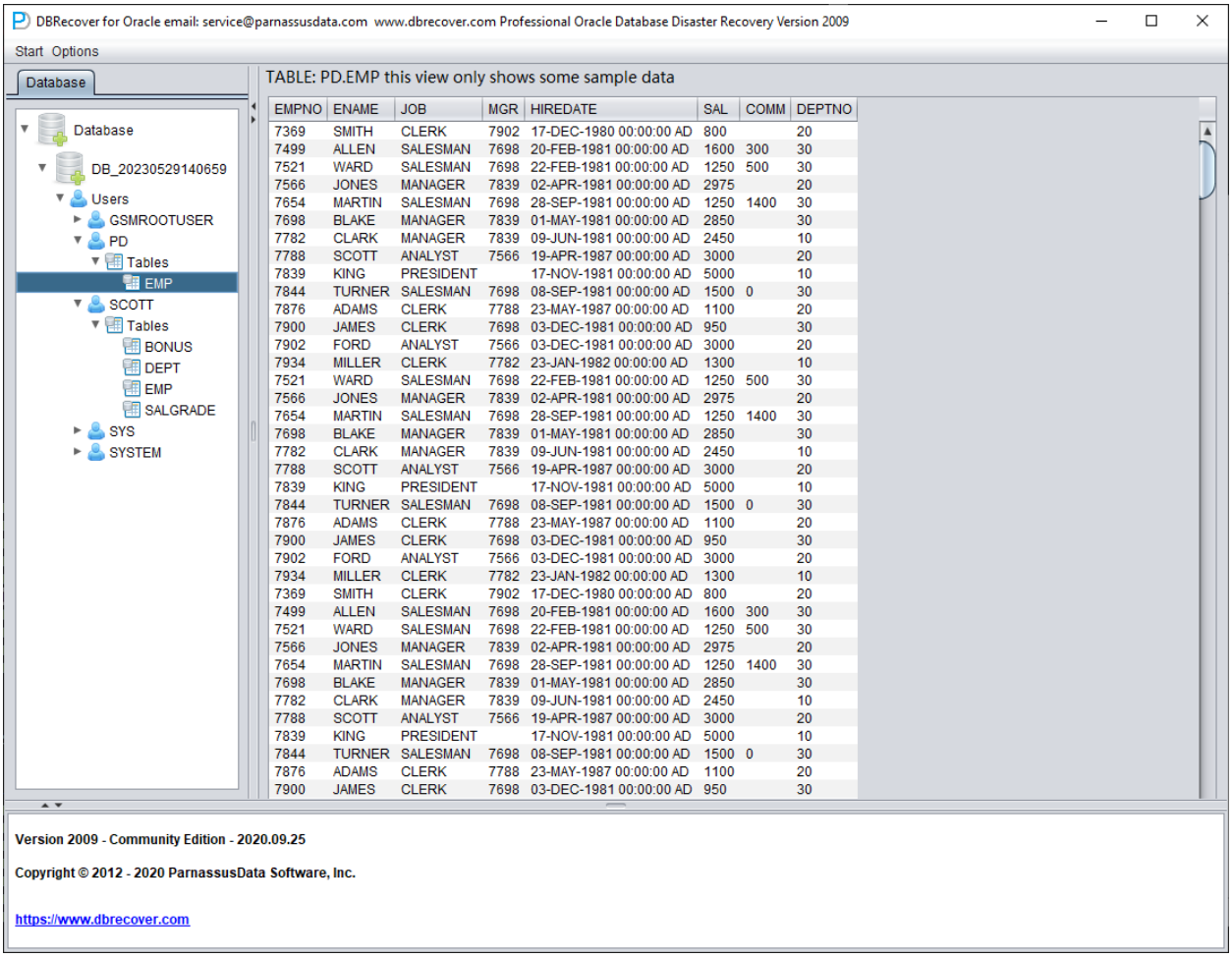
dbrecover for oracle 用户手册

DBRECOVER for Oracle 使用手册0.5

概要

DBRECOVER for Oracle是一款企业级的Oracle数据灾难恢复软件。它能直接从Oracle 8i到21c的数据库数据文件（datafile）中提取并恢复数据表上的数据，无需通过Oracle数据库实例执行SQL来救回数据。基于Java开发的DBRECOVER，无需额外安装，下载解压后即可直接使用。

DBRECOVER采用了直观的GUI图形界面，操作简单便捷。用户无需额外学习一套命令，也无需理解Oracle的底层数据结构原理，就能通过恢复向导（Recovery Wizard）轻松恢复数据库中的数据。



为何选择DBRECOVER？

也许您会疑惑，难道使用RMAN这个传统的Oracle恢复管理器的备份恢复还不够吗？为什么我们需要选择DBRECOVER？让我解答您的疑惑。

随着企业IT系统的快速增长，数据容量正以几何级数增加。Oracle DBA在保证数据完整性的问题上，常面临现有磁盘存储系统容量不足以存放全量备份，基于磁带的数据备份在恢复数据时需要的平均修复时间远超预期等问题。

"对于数据库而言，备份重于一切"，这是所有DBA都谨记在心的格言。然而，现实环境千差万别，企业的数据库环境中数据备份空间不足，采购的存储设备短期内无法到货，甚至在数据恢复过程中发现备份实际不可用，这些都是常见的情况。

为了解决这些在现实世界中常见的数据恢复困境，DBRECOVER软件充分发挥其对Oracle数据库内部数据结构，核心启动流程等内部原理的理解，可以应对在完全没有备份的情况下，如SYSTEM表空间丢失、误操作Oracle数据字典表、由于断电引起的数据字典不一致等导致数据库无法顺利打开的情况。它可以挽回误截断（Truncate）/删除（Drop/Delete）/业务数据表等人为的误操作，并从容地恢复数据。

甚至仅仅接触过Oracle数据库几天的非DBA人员也可以轻松地使用DBRECOVER。这得益于DBRECOVER简单的安装和全程图形化的人机交互界面。实施恢复的人员不需要专业的数据库知识，不需要学习任何命令，更无需了解数据库底层的存储结构。只需轻轻点击几下鼠标，就能从容地恢复数据。DBRECOVER打破了这种只有少数专业人士才能实施数据库恢复任务的限制，大大缩短了从数据库故障到完整恢复数据的时间，降低了企业恢复数据的总成本。

DBRECOVER可恢复的数据可分为两种形式。传统抽取方式将数据从数据文件中完整抽取并写入平面文本文件，随后使用SQLLDR等工具再导入到数据库中。这种方式简单直观，但需要相当于现有数据容量两倍的空间：一是平面文本数据所占的空间，二是将文本数据导入数据库所需的空间；同时在时间上，需要先从数据文件中抽取原始数据，然后才能导入到新建数据库中，通常需要两倍的时间。

我们强烈推荐另一种方式，即DBRECOVER创新的数据搭桥（DataBridge）方式。该方式直接通过DBRECOVER将抽取出的数据加载到新建或其他可用的数据库中，避免了数据落地存储。与传统方式相比，有效节省了数据恢复所需的空间和时间成本。

Oracle的ASM（Automatic Storage Management）技术正在被越来越多的企业所采纳。相较于传统的文件系统，使用ASM存储的数据库具有高性能，支持集群以及便利管理等优势。然而，ASM的问题在于对于普通用户来说，ASM的存储结构过于复杂且难以理解。一旦ASM中的某个Disk Group的内部数据结构损坏导致无法成功地进行MOUNT，用户的重要数据就会被“锁死”在这个ASM的“黑盒”中。在这种情况下，通常需要熟悉ASM内部数据结构的Oracle原厂的资深工程师到达用户现场后手动修复ASM内部结构；而购买Oracle原厂的现场服务对于普通用户来说，往往既昂贵又耗时。

正因为DBRECOVER的研发人员对Oracle ASM内部数据结构有深入的理解，所以DBRECOVER加入了专门针对ASM的数据恢复功能。

目前，DBRECOVER所支持的ASM数据恢复功能包括：

1. 即使在Disk Group无法正常MOUNT的情况下，仍可以通过DBRECOVER直接读取ASM磁盘上的可用元数据(metadata)，并基于这些元数据将Disk Group中的ASM文件拷贝出来。
2. 即使在Disk Group无法正常MOUNT的情况下，仍可以通过DBRECOVER直接读取ASM上的数据文件，并从其中抽取数据，同时支持传统抽取方式和数据搭桥方式。

DBRECOVER For Oracle软件介绍

DBRECOVER For Oracle基于JAVA开发，这保证了其可以跨平台运行，无论是AIX、Solaris、HPUX等Unix平台， Redhat、Oracle Linux、SUSE等Linux平台，还是Windows平台上均可以直接运行。

DBRECOVER支持的操作系统平台：

Platform Name	是否支持
Windows	支持
AIX	支持
Solaris Sparc/X86	支持
Linux x86/64	支持
HPUX	支持
MacOS	支持

DBRECOVER目前支持的数据库版本: 8i ~ 21C

DBRECOVER自带了运行所需的JAVA环境，所以在Windows/Linux上无需另外安装JAVA软件。

在Windows上双击运行start_dbrecover_windows_local_java.bat

在Linux上执行： sh start_dbrecover_linux_local_java.sh

对于AIX/HPUX/Solaris等类UNIX环境，需要用户自行安装JAVA 8环境。

DBRECOVER支持的数据库字符集：

语言	字符集	编码

中文 简体/繁体	ZHS16GBK	GBK
中文 简体/繁体	ZHS16DBCS	CP935
中文 简体/繁体	ZHT16BIG5	BIG5
中文 简体/繁体	ZHT16DBCS	CP937
中文 简体/繁体	ZHT16HKSCS	CP950
中文 简体/繁体	ZHS16CGB231280	GB2312
中文 简体/繁体	ZHS32GB18030	GB18030
日文	JA16SJIS	SJIS
日文	JA16EUC	EUC_JP
日文	JA16DBCS	CP939
韩语	KO16MSWIN949	MS649
韩语	KO16KSC5601	EUC_KR
韩语	KO16DBCS	CP933
法语	WE8MSWIN1252	CP1252
法语	WE8ISO8859P15	ISO8859_15
法语	WE8PC850	CP850
法语	WE8EBCDIC1148	CP1148
法语	WE8ISO8859P1	ISO8859_1
法语	WE8PC863	CP863
法语	WE8EBCDIC1047	CP1047
法语	WE8EBCDIC1147	CP1147
德语	WE8MSWIN1252	CP1252
德语	WE8ISO8859P15	ISO8859_15
德语	WE8PC850	CP850
德语	WE8EBCDIC1141	CP1141
德语	WE8ISO8859P1	ISO8859_1
德语	WE8EBCDIC1148	CP1148
意大利语	WE8MSWIN1252	CP1252
意大利语	WE8ISO8859P15	ISO8859_15
意大利语	WE8PC850	CP850
意大利语	WE8EBCDIC1144	CP1144
泰语	TH8TISASCII	CP874
泰语	TH8TISEBCDIC	TIS620

阿拉伯语	AR8MSWIN1256	CP1256
阿拉伯语	AR8ISO8859P6	ISO8859_6
阿拉伯语	AR8ADOS720	CP864
西班牙语	WE8MSWIN1252	CP1252
西班牙语	WE8ISO8859P1	ISO8859_1
西班牙语	WE8PC850	CP850
西班牙语	WE8EBCDIC1047	CP1047
葡萄牙语	WE8MSWIN1252	CP1252
葡萄牙语	WE8ISO8859P1	ISO8859_1
葡萄牙语	WE8PC850	CP850
葡萄牙语	WE8EBCDIC1047	CP1047
葡萄牙语	WE8ISO8859P15	ISO8859_15
葡萄牙语	WE8PC860	CP860

DBRECOVER支持的表存储类型：

表存储类型	是否支持
Cluster Table簇表	YES
索引组织表，分区或非分区	NO
普通堆表，分区或非分区	YES
普通堆表 启用基本压缩	NO
普通堆表 启用高级压缩	NO
普通堆表 启用混合列压缩	NO
普通堆表 启用加密	NO
带有虚拟字段virtual column的表	NO
链式行、迁移行 chained rows 、migrated rows	YES

注意事项： 对于virtual column、11g optimized default column而言 数据抽取可能没问题，但会丢失对应的字段。 这二者都是11g之后的新特性，使用者较少。

DBRECOVER支持的列字段数据类型：

数据类型	是否支持
BFILE	No
Binary XML	No
BINARY_DOUBLE	Yes
BINARY_FLOAT	Yes
BLOB	Yes
CHAR	Yes
CLOB and NCLOB	Yes
Collections (including VARRAYS and nested tables)	No
Date	Yes
INTERVAL DAY TO SECOND	Yes
INTERVAL YEAR TO MONTH	Yes
LOBs stored as SecureFiles	Yes
LONG	Yes
LONG RAW	Yes
Multimedia data types (including Spatial, Image, and Oracle Text)	No
NCHAR	Yes
Number	Yes
NVARCHAR2	Yes
RAW	Yes
ROWID, UROWID	Yes
TIMESTAMP	Yes
TIMESTAMP WITH LOCAL TIMEZONE	Yes
TIMESTAMP WITH TIMEZONE	Yes
User-defined types	No
VARCHAR2 and VARCHAR	Yes
XMLType stored as CLOB	No
XMLType stored as Object Relational	No

DBRECOVER对ASM的支持:

功能	Supported
支持直接从ASM中抽取数据，无需拷贝到文件系统上	YES

支持从ASM中拷贝数据文件	YES

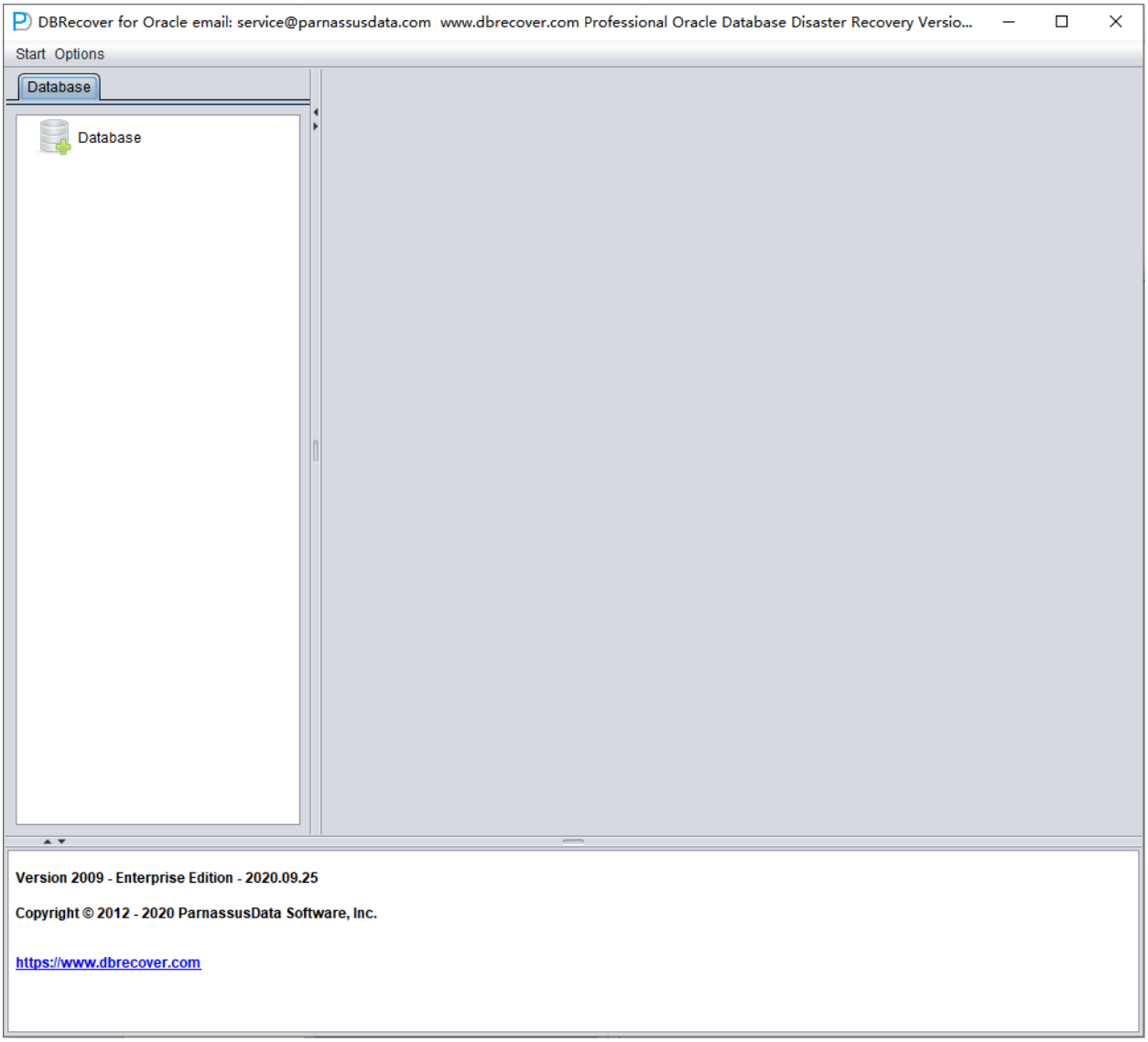
DBRECOVER的安装与启动

由于DBRECOVER是基于JAVA开发的纯绿色软件，所以无需额外安装，用户仅需要在下载软件ZIP包后解压即可用于恢复数据。

在Windows上双击运行start_dbrecover_windows_local_java.bat

在Linux环境下可以在本机图形化界面或者通过Xmanager/VNC等远程图形化工具使用

1. 确认可以打开xclock图形化时钟小程序
2. 在软件解压目录下执行： sh start_dbrecover_linux_local_java.sh



在AIX/HPUX/Solaris环境下可以在本机图形化界面或者通过Xmanager/VNC等远程图形化工具使用

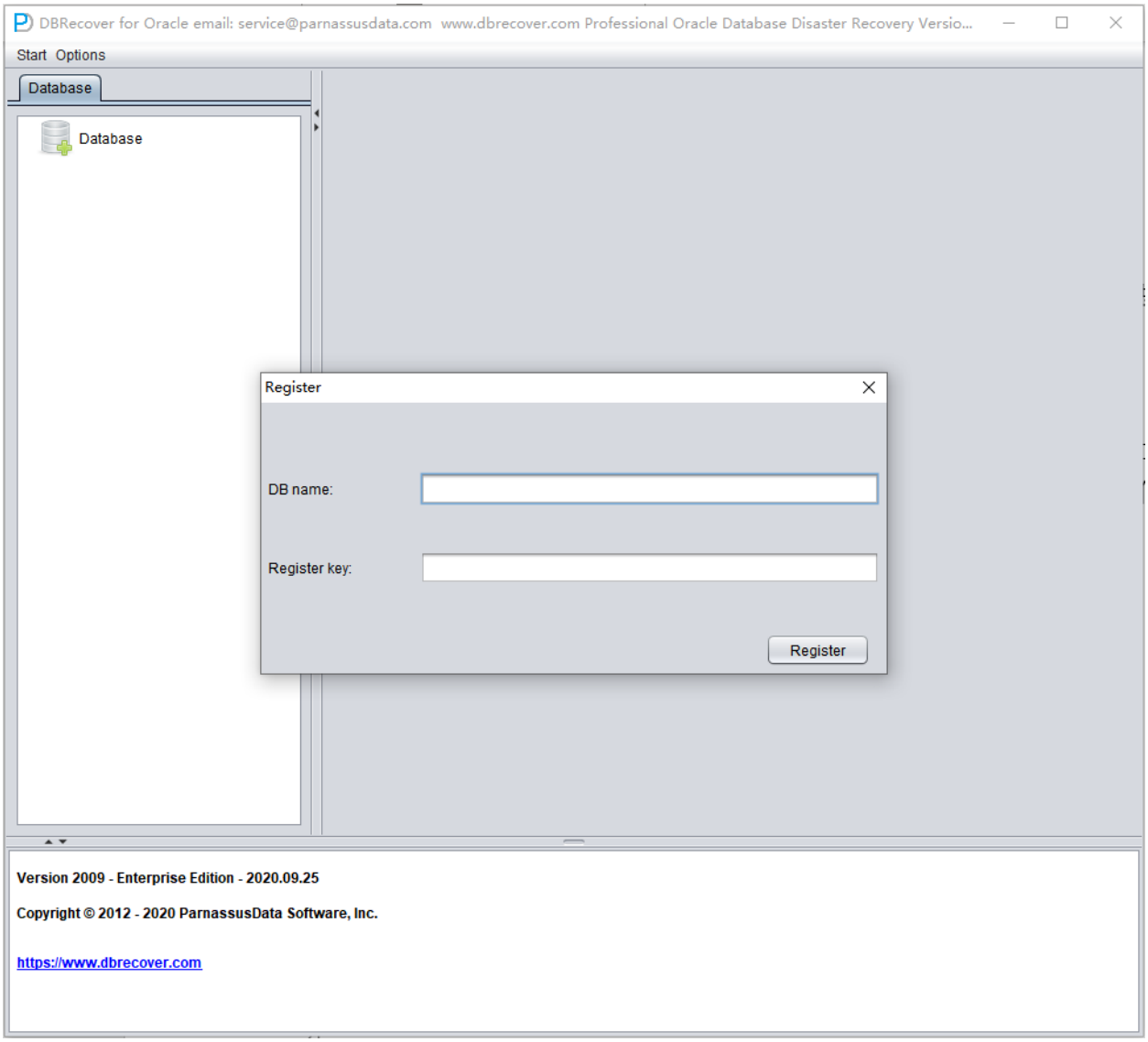
1. 确认已安装对应平台JAVA 8环境，并使用命令确认：java -version
2. 确认可以打开xclock图形化时钟小程序
3. 在软件解压目录下执行： sh start_dbrecover.sh

DBRECOVER的许可证注册

DBRECOVER For Oracle是一款商业软件。DBRECOVER的社区版可供用户测试和学习。

我们目前仅仅提供一种许可证类型，即企业版许可证。可以登陆网址https://www.dbrecover.com/获取购买信息。

用户获得License Key之后可以自行在软件中注册Register，具体使用方法为：



在菜单栏Help => Register，按照购买后发送给您的信息输入DB NAME和密钥并点击Register按钮即可。完成注册后，今后重新启动DBRECOVER将自动检测License注册信息，无需重复注册。

成功注册的信息可以在Help=>About中找到：

基于不同的Oracle数据库恢复场景介绍如何使用DBRECOVER

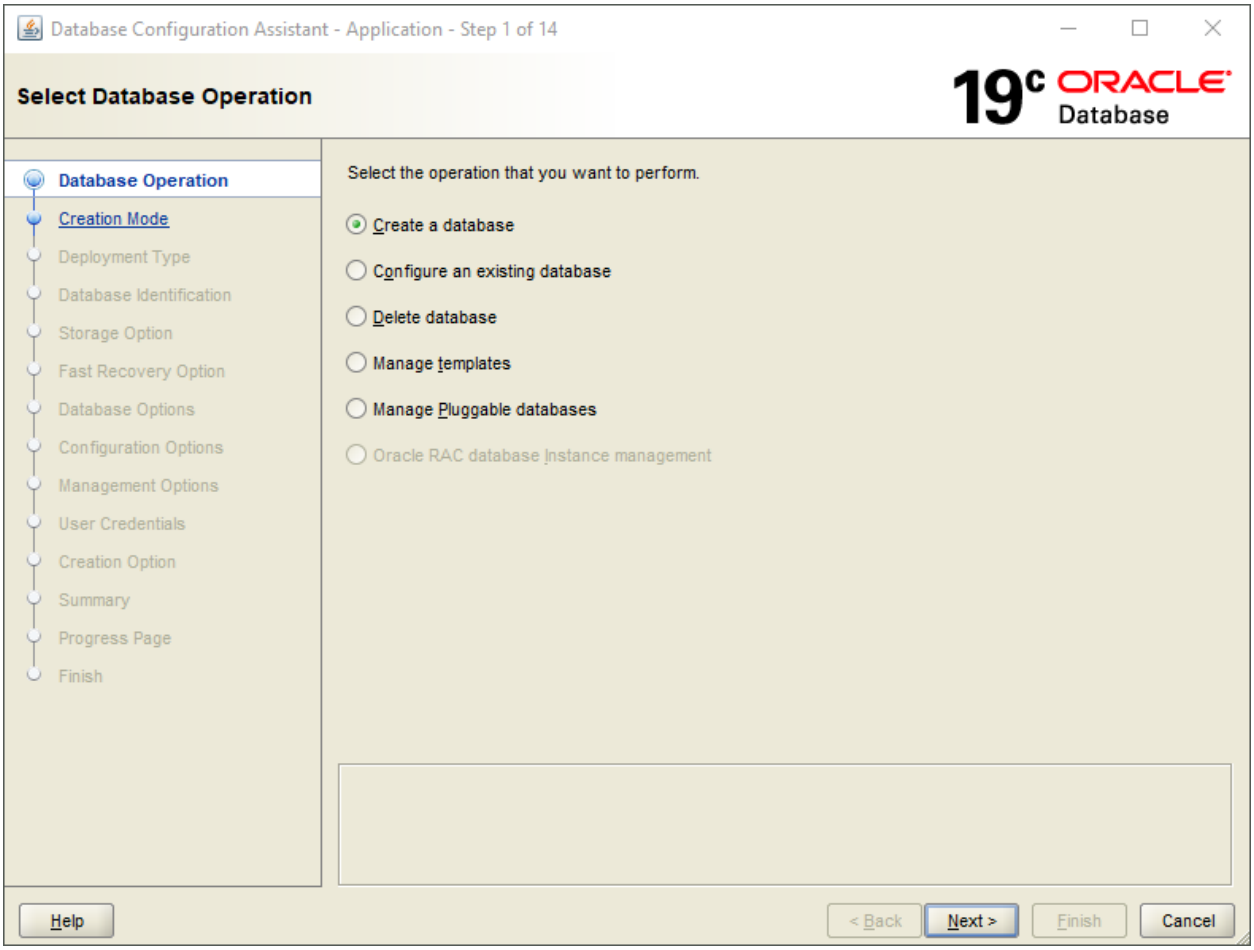
恢复场景1 ORACLE数据文件损坏导致数据库无法打开OPEN

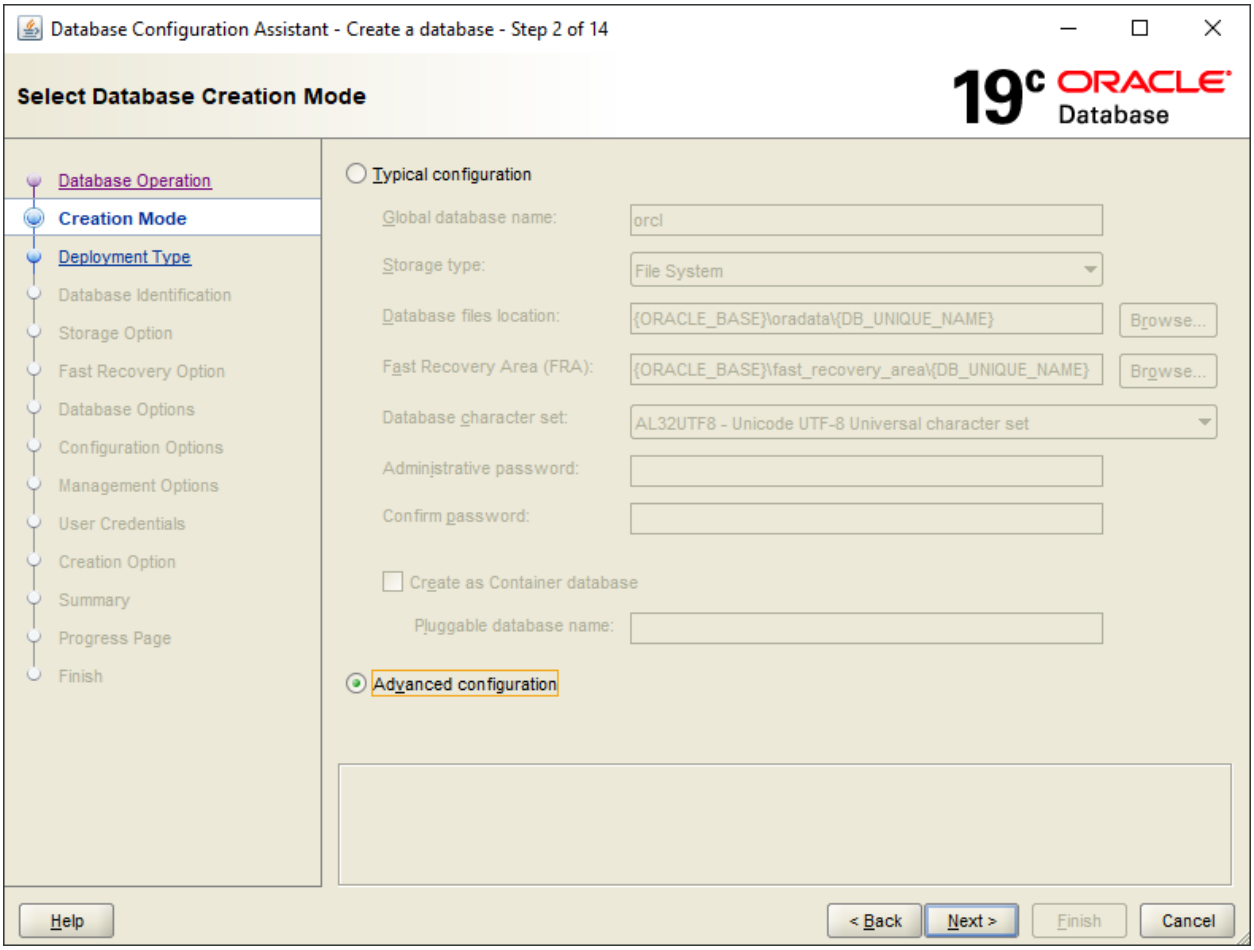
A公司的生产数据库常年运行在非归档模式下，偶尔做下EXP的逻辑备份，从来不做物理备份。某日服务器断电重启后数据库无法正常OPEN使用，检测后发现SYSTEM表空间损坏严重。此时可以使用DBRECOVER快速将受损数据库中的数据传输到新建数据库中，以达到快速恢复业务的目的。

