dkkkkkkkkkk|
111112,"DOCTOR CLUBE ZZANG",2222|
111115,"DANCE MANIA",2456|
```

5.10.2. 固定记录格式 - 记录分隔符为 EOL字符的情况

利用Direct Path Load方法将固定记录格式的数据加载在K-DB服务器中。

控制文件的编辑

控制文件control.ctl的内容如下。

```
LOAD DATA
INFILE '/home/test/data.dat'
APPEND
INTO TABLE MEMBER
(
    id        position (01:04) integer external,
    name      position (06:15),
    job       position (17:25),
    birthdate position (27:36),
    city      position (38:47),
    age       position (49:51) integer external
)
```

下面是在控制文件里指定的信息。

- 数据文件

  /home/test目录下的data.dat文件。

- 日志文件

  用户若在命令提示符里不指明日志文件名，则默认控制文件名将会生成其他control.log的日志文件。

- 错误文件

  用户若在命令提示符当中不指明错误文件名，则默认数据文件名将会生成为其他data.bad的错误文件。但是若不存在错误记录，则不生成错误文件。

- 目标表

  用户要加载member表的id、name、job、birthdate、city、age 列的数据。

- FIELDS TERMINATED BY字符串、FIELDS OPTIONALLY ENCLOSED BY字符串、FIELDS ESCAPED BY字符串、LINES TERMINATED BY字符串

  FIELDS TERMINATED BY、FIELDS OPTIONALLY ENCLOSED BY、FIELDS ESCAPED BY字符串由于是固定记录格式的数据，因此不使用。由于不使用LINES FIX语句，使用LINES TERMINATED BY语句来执行EOL。

- 加载对象

  从数据文件的第一行开始加载数据。

数据文件的编辑

数据文件data.dat的内容如下。由于kdLoader Utility接收固定记录格式的数据来输入，用户必须在正确的位置上输入列数据。

```
7777 KKS        CHAIRMAN  1975-11-18 SEOUL      33
7839 KING       PRESIDEN             DAEGU      45
7934 MILLER     CLERK     1967-01-24 BUSAN      37
7566 JONES      MANAGER
7499 ALLEN      SALESMAN  ddddddddd  KYUNG-JU
aaaa7654 MARTIN SALESMAN
7648 CHAN       ANALYST   1979-10-11 INCHON     28
```

kdLoader Utility的运行

如下运行kdLoader实用程序。

```
kdloader userid=loader/loader_pw@default control=./control.ctl direct=Y
```

确认日志文件与错误文件

确认运行kdLoader Utility之后生成的日志文件与错误文件。

记录kdLoader Utility运行过程的日志文件内容如下。

```
kdLoader 11
Inspur Corporation Copyright (c) 2008-. All rights reserved.

Data File : ./data.dat
Bad File : ./data.bad

Table 'MEMBER' was loaded from the data file.

Column Name                     Position DataType
------------------------------ ---------- ------------
ID                                    1 NUMERIC EXTERNAL
NAME                                  2 CHARACTER
JOB                                   3 CHARACTER
BIRTHDATE                             4 DATE
CITY                                  5 CHARACTER
AGE                                   6 NUMERIC EXTERNAL

Record 4 was rejected.
  TBR-80053 : Some relatively positioned columns are not present in the record.

Record 5 was rejected.
```

```
  TBR-80053 : Some relatively positioned columns are not present in the record.

Record 6 was rejected.
  TBR-80053 : Some relatively positioned columns are not present in the record.

Table 'MEMBER'
---------------
7 Rows were requested to load.
4 Rows were loaded successfully.
3 Rows were failed to load because of some errors

Elapsed time was: 00:00:00.338089
```

日志文件的内容如上所示，可以确认没有加载的3个记录当中不存在最后一个列值。利用「5.9.15. TRAILING NULLCOLS语句」子句，可以用NULL字符绑定该记录的最后一个列值。

kdLoader Utility记录加载失败的错误数据文件如下。发生错误的数必须修改后在K-DB的数据库里重新上载。

```
7566 JONES      MANAGER
7499 ALLEN      SALESMAN  ddddddddd  KYUNG-JU
aaaa7654 MARTIN SALESMAN
```

5.10.3. 固定记录格式 - 固定长度记录情况

利用Conventional Path Load方法，将既有固定长度的记录格式数据加载在K-DB服务器里。

控制文件的编辑

控制文件control.ctl的内容如下。