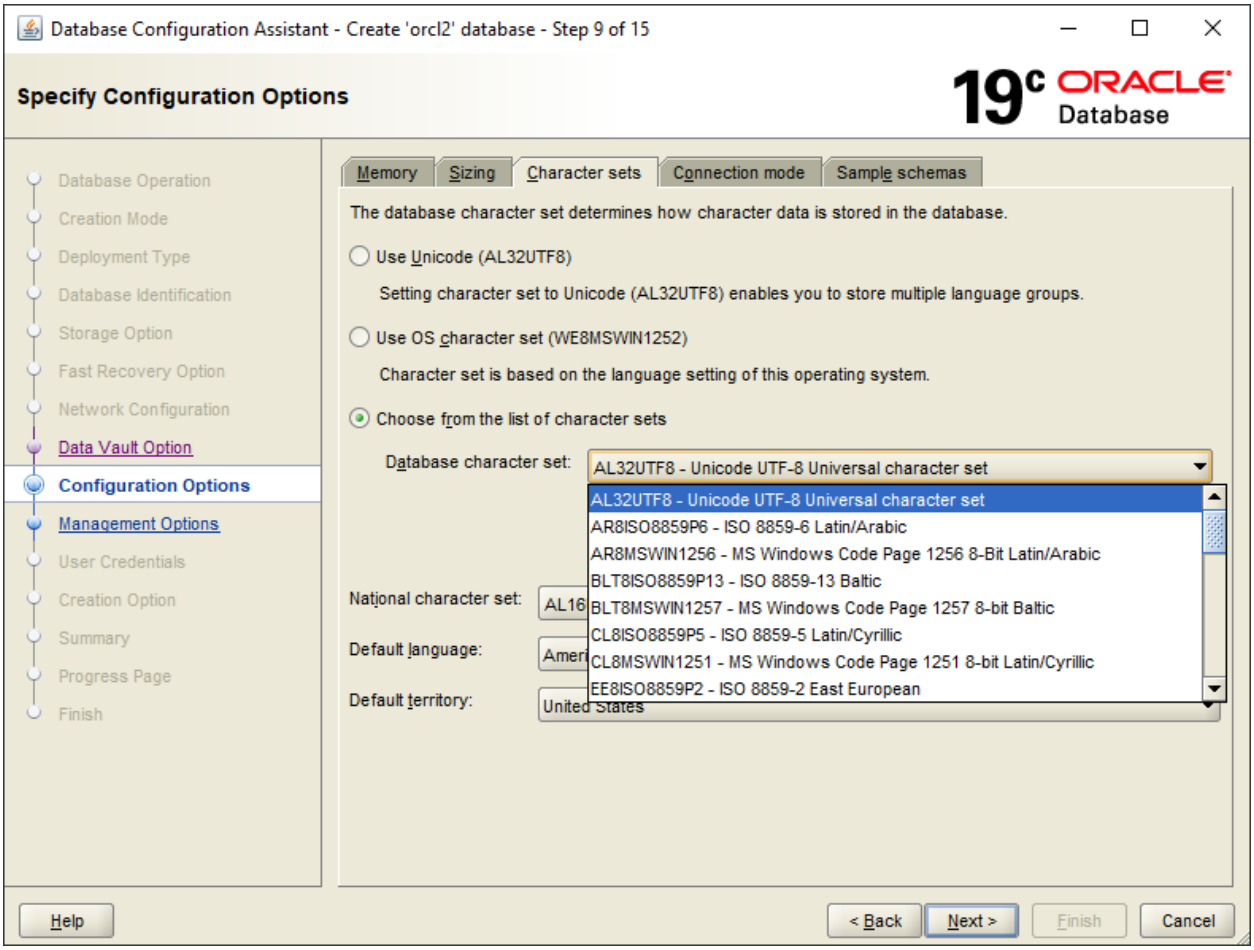
与此场景类似的，如遇到因ORA-01194 ORA-01110 ORA-01033 ORA-01115 ORA-00368 ORA-00600 kcbzib_kcrsds_1 ORA-00333 ORA-01113 ORA-01122 ORA-27027 等报错导致数据库无法打开的情况，均可以使用本恢复场景中所使用的方法尝试恢复数据。

其简要步骤如下：

1. 使用dbca创建新的ORACLE数据库，注意字符集要与损坏的数据库一致
2. 在新的数据库内创建对应数据库用户与表空间，建议暂时授权DBA角色给这些用户
3. 启动监听程序(LISTENER)，保证数据库服务已注册到监听
4. 启动DBRECOVER使用字典模式，并加载原损坏数据库的全部数据文件
5. 在DBRECOVER中点中要恢复的用户名，右键选择数据搭桥
6. 在数据搭桥界面点击加号图标，添加新数据库的连接信息(Connection)
7. 点击Data Bridge开始传输作业，等待SCHEMA下的所有表被传输到目标数据库的目标SCHEMA
8. 选中对应SCHEMA，右键选择EXPORTDDL导出DDL功能，选择所需恢复的对象类型后点击EXPORT
9. 基于EXPORTDDL生成的DDL SQL文件，手动在目标数据库的目标SCHEMA中执行


Database Configuration Assistant - Application - Step 1 of 14
Select Database Operation
19c ORACLE Database
Database Operation
Creation Mode
Deployment Type
Database Identification
Storage Option
Fast Recovery Option
Database Options
Configuration Options
Management Options
User Credentials
Creation Option
Summary
Progress Page
Finish
Select the operation that you want to perform.
Create a database
Configure an existing database
Delete database
Manage templates
Manage Pluggable databases
Oracle RAC database Instance management
Help
< Back
Next >
Finish
Cancel


Database Configuration Assistant - Create a database - Step 2 of 14
Select Database Creation Mode
19c ORACLE Database
Database Operation
Creation Mode
Deployment Type
Database Identification
Storage Option
Fast Recovery Option
Database Options
Configuration Options
Management Options
User Credentials
Creation Option
Summary
Progress Page
Finish
Typical configuration
Global database name:
orcl
Storage type:
File System
Database files location:
{ORACLE_BASE}\oradata\{DB_UNIQUE_NAME}
Browse...
Fast Recovery Area (FRA):
{ORACLE_BASE}\fast_recovery_area\{DB_UNIQUE_NAME}
Browse...
Database character set:
AL32UTF8 - Unicode UTF-8 Universal character set
Administrative password:
Confirm password:
Create as Container database
Pluggable database name:
Advanced configuration
Help
< Back
Next >
Finish
Cancel

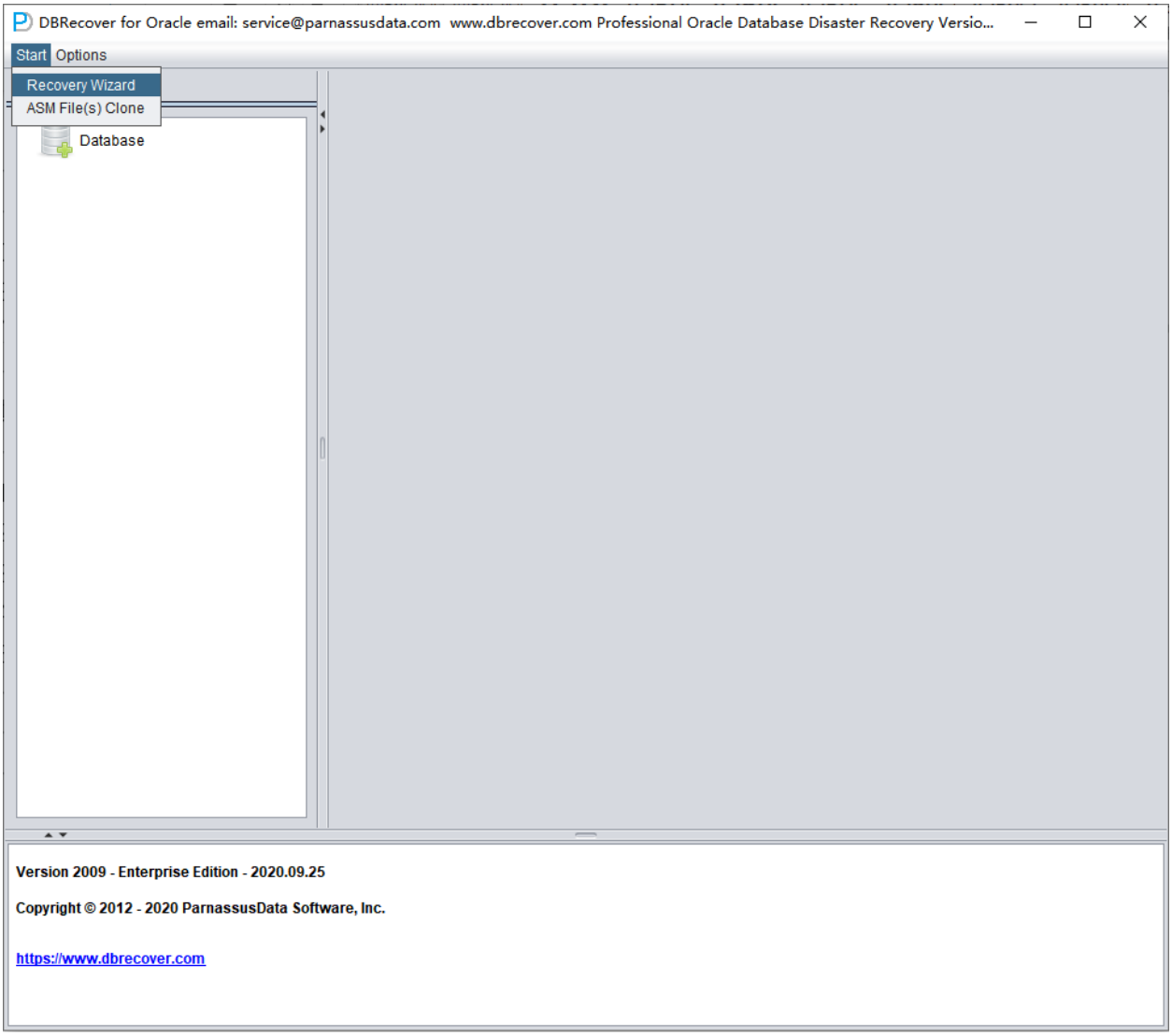


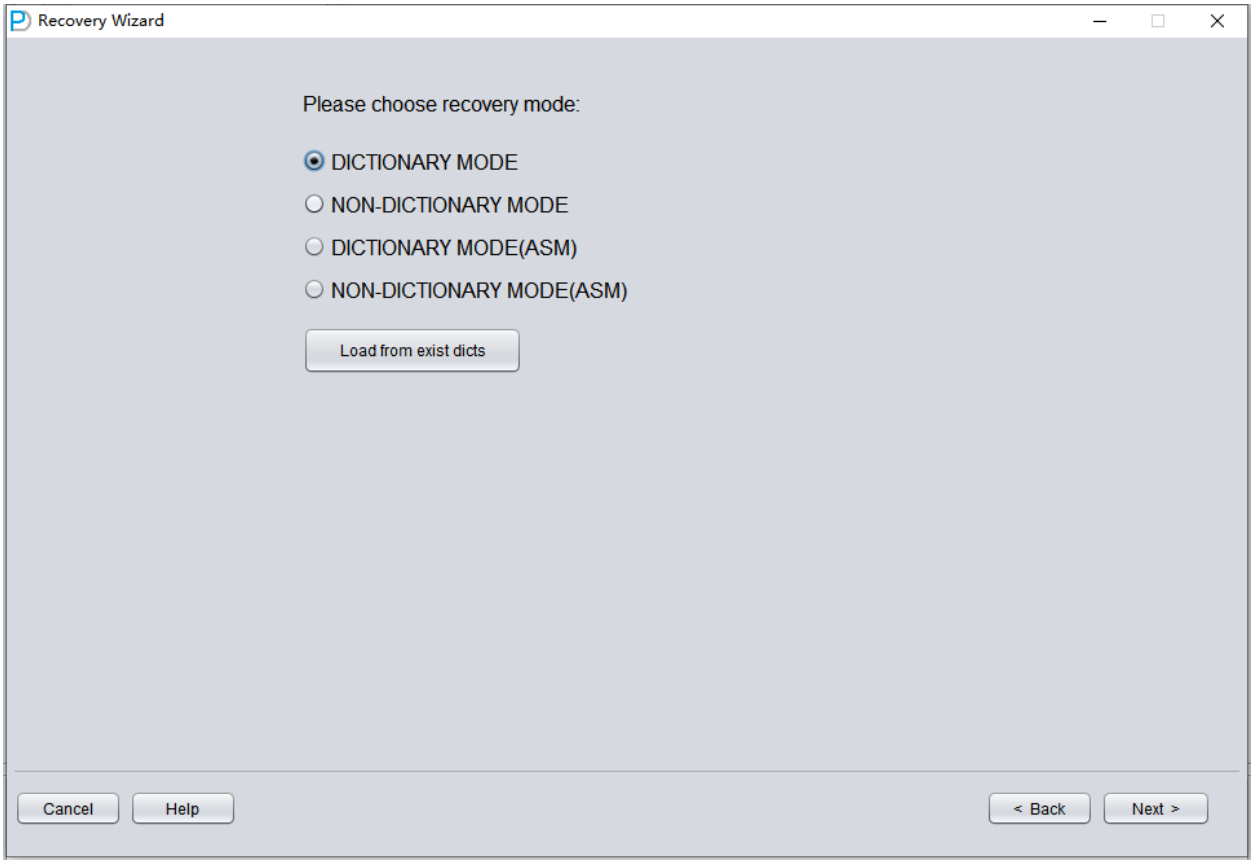

```
//启动监听程序(LISTENER)，保证数据库服务已注册到监听
C:\Users\testenv>lsnrctl status
LSNRCTL for 64-bit Windows: Version 11.2.0.1.0 - Production on 12-MAY-2023 10:01:48
Copyright (c) 1991, 2010, Oracle. All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=DESKTOP-testenv)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias LISTENER
Version TNSLSNR for 64-bit Windows: Version 11.2.0.1.0 - Production
Start Date 12-MAY-2023 10:00:49
Uptime 0 days 0 hr. 0 min. 59 sec
Trace Level off
Security ON: Local OS Authentication
SNMP OFF
```

```
Listener Parameter File D:\app\testenv\product\11.2.0\dbhome_2\network\admin\listener.ora
Listener Log File d:\app\testenv\diag\tnslsnr\DESKTOP-testenv\listener\alert\log.xml
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=DESKTOP-testenv)(PORT=1521)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\EXTPROC1521ipc)))
Services Summary...
Service "CLRExtProc" has 1 instance(s).
Instance "CLRExtProc", status UNKNOWN, has 1 handler(s) for this service...
Service "ORCL1XDB" has 1 instance(s).
Instance "orcl1", status READY, has 1 handler(s) for this service...
Service "ORCLXDB" has 1 instance(s).
Instance "orcl", status READY, has 1 handler(s) for this service...
Service "orcl" has 1 instance(s).
Instance "orcl", status READY, has 1 handler(s) for this service...
Service "orcl1" has 1 instance(s).
Instance "orcl1", status READY, has 1 handler(s) for this service...
The command completed successfully
//在新的数据库内创建对应数据库用户与表空间，建议暂时授权DBA角色给这些用户
set ORACLE_SID=ORCL1
sqlplus / as sysdba
SQL> create user pd identified by oracle;
User created.
SQL> grant dba to pd;
Grant succeeded.
SQL> create tablespace pdtbs datafile size 500M autoextend on next 100M;
Tablespace created.
SQL> alter user pd default tablespace pdtbs;
User altered.
```

启动DBRECOVER，并选择 Tools => Recovery Wizard

点击Next


DBRecover for Oracle email: service@parnassusdata.com www.dbrecover.com Professional Oracle Database Disaster Recovery Versio...
Start
Options
Recovery Wizard
ASM File(s) Clone
Database
Version 2009 - Enterprise Edition - 2020.09.25
Copyright © 2012 - 2020 ParnassusData Software, Inc.
https://www.dbrecover.com



下一步骤我们要选择正确的ENDIAN字节序；由于ORACLE数据文件在不同的操作系统平台上采用了不同的Endian字节序格式，字节序和平台对应列表如下：

platform	endian
Solaris[tm] OE (32-bit)	Big
Solaris[tm] OE (64-bit)	Big
Microsoft Windows IA (32-bit)	Little
Linux IA (32-bit)	Little
AIX-Based Systems (64-bit)	Big
HP-UX (64-bit)	Big
HP Tru64 UNIX	Little
HP-UX IA (64-bit)	Big
Linux IA (64-bit)	Little

HP Open VMS	Little
Microsoft Windows IA (64-bit)	Little
IBM zSeries Based Linux	Big
Linux x86 64-bit	Little
Apple Mac OS	Big
Microsoft Windows x86 64-bit	Little
Solaris Operating System (x86)	Little
IBM Power Based Linux	Big
HP IA Open VMS	Little
Solaris Operating System (x86-64)	Little
Apple Mac OS (x86-64)	Little