```
LOAD DATA
INFILE './data.dat'
APPEND
INTO TABLE MEMBER
LINES FIX 51
TRAILING NULLCOLS
(
    id        position (01:04) integer external,
    name      position (06:15),
    job       position (17:25),
    birthdate position (27:36),
    city      position (38:47),
```

```
    age       position (49:50) integer external
)
```

下面是在控制文件当中指定的信息。

- 数据文件

  位于运行kdloader命令当前目录的data.dat文件。

- 日志文件

  用户若在命令提示符当中没有指明日志文件名，则默认控制文件名将生成为其他control.log的日志文件。

- 错误文件

  用户若在命令提示符当中没有指明日志文件名，则默认数据文件名将会生成为其他data.bad的错误文件。但是若不存在错误记录，则不生成错误文件。

- 对象表

  用户要加载member表的id、name、job、birthdate、city、age 列的数据。

- FIELDS TERMINATED BY字符串、FIELDS OPTIONALLY ENCLOSED BY字符串、FIELDS ESCAPED BY字符串、LINES TERMINATED BY字符串

  FIELDS TERMINATED BY、FIELDS OPTIONALLY ENCLOSED BY、FIELDS ESCAPED BY字符串由于是固定记录格式的数据，因此不使用它。由于使用LINES FIX语句，因此不必使用LINES TERMINATED BY字符串。

- 加载对象

  从数据文件的第一行数据开始加载。

- 「5.9.15. TRAILING NULLCOLS语句」

  若不存在数据文件的记录值，则用NULL字符来绑定。

数据文件的编辑

数据文件data.dat的内容如下。由于kdLoader Utility接收输入固定记录格式的数据，用户必须得在正确的位置上输入列的数据。

```
7777 KKS        CHAIRMAN  1975-11-18 SEOUL      33
7839 KING       PRESIDEN             DAEGU      45
7934 MILLER     CLERK     1967-01-24 BUSAN
```

kdLoader的运行

如下运行kdLoader实用程序。

```
kdloader userid=loader/loader_pw@default control=./control.ctl
```

确认日志文件和错误文件

确认运行kdLoader Utility之后生成的日志文件和错误文件。

记录kdLoader Utility运行过程的日志文件内容如下。

```
kdLoader 11
Inspur Corporation Copyright (c) 2008-. All rights reserved.

Data File : ./data.dat
Bad File : ./data.bad

Table 'MEMBER' was loaded from the data file.

Column Name                     Position DataType
------------------------------ ---------- ------------
ID                                      1 NUMERIC EXTERNAL
NAME                                    2 CHARACTER
JOB                                     3 CHARACTER
BIRTHDATE                               4 DATE
CITY                                    5 CHARACTER
AGE                                     6 NUMERIC EXTERNAL

Record 4 was rejected.
  TBR-80053 : Some relatively positioned columns are not present in the record.

Record 5 was rejected.
  TBR-80053 : Some relatively positioned columns are not present in the record.

Record 6 was rejected.
  TBR-80053 : Some relatively positioned columns are not present in the record.

Table 'MEMBER'
----------------
6 Rows were requested to load.
3 Rows were loaded successfully.
3 Rows were failed to load because of some errors

Elapsed time was: 00:00:00.022404
```

kdLoader Utility若不存在错误数据，则不会生成错误文件。

5.10.4. 包含大对象型数据的情况

利用Conventional Path Load方法将被分隔记录形态的数据加载到K-DB服务器里。

本例中将显示包含保存大容量二进制数据的数据类型BLOB 列的情况下，用户编辑输入文件的方法。可以保存大容量文本数据的数据类型CLOB的情况与此类似。

控制文件的编辑

控制文件的control.ctl内容如下。

```
LOAD DATA
INFILE  './data.dat'
LOGFILE './logfile.log'
BADFILE './badfile.bad'
APPEND
INTO TABLE member
FIELDS TERMINATED BY ','
       OPTIONALLY ENCLOSED BY '"'
       ESCAPED BY '\\'
LINES TERMINATED BY '\n'
TRAILING NULLCOLS
IGNORE 2 LINES
(
    id integer external,
    name,
    job,
    birthdate,
    city,
    age integer external,
    picture outfile
)
```

下面是在控制文件里指定的信息。

- 数据文件

  当前目录中的data.dat文件。

- 日志文件

  用户若在命令提示符里不指明日志文件名，则在当前的目录里生成名为logfile.log的日志文件。

- 错误文件

  用户若在命令提示符里不指明错误文件名，则在当前的目录里生成名为badfile.bad的错误文件。

  但若不存在错误记录，则不生成错误文件。

- 对象表

用户要加载的名为member的表的id、name、job、birthdate、city、age、picture 列的数据

- FIELDS TERMINATED BY字符串、FIELDS OPTIONALLY ENCLOSED BY字符串、FIELDS ESCAPED BY字符串、LINES TERMINATED BY字符串

  FIELDS TERMINATED BY使用逗号(,)，FIELDS OPTIONALLY ENCLOSED BY字符串使用双引号(")，FIELDS ESCAPED BY字符串使用'\\'，LINES TERMINATED BY字符串使用'\n'。

- 加载对象

  由于存在IGNORE LINES语句，因此从数据文件的第三个行的数据开始加载。

- 「5.9.15. TRAILING NULLCOLS语句」

  不存在数据文件的记录值时，则用NULL字符绑定。

数据文件的编辑

数据文件data.dat的内容如下。为了在BLOB列里上载二进制数据，输入该二进制文件所在的路径。

```
忽略第一行
7782,"Clark","Manager",1981-01-11 , DAEGU,26,./blob.jpg
7839,"King",President,1960/11/17,SEOUL,47
7934,"Miller","Clerk",1977/10/12,BUSAN,30,./blob2.jpg
7566,"Jones",Manager\, ,1981/04/02,31
7499, "Allen", "Salesman" a,1981:02/20,26
7654, "Martin", "Sale smn", 1981/10/28,26
7658, "Chan",Ana lyst, 1982/05/03,25,25
```

kdLoader

如下运行kdLoader实用程序。

```
kdloader userid=loader/loader_pw@default control=./control.ctl
```

确认日志文件与错误文件

确认运行kdLoader Utility以后生成的日志文件和错误文件。

确认运行kdLoader Utility的运行过程的日志文件内容如下。

```
kdLoader 11
Inspur Corporation Copyright (c) 2008-. All rights reserved.

Data File : ./data.dat
Bad File : ./badfile.bad
```

```
Table 'MEMBER' was loaded from the data file.

Column Name                    Position DataType
------------------------------ ---------- ------------
ID                                    1 NUMERIC EXTERNAL
NAME                                  2 CHARACTER
JOB                                   3 CHARACTER
BIRTHDATE                             4 DATE
CITY                                  5 CHARACTER
AGE                                   6 NUMERIC EXTERNAL
PICTURE                               7 RAW

Record 4 was rejected.
  TBR-80025 : Column data exceeds data buffer size.

Table 'MEMBER'
----------------
6 Rows were requested to load.
5 Rows were loaded successfully.
1 Rows were failed to load because of some errors

Elapsed time was: 00:00:00.055475
```

kdLoader Utility记录加载失败的错误数据的错误文件如下。发生错误的数据修改以后必须重新上载到K-DB的数据库里。

```
7499, "Allen", "Salesman" a,1981:02/20,26
```

第6章 kddv

本章将对kddv utility以及使用方法进行说明。

6.1. 概要

kddv是检查K-DB 数据库数据文件整合性的简单实用程序。通过该程序让数据库在脱机状态下也能对数据文件进行基本的整合性检查。

对kddv utility数据块检查如下事项。

- 数据块当中的DBA是否书写正确
- 数据块的检查点是否一致
- 数据块的可用空间和实际使用空间之和是否与数据块大小一致
- 进入数据块的row-piece有无侵犯相互领域

在上述检查项中哪怕有一项不符就会被判断为媒体故障，需要执行media recovery来恢复数据库。

6.2. 快速开始

kddv utility在安装K-DB时会被同时安装，卸载K-DB时会被一起卸载。

kddv utility在命令提示符当中以如下格式运行。

```
$ kddv [-s BLKSIZE] /path/to/datafile
```

命令提示符当中可以用选项指定数据块大小，默认为8192byte。

下面是运行kddv utility的示例。

[例 **6.1] kddv utility**的运行

```
$ kddv df1.dtf
================================================================
= Database Verifier (DV) starts                                =
=                                                              =
= Inspur Corporation Copyright (c) 2008-. All rights reserved. =
================================================================
Verifying 'df1.dtf'...