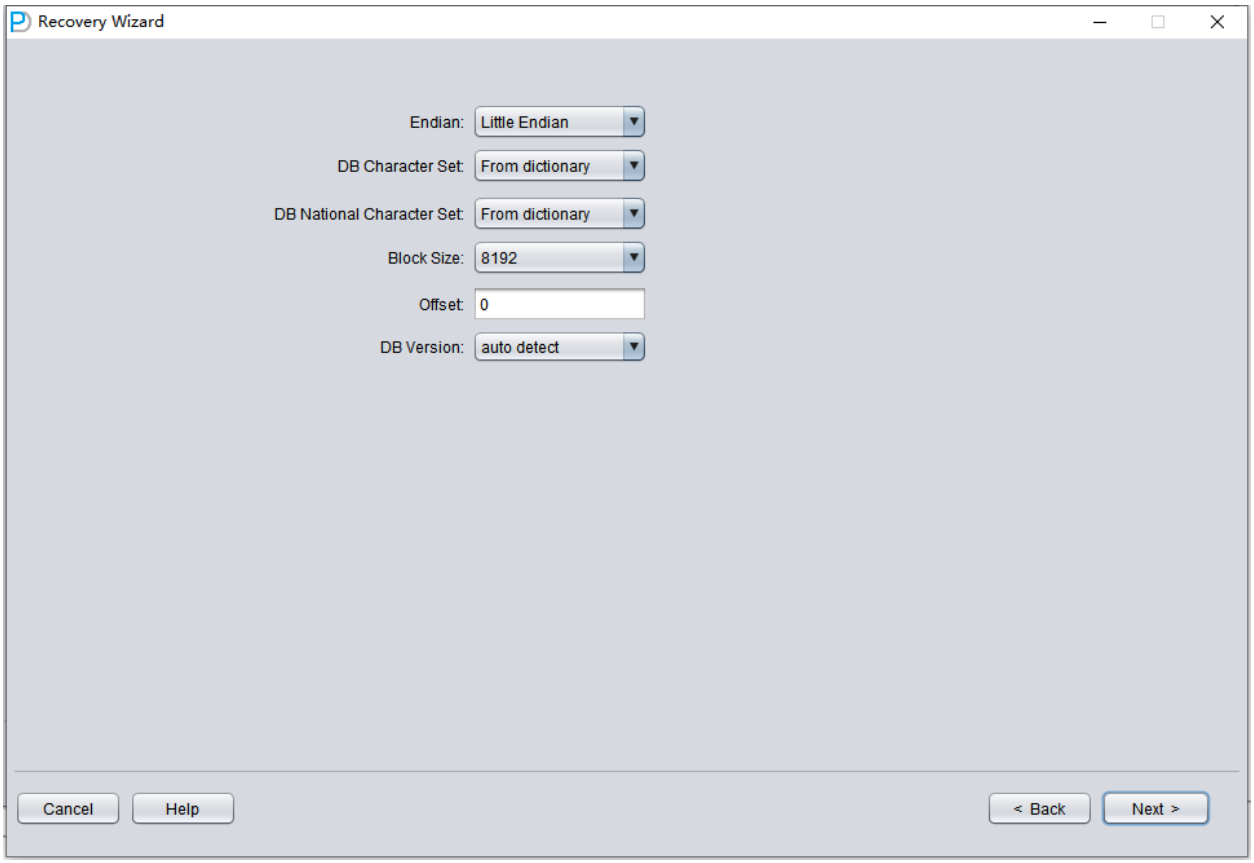
只需要注意我们最常用的Windows和Linux平台均是Little Endian，即不需要做任何设置保持默认即可。

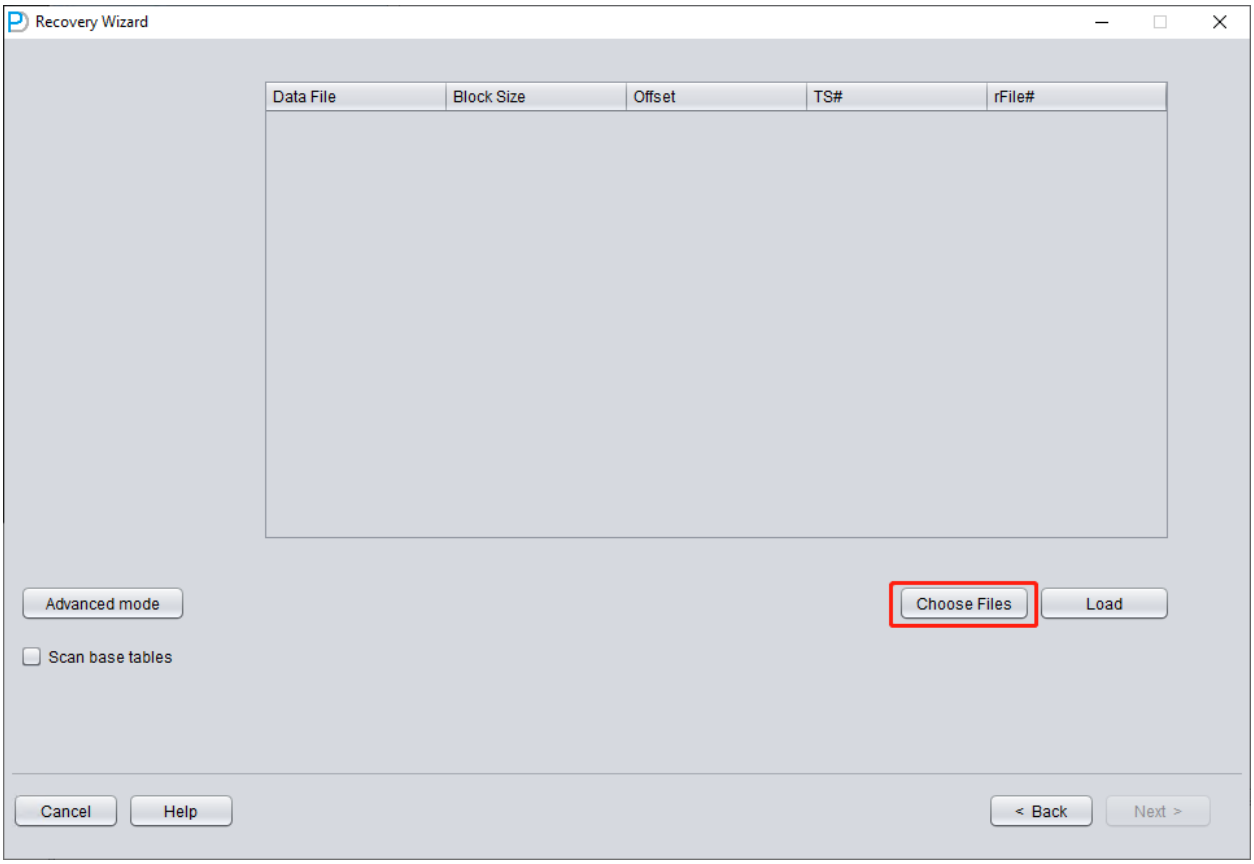
而在小型机平台上，包括AIX-Based Systems (64-bit) 、HP-UX (64-bit) 上使用的是Big Endian大端字节序，在这里要选为Big Endian。

注意事项： 如果你的数据文件是在AIX(即Big Endian的)上生成的，你为了方便而将这些数据文件拷贝到Windows服务器上并使用DBRECOVER来恢复数据，那么你仍应当选择其原生的Big Endian格式。

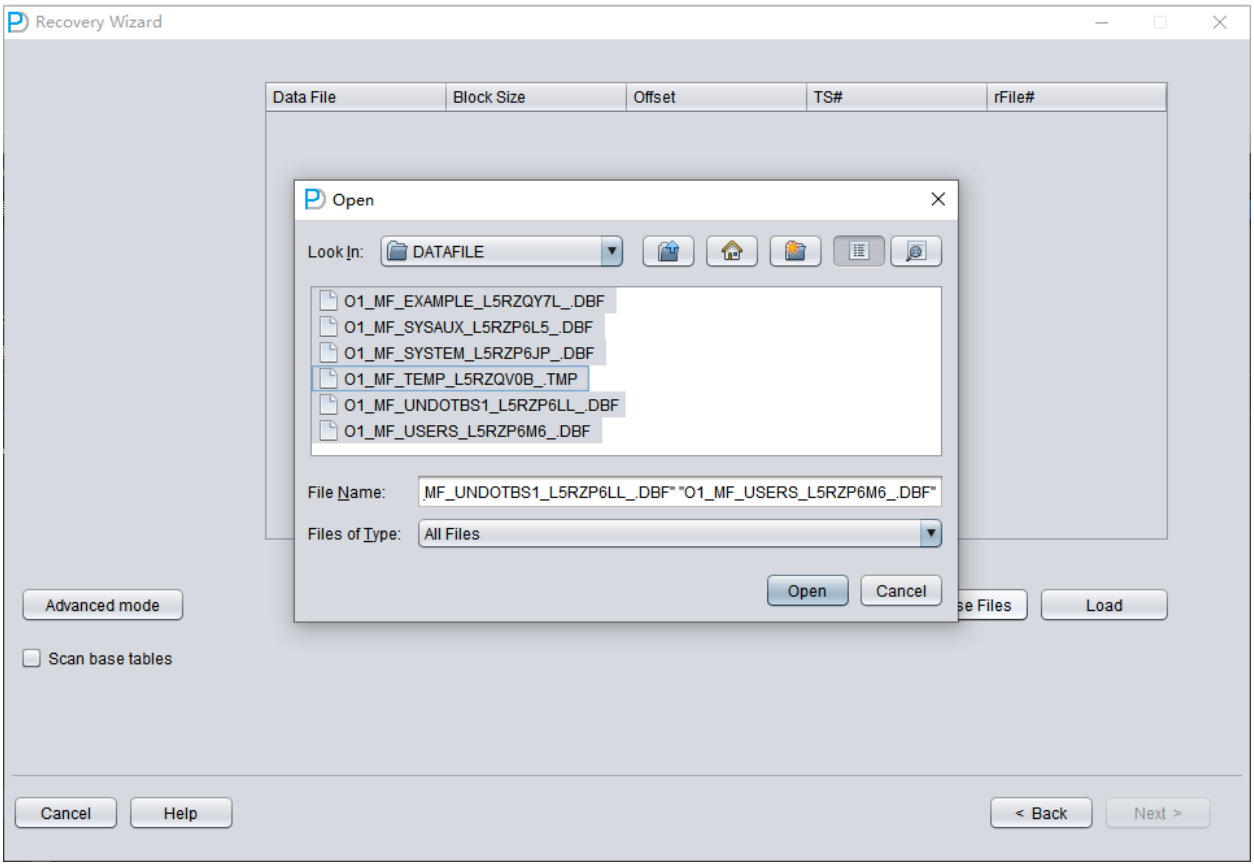
这里由于我们要恢复的是Linux x86-64平台上的Oracle数据库文件，所以我们选择Endian为Little。

点击Next

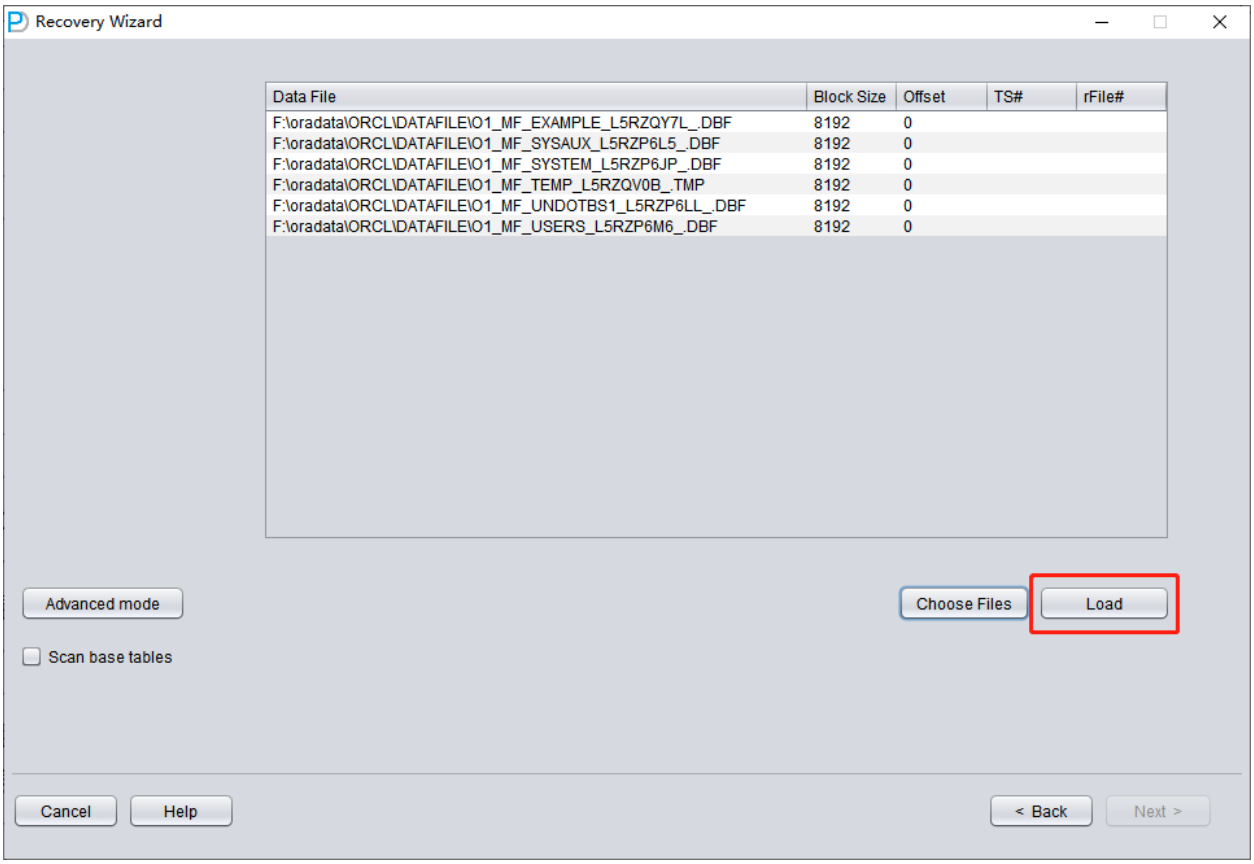

Recovery Wizard
Endian:
Little Endian
DB Character Set:
From dictionary
DB National Character Set:
From dictionary
Block Size:
8192
Offset:
0
DB Version:
auto detect
Cancel
Help
< Back
Next >



点击Choose Files，一般我们推荐如果数据库不大，那么将该库所有的数据文件都选择进来；如果你的数据库很大，且你了解你的数据表位于哪些数据文件上，则你可以仅仅选择SYSTEM表空间的数据文件(必须！)以及数据所在的数据文件。



注意Choose界面支持Ctrl + A 和Shift等键盘操作。



注意：添加完所有数据文件后，此界面上的其他参数若你不了解，那么就都保持默认，无需修改！

之后需要为指定的数据文件指定其Block Size即ORACLE数据块的大小，这里根据实际情况修改即可， 例如你的DB_BLOCK_SIZE是8K，但是部分表空间指定16K作为数据块大小的，仅仅需要为那些不是8k的数据文件修改BLOCK_SIZE即可。

使用普通文件系统的情况不需要在这里指定OFFSET，OFFSET 参数主要是为了那些采用裸设备存放数据文件的场景，例如在AIX上基于普通VG的LV作为数据文件，则存在4k的OFFSET，需要在此处指定。

如果你恰巧正在使用裸设备数据文件，而又不知道OFFSET到底是多少？ 则可以使用$ORACLE_HOME/bin下自带的dbfsize工具查看:

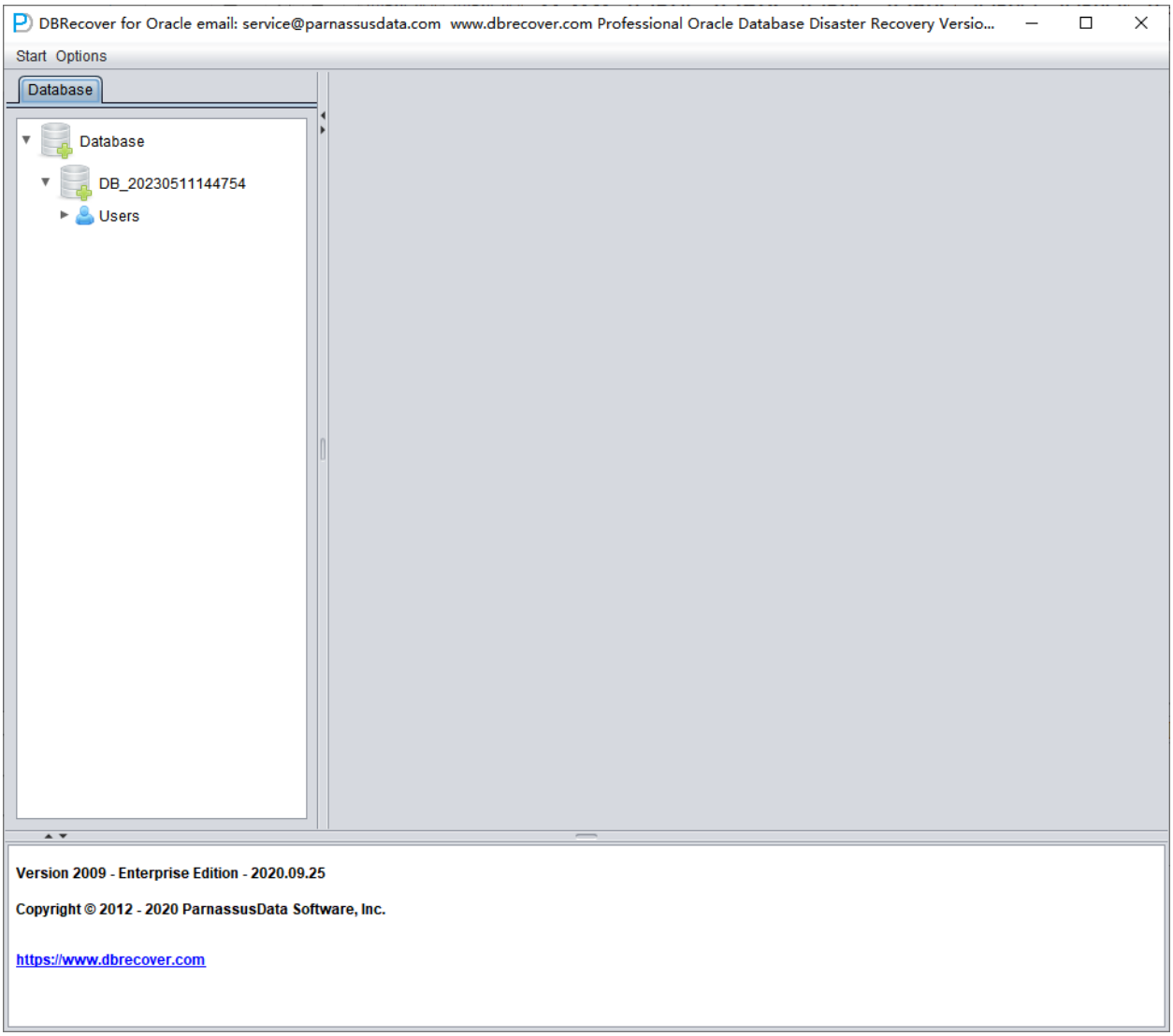
```
$ dbfsize /dev/lv_control_01
Database file: /dev/lv_control_01
Database file type: raw device without 4K starting offset
```

```
Database file size: 334 16384 byte blocks
```

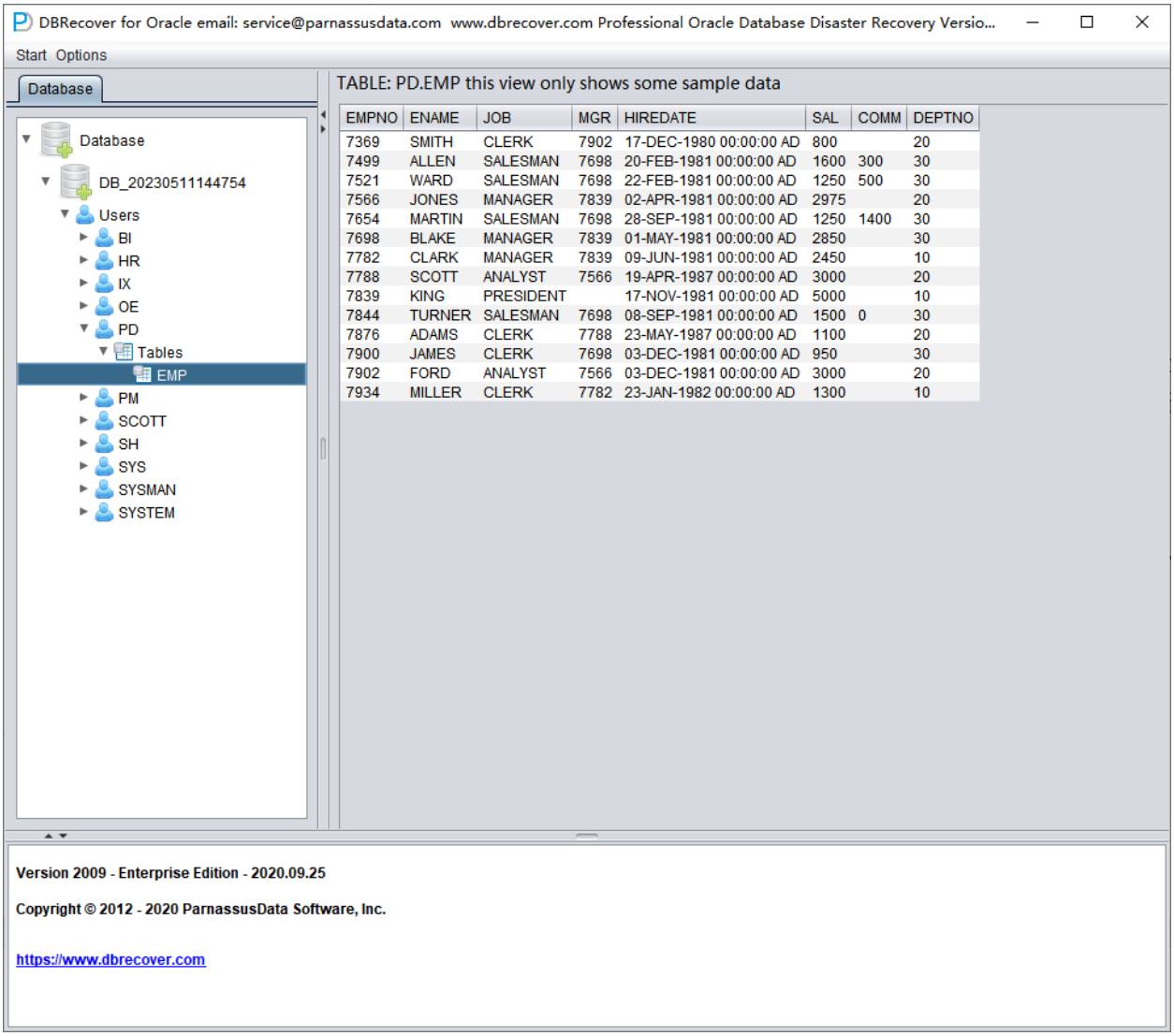
由于此场景中所有数据文件均为8K的BLOCK SIZE，且基于文件系统所以均没有OFFSET，点击Load

Load阶段DBRECOVER会从SYSTEM表空间中读取ORACLE数据字典信息，并在自带的Derby中自建一个数据字典，这让DBRECOVER有能力分析ORACLE数据库中的各种数据。