Verifying complete.
```

```
Total blocks: 1152
Processed blocks: 1063
Empty blocks: 89
Corrupt blocks: 0
```

6.3. 执行示例

在本节中对正常数据文件和有整合性漏洞的数据文件分别进行观察，来确认kddv utility是怎样发现整合性错误的。

记载数据块dba的部分当中发起故障时，dv输出如下结果。

[例 6.2] DBA错误时输出**dv**

```
$ kddv df1.dtf
===================================================================
= Database Verifier (DV) starts                                   =
=                                                                 =
= Inspur Corporation Copyright (c) 2008-. All rights reserved. =
===================================================================
Verifying 'df1.dtf'...

block #2351 is misplaced.dba differs (expected=16779567, real=16779551)

Verifying complete.

Total blocks: 2433
Processed blocks: 2343
Empty blocks: 90
Corrupt blocks: 1
```

从数据文件中发现fractured block时输出数据块未consistent的结果。

[例 6.3] 发现**Fractured block(Inconsistent block)**时输出**dv**

```
$ kddv df1.dtf
===================================================================
= Database Verifier (DV) starts                                   =
=                                                                 =
= Inspur Corporation Copyright (c) 2008-. All rights reserved. =
===================================================================
Verifying 'df1.dtf'...

block #2311 isn't consistent.

Verifying complete.
```

```
Total blocks: 2433
Processed blocks: 2343
Empty blocks: 90
Corrupt blocks: 1
```

记载数据块所剩空间的部分与数据块实际使用空间之差不一致时，dv的输出如下。

[例 **6.4]** 数据块的多余空间与实际使用的空间整合性不符时输出**dv**

```
$ kddv df1.dtf
=====================================================================
= Database Verifier (DV) starts                                     =
=                                                                   =
= Inspur Corporation Copyright (c) 2008-. All rights reserved. =
=====================================================================
Verifying 'df1.dtf'...

block #2004 has incorrect freespace.

Verifying complete.