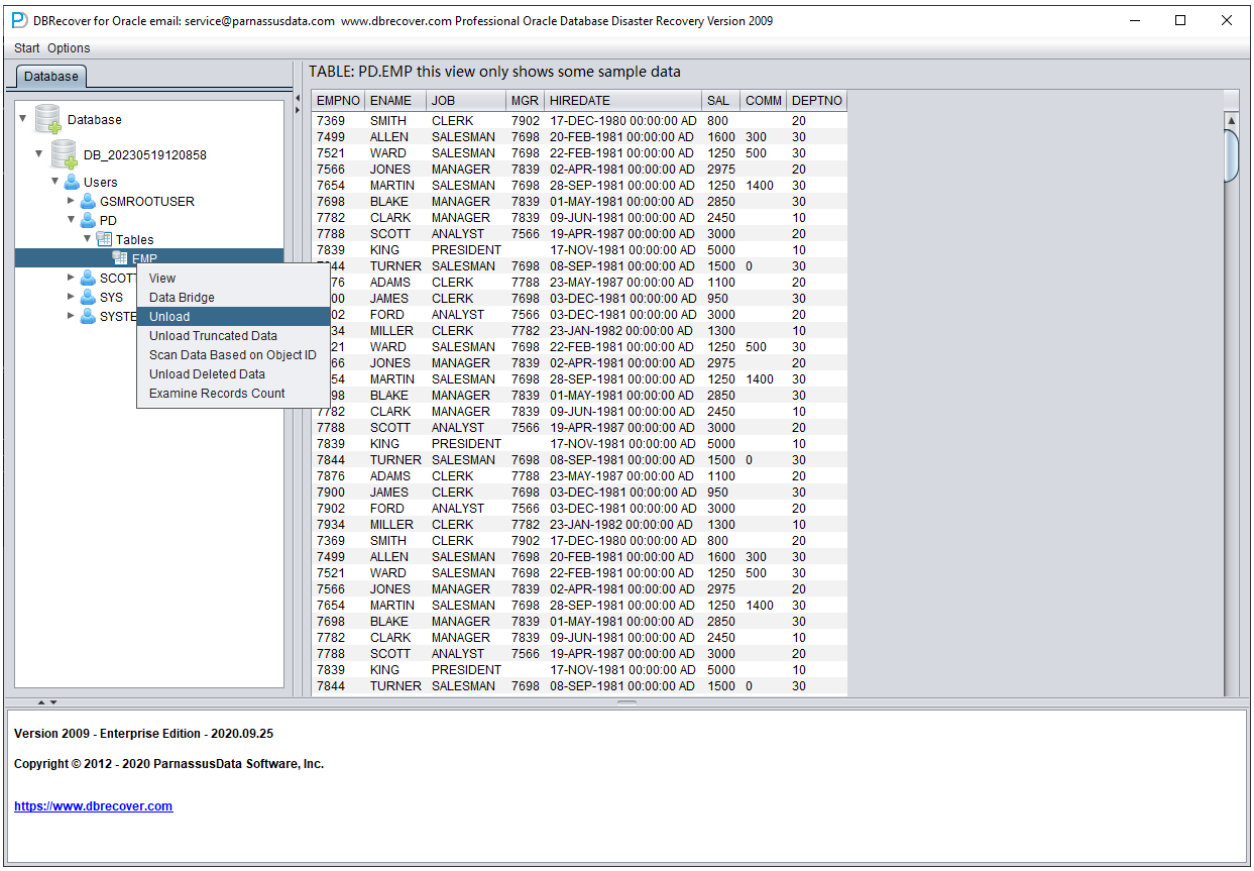
Load完成后DBRECOVER界面左侧出现按照数据库用户分组的树形图：



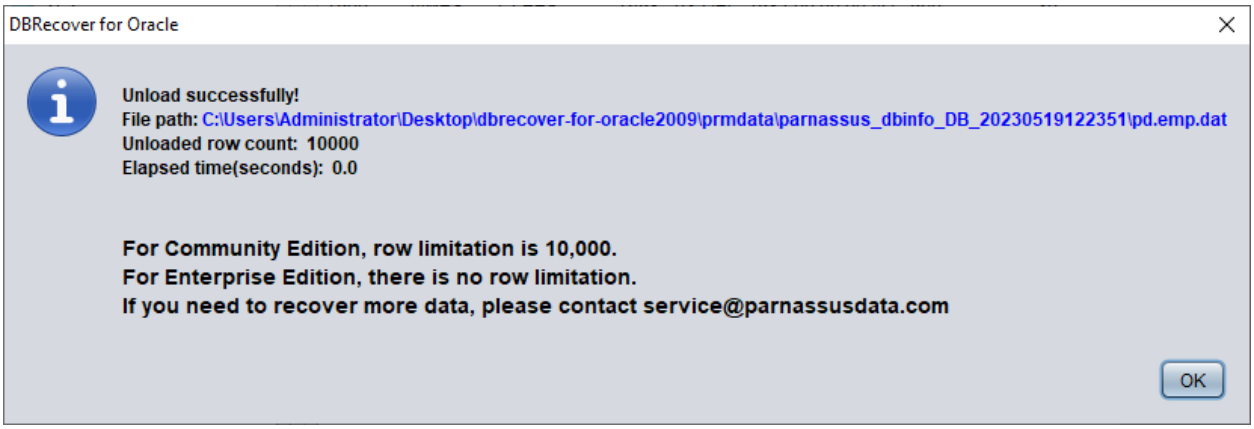
点选一张你要恢复的表，双击查看数据：



在尚未购买软件许可证的情况下，我们可以通过查看数据表、抽取至少一万行数据和检验可恢复行数的方法来考察DBRECOVER是否能够恢复足够多的数据。

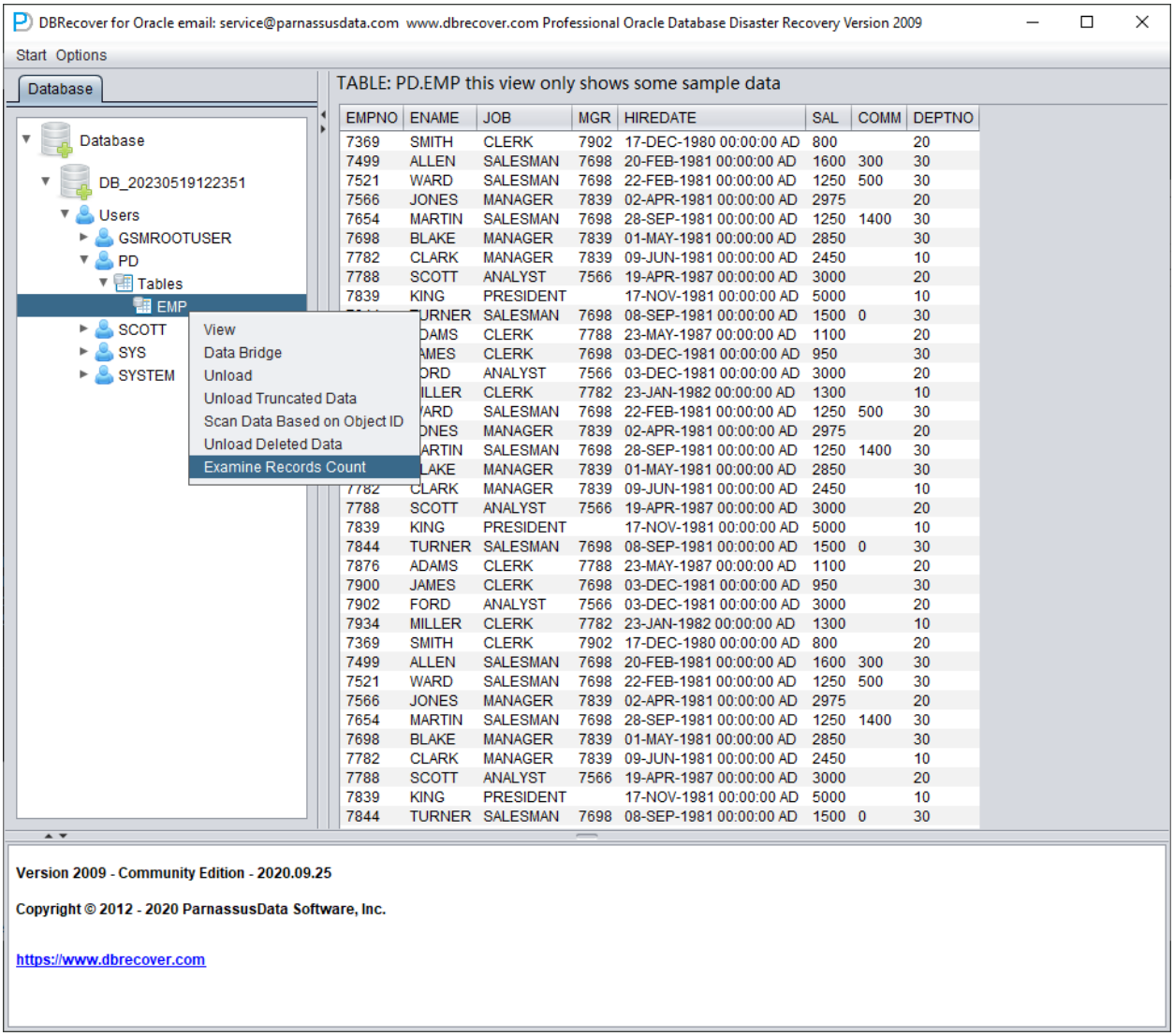


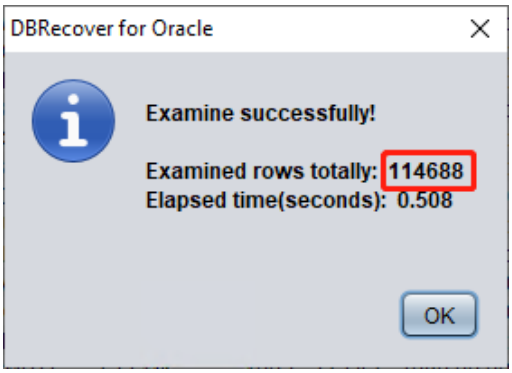
选中表之后右键UNLOAD，会将表数据导出为文本格式：



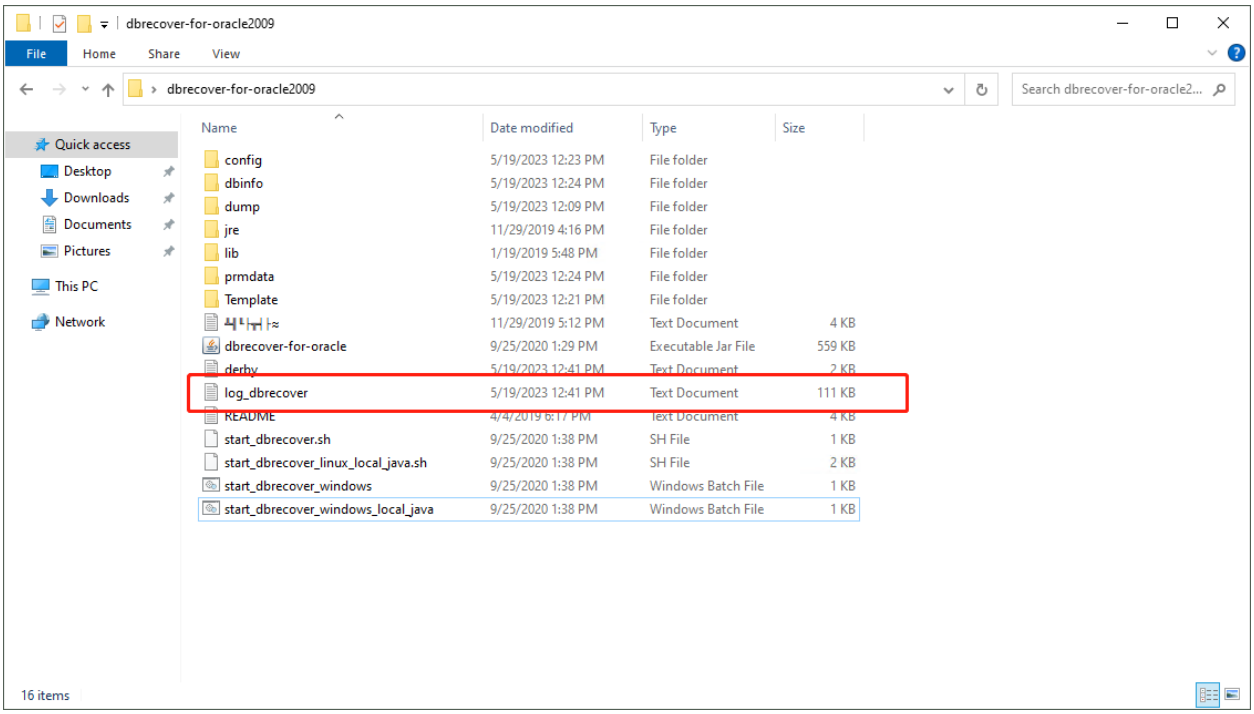
在没有注册软件许可证的情况下，单张表最多可以抽取一万行数据。

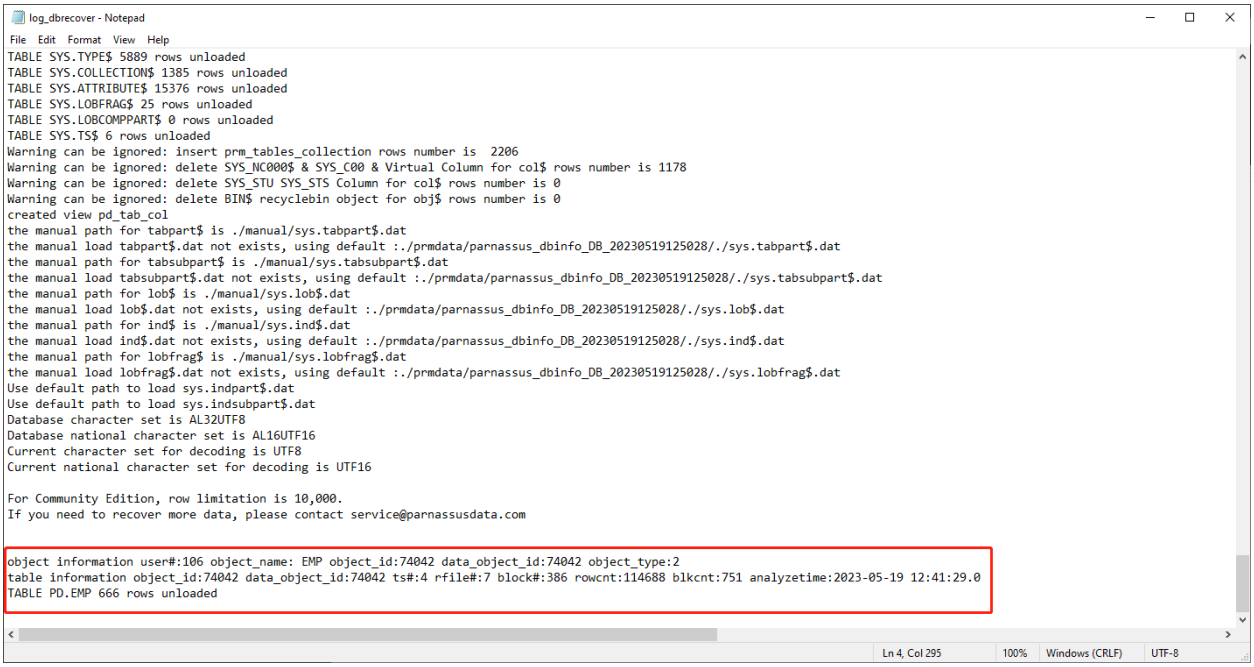
对于存放超过一万行数据的表，可以通过检验可恢复行数功能来进一步验证。选中要检查的表，右键EXAMINE RECORDS COUNT：





Oracle从10g开始引入了自动收集统计信息作业的特性，利用该特性我们可以查看表的历史统计信息，其中包括行数。我们在字典模式下，每对一张表执行查看、抽取、检查等操作时该表的一些信息都会记录到软件日志log_dbrecover.txt中。在日志文件存放于软件目录下：





日志信息：

```
object information user#:106 object_name: EMP object_id:74042 data_object_id:74042 object_type:2
table information object_id:74042 data_object_id:74042 ts#:4 rfile#:7 block#:386 rowcnt:114688 blkcnt:751 analyzetime:2023-05-19 12:41:29.0
TABLE PD.EMP 666 rows unloaded
```

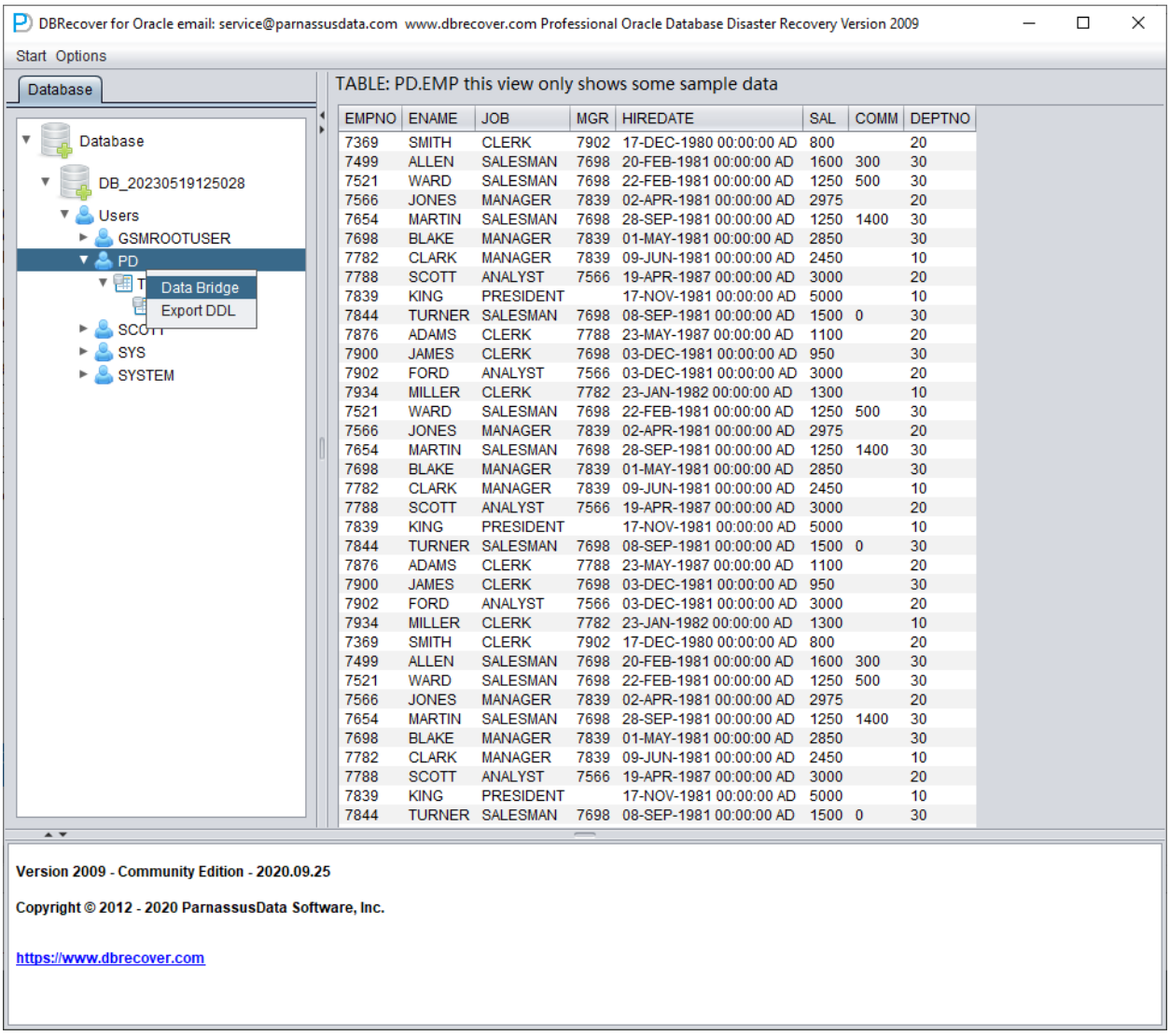
日志中出现了很多有用的信息：

object_id对象号	74042
data_object_id数据对象号	74042
ts#表空间号	4
rfile#表头所在相对文件号	7
block#表头所在数据块号	386
rowcnt统计信息中记录的行数（统计信息是估算值）	114688
blkcnt该表的总块数	751
analyzetime统计信息收集的时间	2023-05-19 12:41:29.0

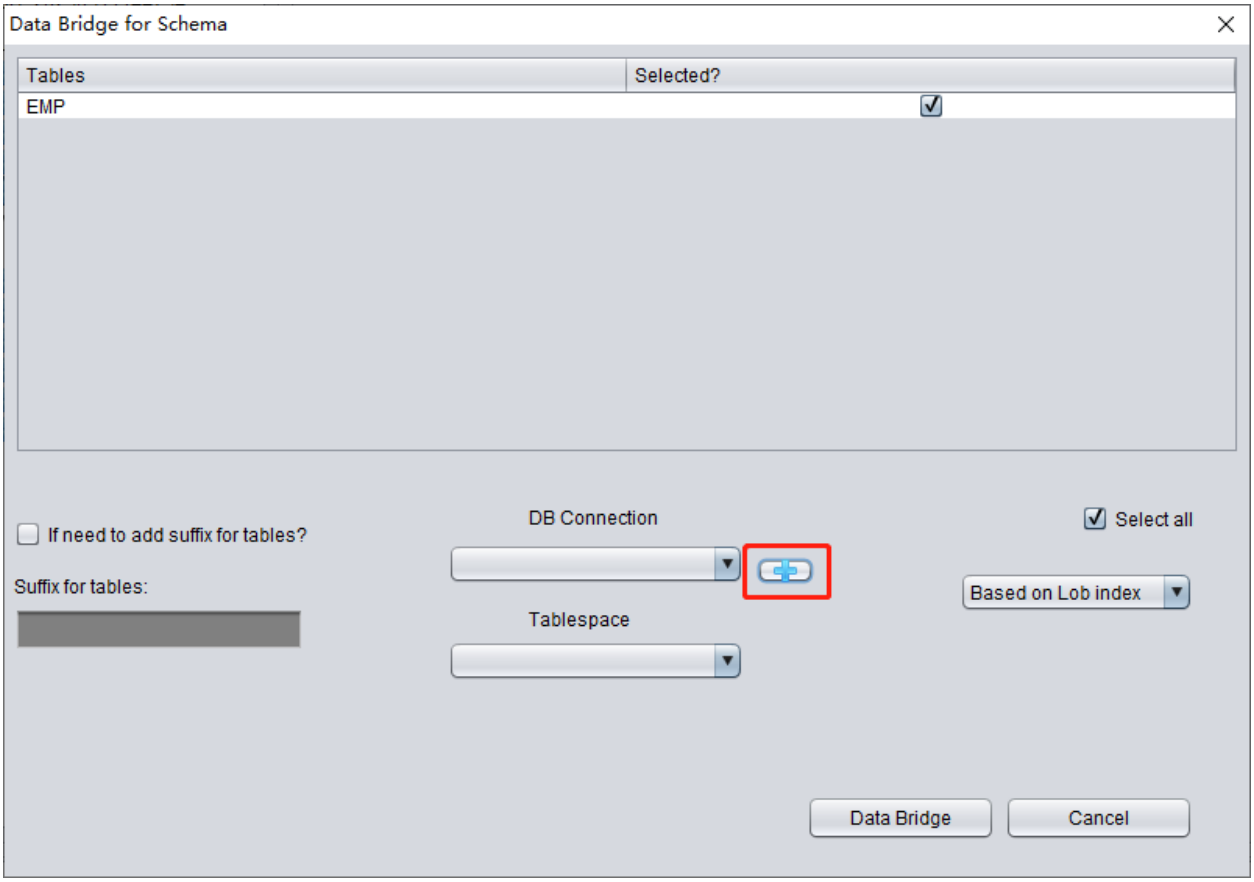
一般来说统计信息的误差不超过10%，因此我们可以基于这里的rowcnt对比检验行数的结果。例如这里的rowcnt是114688（对于小于100万行的表而言，统计信息误差很小），而EXAMINE的结果是114688行，则可以验证该结果的真实性。