Total blocks: 2433
Processed blocks: 2343
Empty blocks: 90
Corrupt blocks: 1
```

参考

kddv utility在执行过程当中zero-out的块被视为未被格式化，因此实际磁盘总发生故障使写入数据的块发生zero-out的现象，但kddv还是被判断为未被格式化（即，还未从数据文件被分配的部分）因为不会发出错误。

第7章 Utility API

K-DB里提供与Utility有关的C与C++函数。应用程序开发员可以使用Utility API来从应用程序中调出在之前命令提示符中运行过的Utility。

7.1. 头文件

下面是Utility API使用的头文件。

头文件名	说明
kdutil.h	声明Utility API并定义与其相关的结构。 要定义的结构如下。 – struct sqlstr – struct TBExpImpMeta – struct KDExportIn – struct KDExportOut – struct KDExportStruct – struct KDImportIn – struct KDImportOut – struct KDImportStruct
sqlca.h	将Utility内部所发生的错误信息传送到应用程序的结构。 要声明的结构如下。 – struct sqlca
sqlcli.h	是ODBC标准头文件，定义标准API和宏。

7.2. 结构

下面是Utility API里可以定义的结构。

- sqlstr结构

字段	用途	类型	说明
length	输入	SQLSMALLINT	指明字符串的长度。
data	输入	SQLCHAR *	指明字符串所保存的指针。

- KDExpImp结构体

字段	用途	类型	说明
fieldTerm	输入	SQLCHAR *	是列（字段）的区分符，指明要使用的字符串。
fieldTermLen	输入	SQLSMALLINT	是列（字段）的区分符，指明要使用的字符串的长度。
lineTerm	输入	SQLCHAR *	是记录的区分符，指明要使用的字符串。
lineTermLen	输入	SQLSMALLINT	是记录的区分符，指明要使用的字符串的长度。
enclStart	输入	SQLCHAR *	指明包括列的数据的开始字符串。 开始字符串由双引号(")指明在列的数据最前面会有双引号。
enclStartLen	输入	SQLSMALLINT	指明包括列的数据的开始字符串的长度。
enclEnd	输入	SQLCHAR *	指明包含数据的结束字符串。 结束字符串由双引号(")指明在列的数据最后面会有双引号。
enclEndLen	输入	SQLSMALLINT	指明包含列的数据的结束字符串的长度。
escape	输入	SQLCHAR *	解释列的数据时，指明所需ESCAPE字符串。 但是只在KDImport结构里使用。
escapeLen	输入	SQLSMALLINT	解释列的数据时，指明所需ESCAPE字符串的长度。 但是只在KDImport结构里使用。

- KDExportIn结构

字段	用途	类型	说明
iMeta	输入	KDExpImpMeta *	是KDExpImp结构的指针。 指明在抽取数据时索取输入源数据。

- KDExportOut结构

字段	用途	类型	说明
oRowsExported	输出	SQLINTEGER	在数据文件里被记录所抽取记录的个数。

- KDExportStruct结构

字段	用途	类型	说明
piDataFileName	输入	SQLCHAR *	指明保存所提取数据的文件路径。
iDataFileNameLen	输入	SQLSMALLINT	指明数据文件名的长度。 文字名以NULL字符结束时，指明SQL_NTS。
piActionString	输入	sqlstr *	利用SELECT字符指定目标表或视图数据。SELECT语句中指明的列的顺序来提取的数据。
iFileType	输入	SQLSMALLINT	现在只可以使用使用了列区分符的格式，SQL_DEL。
piMsgFileName	输入	SQLCHAR *	指明记录抽取数据时发生的错误和警告以及有用信息的消息文件名。
iMsgFileNameLen	输入	SQLSMALLINT	指明消息文件名的长度。 文件名以NULL字符结束时，指明SQL_NTS。
piExportInfoIn	输入	KDExportIn *	KDExportIn结构的指针。 指明抽取数据时所需的输入源数据。
piExportInfoOut	输入	KDExportOut *	KDExportOut结构的指针。 在抽取数据以后，保存发生的结果信息。

- KDImport结构

字段	用途	类型	说明
iMeta	输入	KDExpImpMeta *	KDExpImp结构的指针。 指明加载数据时所需的输入源数据。

字段	用途	类型	说明
iRowCount	输入	SQLINTEGER	指明要加载的记录。 用户把值指定为0时，加载数据文件的所有记录。
iSkipCount	输入	SQLINTEGER	在数据文件里为了除去要加载的对象，指明要跳过的个数。
iCommitCount	输入	SQLINTEGER	在加载数据时若要提交一次时，则指明其个数。但是由于性能型的问题，不会以用户指明的个数单位进行提交。
iErrorCount	输入	SQLINTEGER	是用户允许的错误记录的个数。在加载数据时，若错误记录个数比允许的值多时，中断加载。 输入-1 时，可以忽略错误记录的个数，执行工具。

- KDImport结构

字段	用途	类型	说明
oRowsRead	输出	SQLINTEGER	记录数据文件里读取的记录个数。
oRowsSkipped	输出	SQLINTEGER	记录数据文件里跳过的记录个数。
oRowsInserted	输出	SQLINTEGER	记录该表或视图里输入（INSERT运算）的记录个数。
oRowsUpdated	输出	SQLINTEGER	记录该表或视图里被修改（UPDATE运算）的记录个数。
oRowsRejected	输出	SQLINTEGER	记录数据加载失败的记录个数。
oRowsCommitted	输出	SQLINTEGER	记录提交成功的记录个数。

- KDImportStruct结构

字段	用途	类型	说明
piDataFileName	输入	SQLCHAR *	指明要加载的数据文件的路径名。
iDataFileNameLen	输入	SQLSMALLINT	指明数据文件名的长度。 文件名以NULL字符结束时，指明SQL_NTS。
piActionString	输入	sqlstr *	是指明要下载的对象表和列，指明详细源数据的字符串。 是与kdLoader Utility控制文件里使用的字符串相同的语法。该语法的相关内容请参照kdLoader Utility的控制文件的格式。

字段	用途	类型	说明
iFileType	输入	SQLSMALLINT	只能使用现在使用列区分符的格式SQL_DEL。
piMsgFileName	输入	SQLCHAR *	指明记录下载数据时发生的错误和警告以及有用信息的消息文件名。
iMsgFileNameLen	输入	SQLSMALLINT	指明消息文件名的长度。 文件名以NULL字符结束时，指明SQL_NTS。
piBadFileName	输入	SQLCHAR *	收集下载数据时发生错误的数据记录的错误数据文件。
iBadFileNameLen	输入	SQLSMALLINT	指明错误数据文件名的长度。 文件名以NULL字符结束时，指明SQL_NTS。
iDPL	输入	BOOL	下载数据时，指定是否Direct Path Load方式。
iTrailNullCols	输入	BOOL	指定是否将数据文件记录中没有的最后一栏的数据视为NULL。
piImportInfoIn	输入	KDImportIn *	KDImportIn结构的指针。 指定下载数据时所需的输入源数据。
piImportInfoOut	输入	KDImportOut *	KDImportOut结构的指针。 加载下载数据后所发生的结果信息。

7.3. Utility API目录

下面是Utility API目录。

函数类型	函数	头文件
数据库连接	KDConnect	kdutil.h
解除数据库连接	KDDisconnect	kdutil.h
抽取数据	KDExport	kdutil.h
下载数据	KDImport	kdutil.h

7.3.1. KDConnect

是利用数据库连接信息（数据库，用户名，密码）连接K-DB服务器的函数。

KDConnect函数的详细内容如下。

- 语法

```
SQLRETURN SQL_API
TBConnect(SQLCHAR *dnsname, SQLCHAR *username, SQLCHAR *pwd,
          struct sqlca *pSqlca);
```

- 参数

参数	用途	说明
dnsname	输入	kddsn.tbr(或 kdnet_alias.tbr）文件里定义的数据库。
username	输入	指定用户名。
pwd	输入	指定密码。
pSqlca	输出	返回的代码若不是SQL_SUCCESS，则将包含Utility内部发生的错误信息。

- 返回代码

返回代码	说明
SQL_SUCCESS	函数成功结束的状态。
SQL_SUCCESS_WITH_INFO	函数成功结束，但存在警告消息的状态。
SQL_ERROR	发生致命性错误的状态。

- 相关函数

  KDDisconnect

7.3.2. KDDisconnect

相当于数据库连接信息（数据库）解除与K-DB服务器的连接的函数

`KDDisconnect` 函数的详细内容如下。

- 语法

```
SQLRETURN SQL_API
KDDisconnect(SQLCHAR *dnsname, struct sqlca *pSqlca);
```

- 参数

参数	用途	说明
dnsname	输入	kddsn.tbr(或 kdnet_alias.tbr）文件里定义的数据库。
pSqlca	输出	返回代码若不是SQL_SUCCESS，则将包含Utility内部发生的错误信息。

- 返回代码

返回代码	说明
SQL_SUCCESS	函数成功结束的状态。
SQL_SUCCESS_WITH_INFO	函数成功结束，但存在警告消息的状态。
SQL_ERROR	发生致命性错误的状态。

- 相关函数

KDConnect

7.3.3. KDExport

从外部文件抽取数据库里存在的数据的函数。将数据库数据提取到外部文件中的函数。外部函数将抽取的数据指明为文本格式，利用列区分符和记录区分符抽取列和记录。这样抽取的外部文件采取kdLoader Utility的数据文件格式。用户可以利用SELECT语句选择要抽取的数据。但必须要有表或视图的SELECT权限。

`KDExport`函数的详细内容如下。

- 语法

```
SQLRETURN SQL_API
KDExport(SQLINTEGER versionNumber, KDExportStruct *pParamStruct,
         struct sqlca *pSqlca);
```

- 参数

参数	用途	说明
versionNumber	输入	是Utility Libarary版本编号。该参数在应对Utility libarary的更改时，为了与下级版本兼容而存在。 现在版本编号为1。
pParamStruct	输入输出	利用该参数指明要抽取的对象信息，抽取后接收结果信息。 详细信息请参照KDExportStruct结构。
pSqlca	输出	返回代码不是SQL_SUCCESS时，包含Utility内部发生的错误信息。

- 返回代码

返回代码	说明
SQL_SUCCESS	函数成功结束的状态。
SQL_SUCCESS_WITH_INFO	函数成功结束，但仍然存在警告消息的状态。
SQL_ERROR	发生致命性错误的状态。

- 示例

```
#include "kdutil.h"

#define DNS_NAME  "DEFAULT"
#define USER_NAME "SYS"
#define PWD       "kdb"

int main(int argc, char *argv[]) {
    SQLRETURN      rc = SQL_SUCCESS;
    SQLINTEGER     versionNumber = 1;
    SQLCHAR        dataFileName[256];
    SQLCHAR        actionString[256];
    SQLCHAR        msgFileName[256];
    SQLCHAR        fieldTerm[5];
    SQLCHAR        lineTerm[5];
    SQLCHAR        enclStart[5];
    SQLCHAR        enclEnd[5];

    struct sqlca   ca           = {"\0", 0, 0, {0, "\0"}, "\0",
                                   {0, 0, 0, 0, 0, 0}, "\0", "\0"};
    KDExportStruct exportStruct = {NULL, 0, NULL, NULL, NULL, 0, NULL,
                                   0, NULL, NULL};
    KDExportIn     exportIn     = {{NULL, 0, NULL, 0, NULL, 0, NULL, 0, NULL, 0}};