用户可以基于自身需求，对每一张重要的数据表做上述检验。我们建议用户在购买软件许可证前充分检验可恢复数据的完整性。

完成上述检验后我们开始SCHEMA用户级别的数据搭桥传输，选中要恢复的用户名字右键Data Bridge



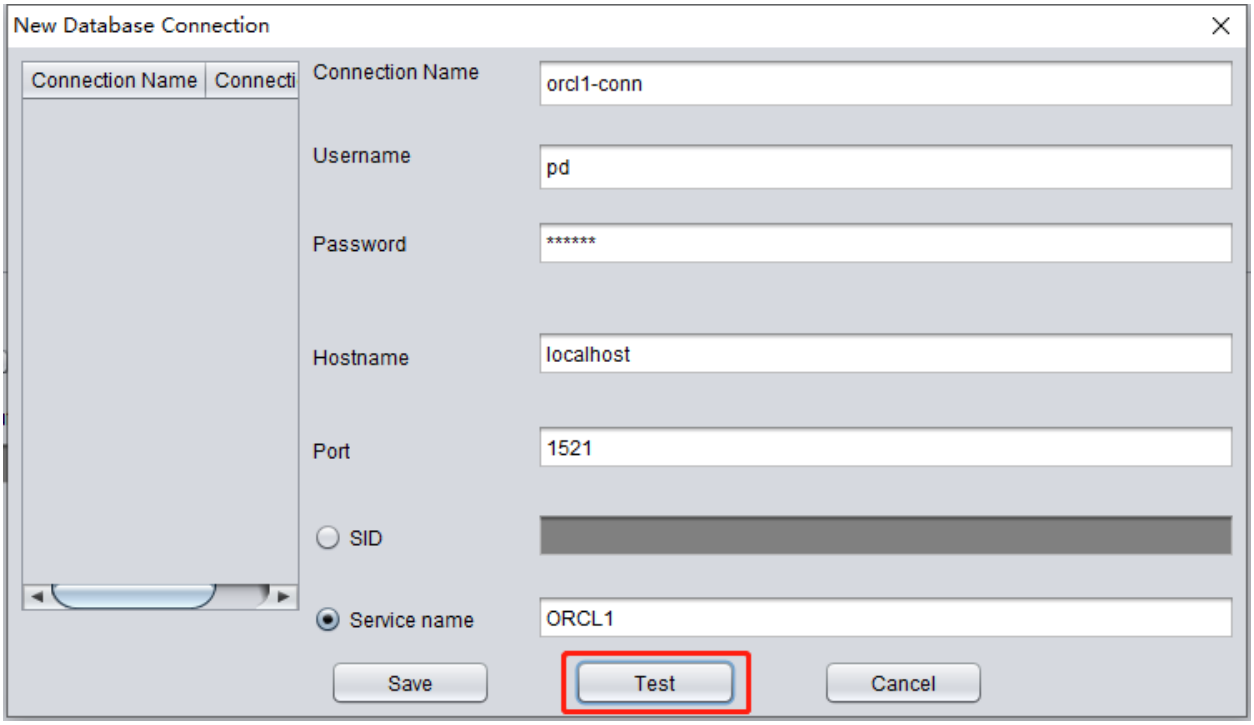
在SCHEMA级别数据搭桥界面点选"+"号按钮，添加目标数据库链接信息：

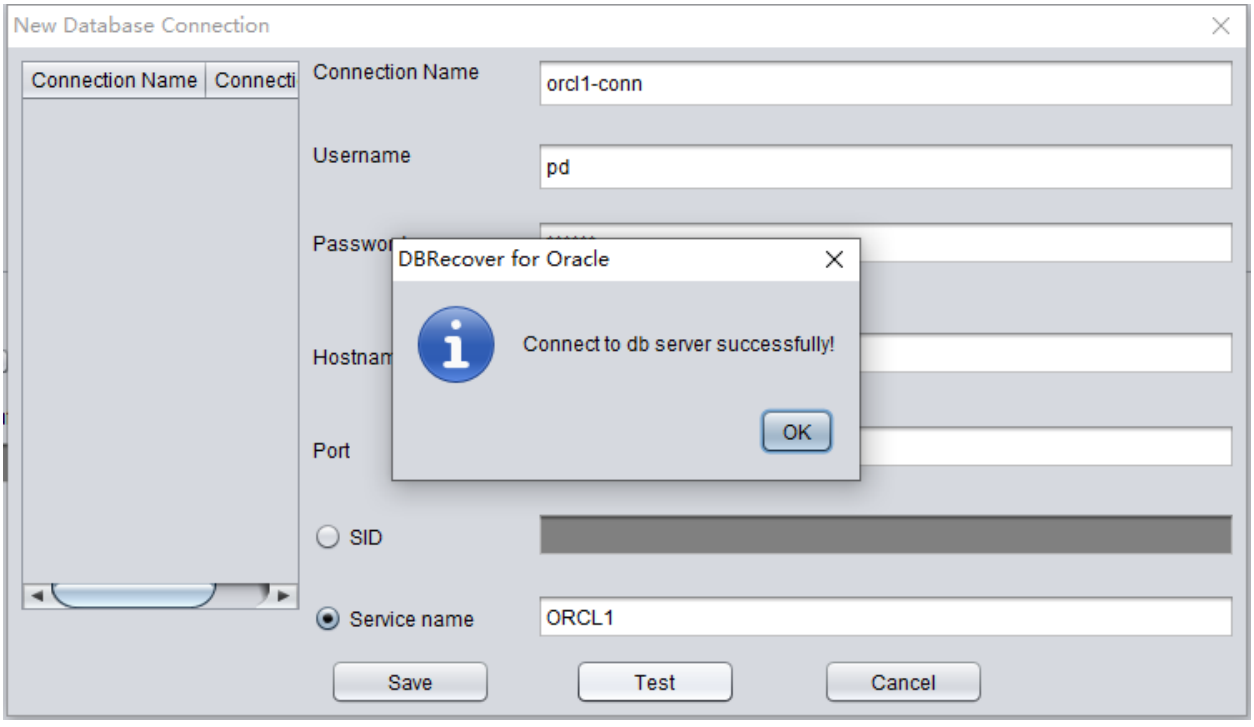


输入之前创建的新库的链接信息，这里使用了 PD用户。

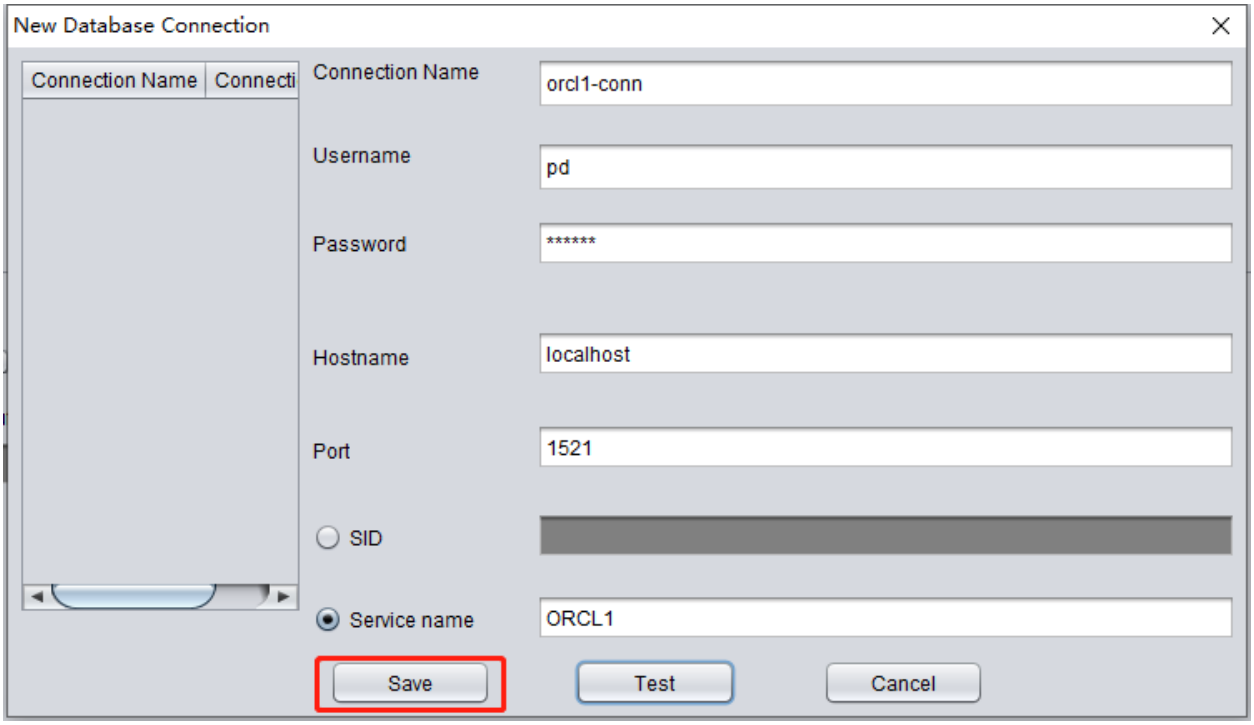
注意：DBRECOVER软件只会将数据传输到数据库链接信息中的用户下，即这里填入PD就会将数据传输到PD下。客户在这里应当基于简单的一对一原则，即若有一个要恢复的数据库用户例如EAS，则要在目标数据库创建一个EAS用户及其表空间并赋予必要权限（DBA角色），并在此数据库链接中填入EAS，以确保数据传输到EAS下，此处的PD仅为一个示例。客户若要恢复多个数据库用户，例如 EAS、MES、NC001等，则要一一对应地在目标数据库中创建上述这些账户及其表空间，并赋予必要权限（DBA角色），之后在DBRECVOER中创建多个数据库链接信息（DB Connection），在具体某个用户SCHEMA传输时指定对应的数据库链接信息（DB Connection）。