    KDExportOut    exportOut    = {0};
```

```
rc= KDConnect((SQLCHAR *)DNS_NAME, (SQLCHAR *)USER_NAME, (SQLCHAR *)PWD, &ca);


if (rc != SQL_SUCCESS) return -1;

strcpy((char *)dataFileName, "./all_tables.dat");
strcpy((char *)actionString, "select * from all_tables");
strcpy((char *)msgFileName, "./all_tables.log");
strcpy((char *)fieldTerm, ",");
strcpy((char *)lineTerm, "\n");
strcpy((char *)enclStart, "\"");
strcpy((char *)enclEnd, "\"");

/* setting data file name */
exportStruct.piDataFileName   = dataFileName;
exportStruct.iDataFileNameLen = strlen((char *)dataFileName);

/* setting action String */
exportStruct.piActionString = calloc(1, sizeof(sqlstr));
exportStruct.piActionString->data   = actionString;
exportStruct.piActionString->length = strlen((char *)actionString);

/* setting file type */
exportStruct.iFileType = SQL_DEL;

/* setting message file name */
exportStruct.piMsgFileName   = msgFileName;
exportStruct.iMsgFileNameLen = strlen((char *)msgFileName);

/* setting field term, line term etc.. */
exportIn.iMeta.fieldTerm    = fieldTerm;
exportIn.iMeta.fieldTermLen = strlen((char *)fieldTerm);
exportIn.iMeta.lineTerm     = lineTerm;
exportIn.iMeta.lineTermLen  = strlen((char *)lineTerm);
exportIn.iMeta.enclStart    = enclStart;
exportIn.iMeta.enclStartLen = strlen((char *)enclStart);
exportIn.iMeta.enclEnd      = enclEnd;
exportIn.iMeta.enclEndLen   = strlen((char *)enclEnd);

/* setting export input, output information */
exportStruct.piExportInfoIn  = &exportIn;
exportStruct.poExportInfoOut = &exportOut;

/* setting file type */
rc = KDExport(versionNumber, &exportStruct, &ca);
if (rc != SQL_SUCCESS) return -1;
```

```
    /* disconnect */
    rc= KDDisconnect((SQLCHAR *)DNS_NAME, &ca);
    if (rc != SQL_SUCCESS) return -1;

    return 1;
}
```

7.3.4. KDImport

是将外部文件里的数据下载到数据库里的函数。在这里外部文件支持kdLoader Utility所支持的固定记录格式和分隔记录格式。但是必须要有对象表或视图的INSERT权限。

KDImport函数的详细内容如下。

- 语法

```
SQLRETURN SQL_API KDImport(SQLINTEGER versionNumber,
                           KDImportStruct *pParamStruct,
                           struct sqlca *pSqlca);
```

- 参数

参数	用途	说明
versionNumber	输入	是Utility Libarary版本编号。该参数在应对Utility libarary的更改时，为了与下级版本兼容而存在。 现在版本编号为1。
pParamStruct	输出入	利用该参数指明要下载的对象信息，下载后接收结果信息。 详细信息请参照"TBImportStruct结构"。
pSqlca	输出	返回代码不是SQL_SUCCESS时，包含Utility内部发生的错误信息。

- 返回代码

返回代码	说明
SQL_SUCCESS	函数成功结束的状态。
SQL_SUCCESS_WITH_INFO	函数成功结束，但仍然存在警告消息的状态。
SQL_ERROR	发生致命性错误的状态。

- 示例

```
#include "kdutil.h"

#define DNS_NAME  "DEFAULT"
#define USER_NAME "SYS"
#define PWD       "kdb"

int main(int argc, char *argv[]) {
 SQLRETURN      rc = SQL_SUCCESS;
 SQLINTEGER     versionNumber = 1;
 SQLCHAR        dataFileName[256];
 SQLCHAR        actionString[256];
 SQLCHAR        msgFileName[256];
 SQLCHAR        badFileName[256];
 SQLCHAR        fieldTerm[5];
 SQLCHAR        lineTerm[5];
 SQLCHAR        enclStart[5];
 SQLCHAR        enclEnd[5];
 SQLCHAR        escape[5];

 struct sqlca   ca           = {"\0", 0, 0, {0, "\0"}, "\0",
                               {0, 0, 0, 0, 0, 0}, "\0", "\0"};
 KDImportStruct importStruct = {NULL, 0, NULL, NULL, NULL, 0, NULL,
                                0, NULL, 0, 0, NULL, NULL};
 KDImportIn     importIn     = {{NULL, 0, NULL, 0, NULL, 0, NULL, 0, NULL, 0},
                                0, 0, 0, 0};
 KDImportOut    importOut    = {0, 0, 0, 0, 0, 0};

 rc= KDConnect((SQLCHAR *)DNS_NAME, (SQLCHAR *)USER_NAME, (SQLCHAR *)PWD, &ca);
 if (rc != SQL_SUCCESS) return -1;

 strcpy((char *)actionString, "LOAD DATA "
                              "APPEND "
                              "INTO TABLE DEPT "
                              "MULTI INSERT INDEXES "
                             "(position, deptno, dname CONSTANT \"co dep\", loc)");

 strcpy((char *)dataFileName, "./test.dat");
 strcpy((char *)msgFileName, "./test.log");
 strcpy((char *)badFileName, "./test.bad");
 strcpy((char *)fieldTerm,   ",b");
 strcpy((char *)lineTerm,    "abbb\n");
 strcpy((char *)enclStart,   "{$");
 strcpy((char *)enclEnd,     "$}");
 strcpy((char *)escape,      "XX");

 /* setting data file name */
 importStruct.piDataFileName   = dataFileName;
```

```
importStruct.iDataFileNameLen = strlen((char *)dataFileName);

/* setting action String */
importStruct.piActionString = calloc(1, sizeof(sqlstr));
importStruct.piActionString->data   = actionString;
importStruct.piActionString->length = strlen((char *)actionString);

/* setting file type */
importStruct.iFileType = SQL_DEL;

/* setting message file name */
importStruct.piMsgFileName   = msgFileName;
importStruct.iMsgFileNameLen = strlen((char *)msgFileName);

/* setting bad data file name */
importStruct.piBadFileName   = badFileName;
importStruct.iBadFileNameLen = strlen((char *)badFileName);

/* turn on DPL mode */
importStruct.iDPL            = 2;

/* setting field term, line term etc.. */
importIn.iMeta.fieldTerm    = fieldTerm;
importIn.iMeta.fieldTermLen = strlen((char *)fieldTerm);
importIn.iMeta.lineTerm     = lineTerm;
importIn.iMeta.lineTermLen  = strlen((char *)lineTerm);
importIn.iMeta.enclStart    = enclStart;
importIn.iMeta.enclStartLen = strlen((char *)enclStart);
importIn.iMeta.enclEnd      = enclEnd;
importIn.iMeta.enclEndLen   = strlen((char *)enclEnd);
importIn.iMeta.escape       = escape;
importIn.iMeta.escapeLen    = strlen((char *)escape);

importIn.iRowcount          = 0;
importIn.iSkipcount         = 0;
importIn.iCommitcount       = 2;
importIn.iErrorcount        = 50;

/* setting export input, output information */
importStruct.piImportInfoIn  = &importIn;
importStruct.poImportInfoOut = &importOut;

/* setting file type */
rc = KDImport(versionNumber, &importStruct, &ca);
if (rc != SQL_SUCCESS) return -1;

  fprintf(stdout, "oRowsRead[%ld]", importOut.oRowsRead);
```

```
   fprintf(stdout, "oRowsSkipped[%ld]", importOut.oRowsSkipped);
   fprintf(stdout, "oRowsInserted[%ld]", importOut.oRowsInserted);
   fprintf(stdout, "oRowsUpdated[%ld]", importOut.oRowsUpdated);
   fprintf(stdout, "oRowsRejected[%ld]", importOut.oRowsRejected);
   fprintf(stdout, "oRowsCommitted[%ld]", importOut.oRowsCommitted);

   /* disconnect */
   rc= KDDisconnect((SQLCHAR *)DNS_NAME, &ca);
   if (rc != SQL_SUCCESS) return -1;

   return 1;
}
```

索引

S

U

Y