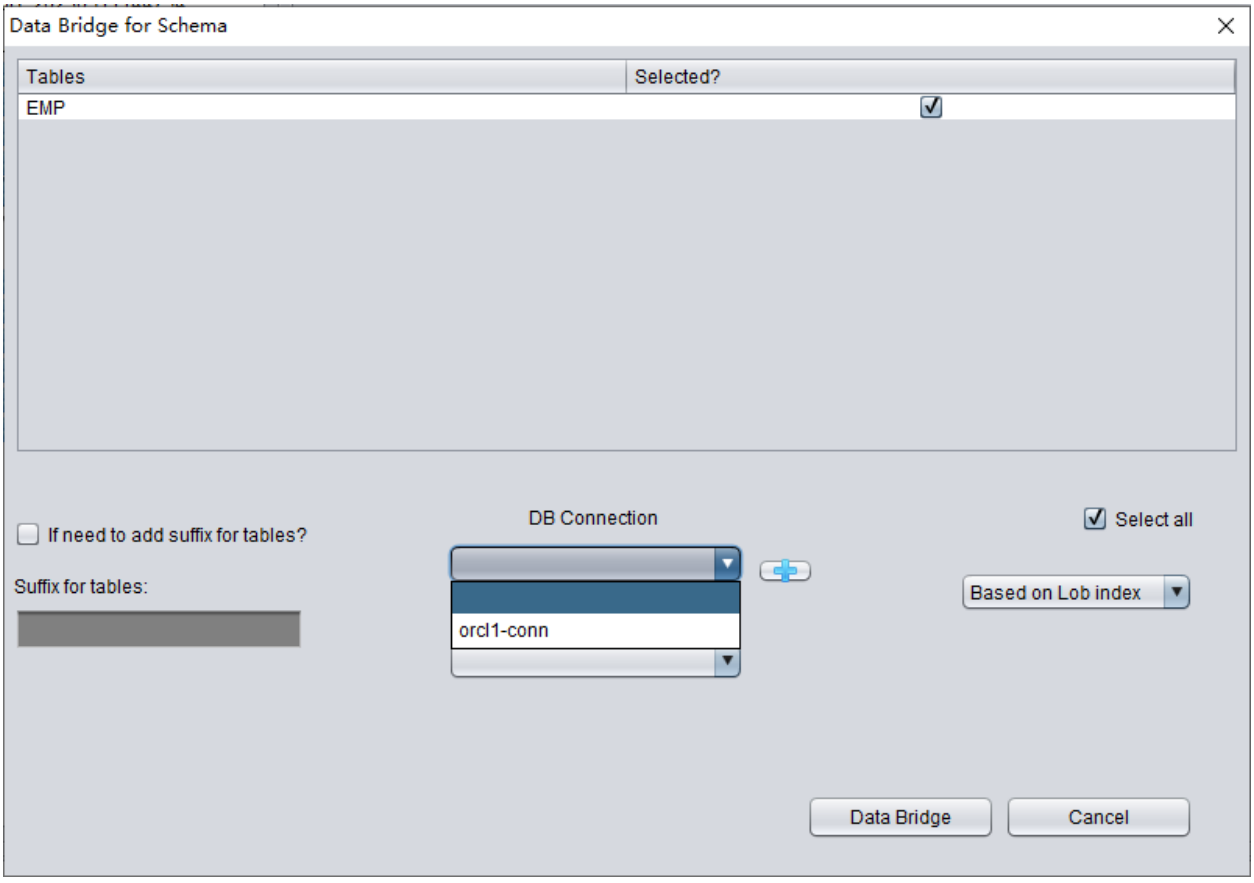
点击TEST测试目标数据库链接可用性：

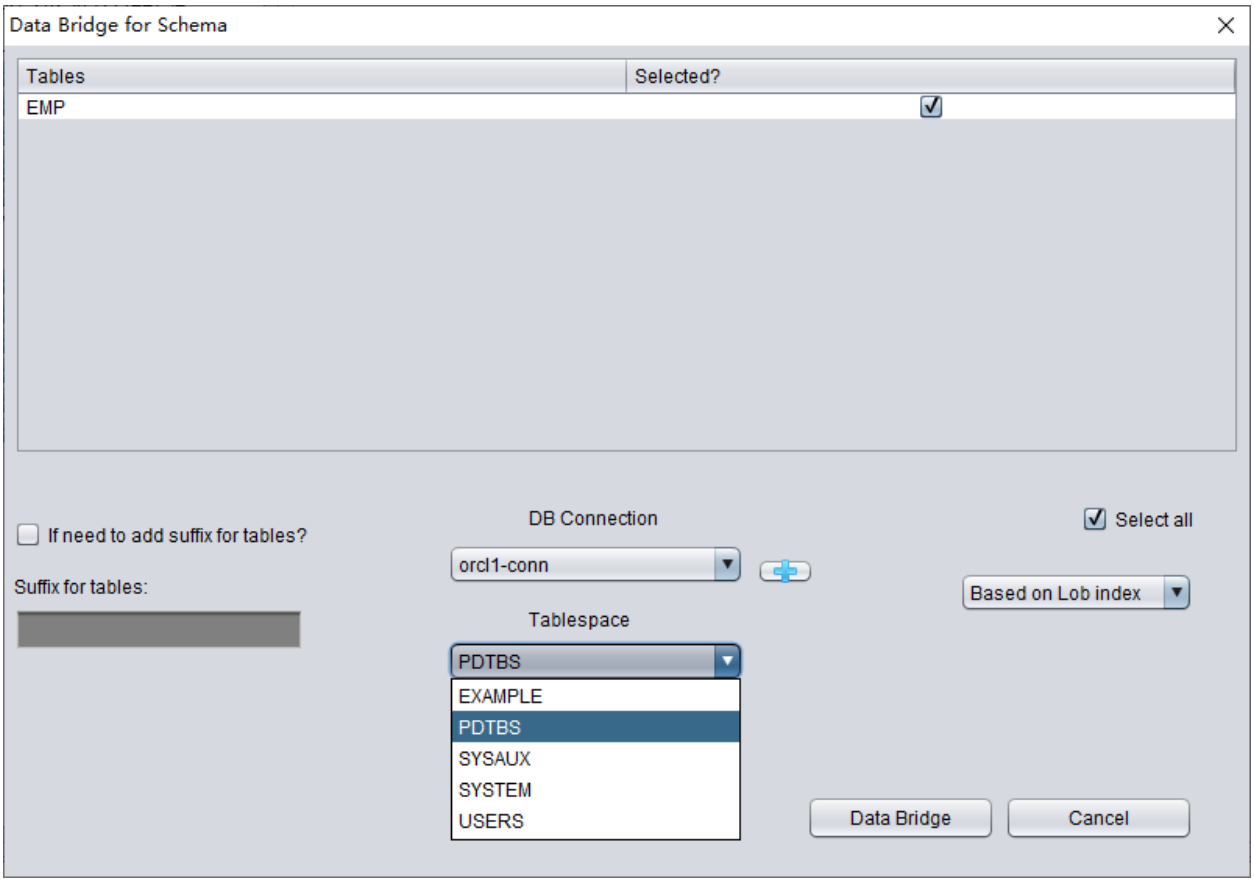


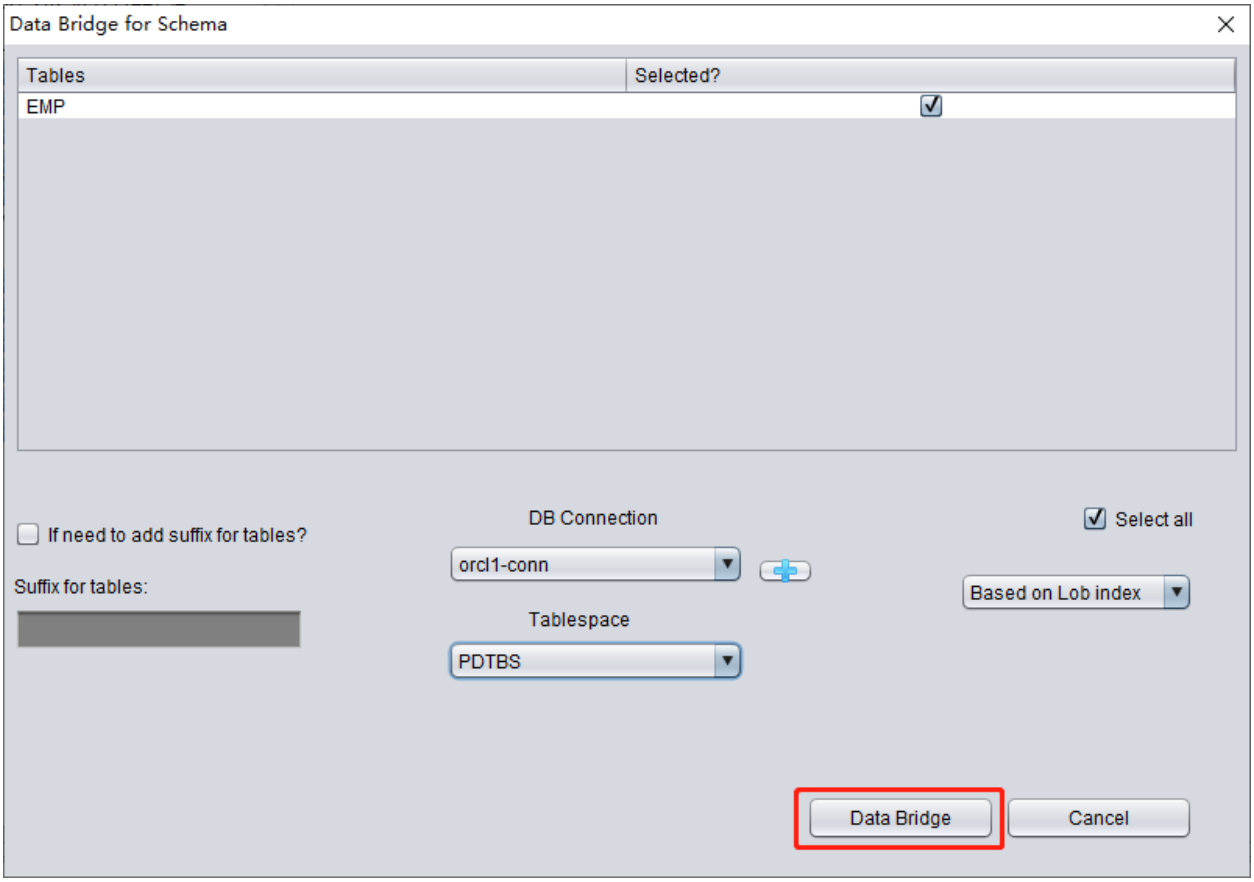


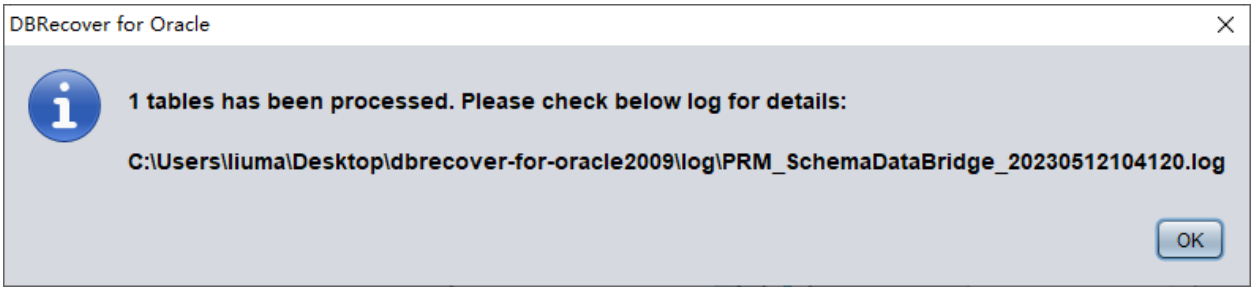
若成功则点击SAVE保存：


New Database Connection
Connection Name
Connecti
Connection Name
orcl1-conn
Username
pd
Password
******
Hostname
localhost
Port
1521
SID
Service name
ORCL1
Save
Test
Cancel


Data Bridge for Schema
Tables
Selected?
EMP
If need to add suffix for tables?
Suffix for tables:
DB Connection
orcl1-conn
Select all
Based on Lob index
Data Bridge
Cancel


Data Bridge for Schema
Tables
Selected?
EMP
If need to add suffix for tables?
Suffix for tables:
DB Connection
orcl1-conn
Tablespace
PDTBS
EXAMPLE
PDTBS
SYSAUX
SYSTEM
USERS
Select all
Based on Lob index
Data Bridge
Cancel





检查数据：

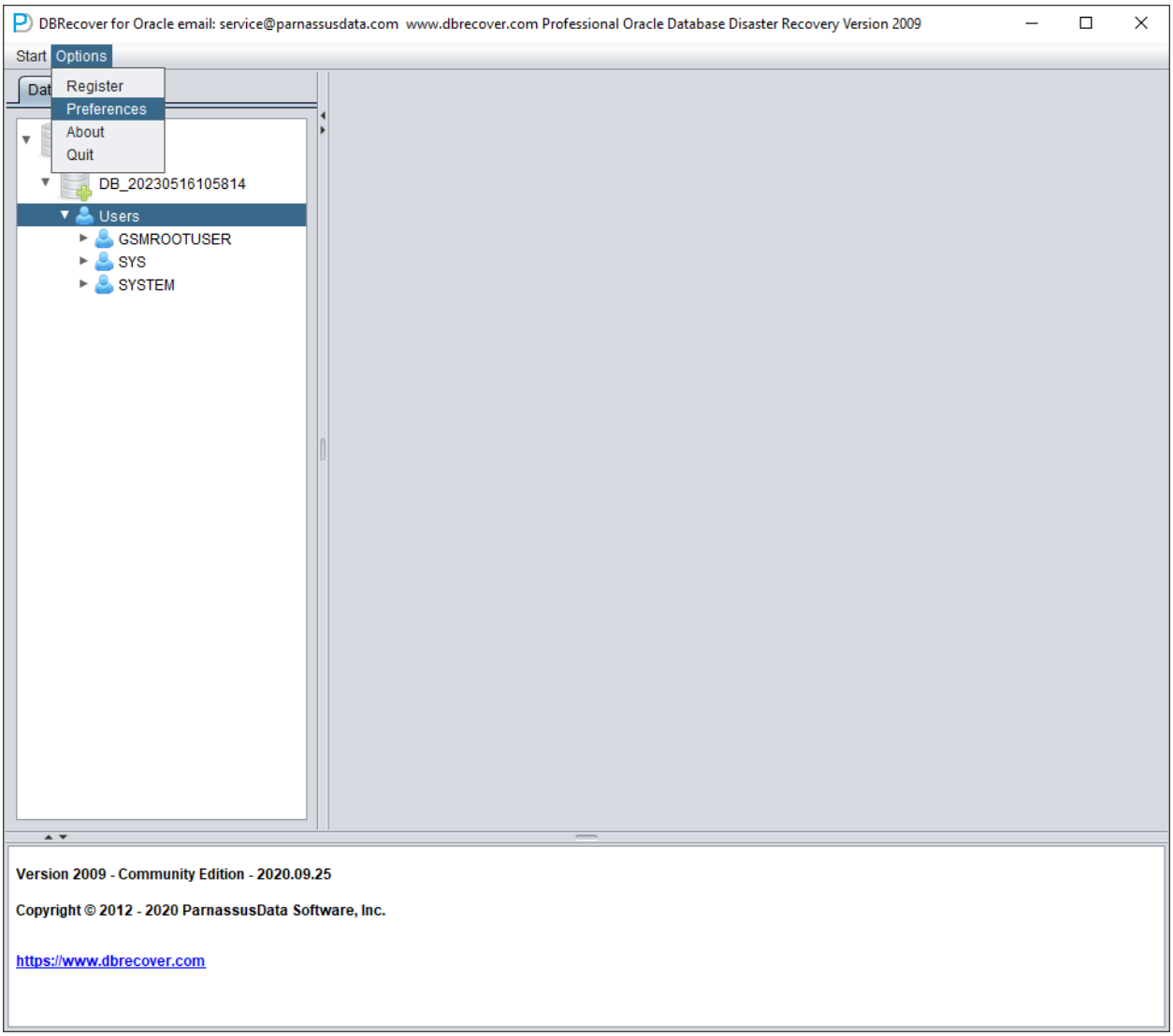
```
SQL> show parameter db_name
NAME TYPE VALUE
---------------------------------- --------------------- ------------------------------
db_name string ORCL1
SQL> select count(*) from pd.emp;
COUNT(*)
```

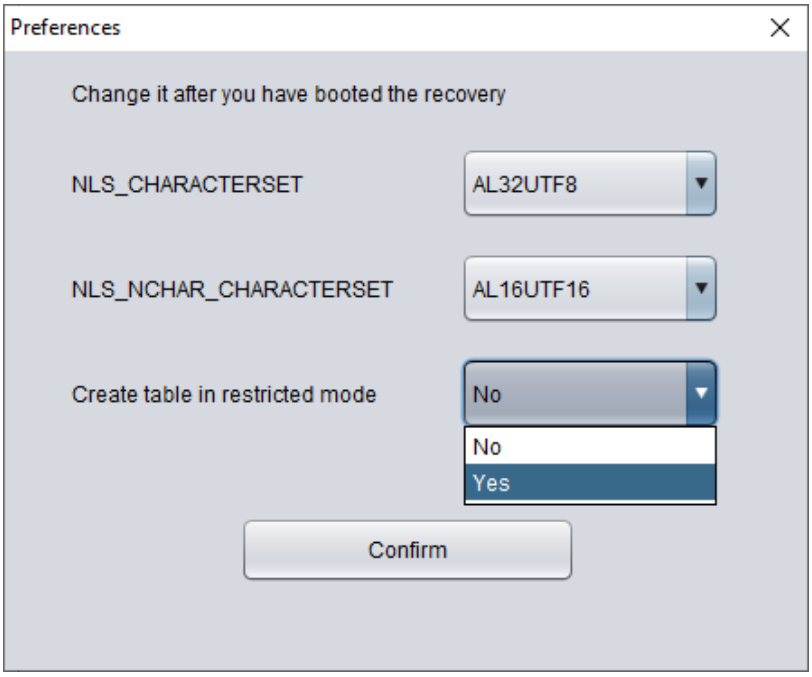
```
---------
14
```

WIDE TABLE宽表模式介绍

注意：以上数据搭桥默认采用宽表模式(wide table mode)传输数据，即默认将所有CHAR，NCHAR，VARCHAR，NVARCHAR的字段类型转换为该类型的最长长度，即2000或4000。这样做的目的是避免实际可能发生的因字段过短导致无法顺利插入恢复字符串的问题。

如果不希望使用宽表模式，可以点击菜单栏 Options => Preferences


DBRecover for Oracle email: service@parnassusdata.com www.dbrecover.com Professional Oracle Database Disaster Recovery Version 2009
Start
Options
Register
Preferences
About
Quit
DB_20230516105814
Users
GSMROOTUSER
SYS
SYSTEM
Version 2009 - Community Edition - 2020.09.25
Copyright © 2012 - 2020 ParnassusData Software, Inc.
https://www.dbrecover.com



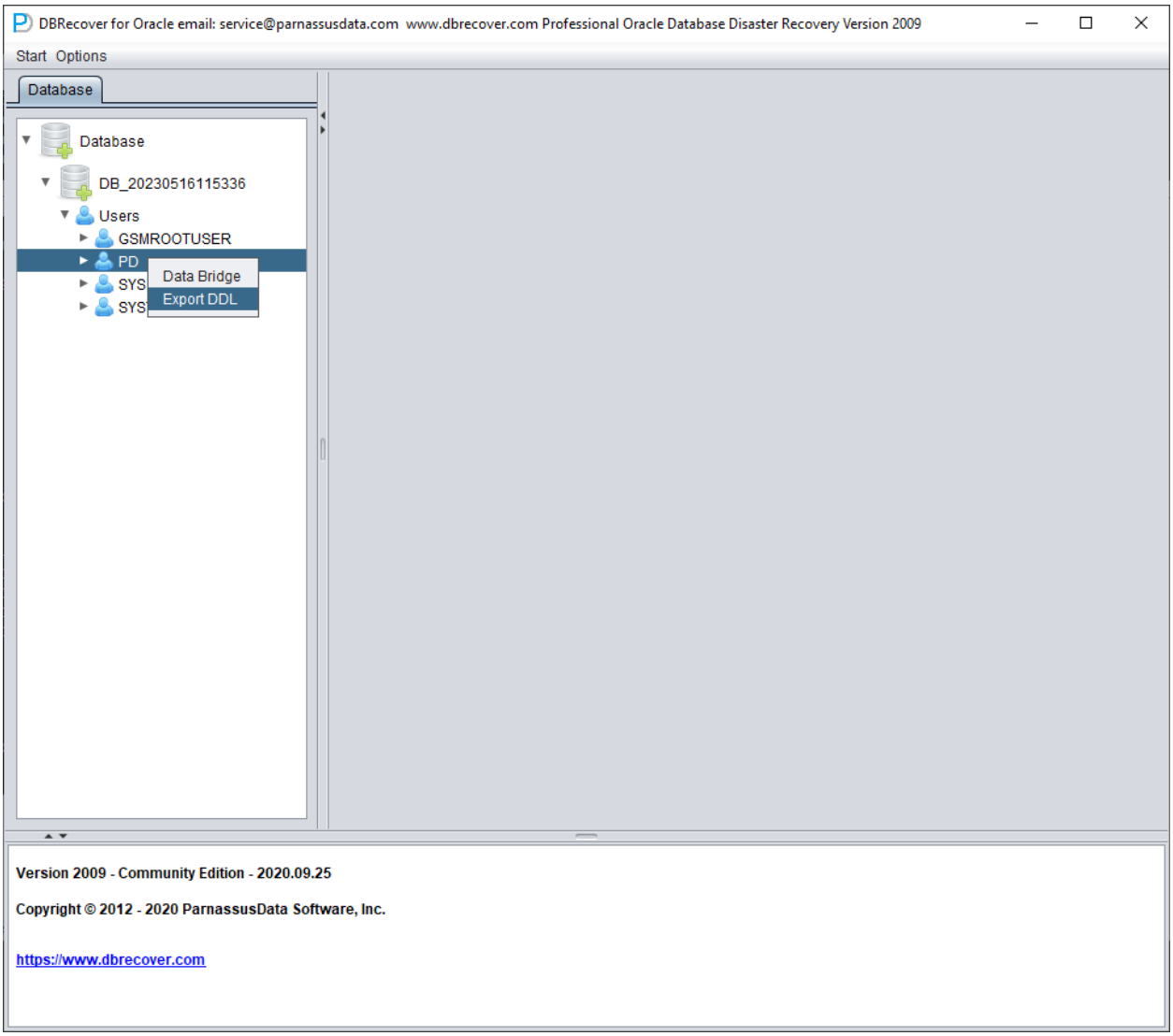
以上在”Create table in restricted mode“下拉库中勾选Yes后，即不再使用宽表模式创建数据表。

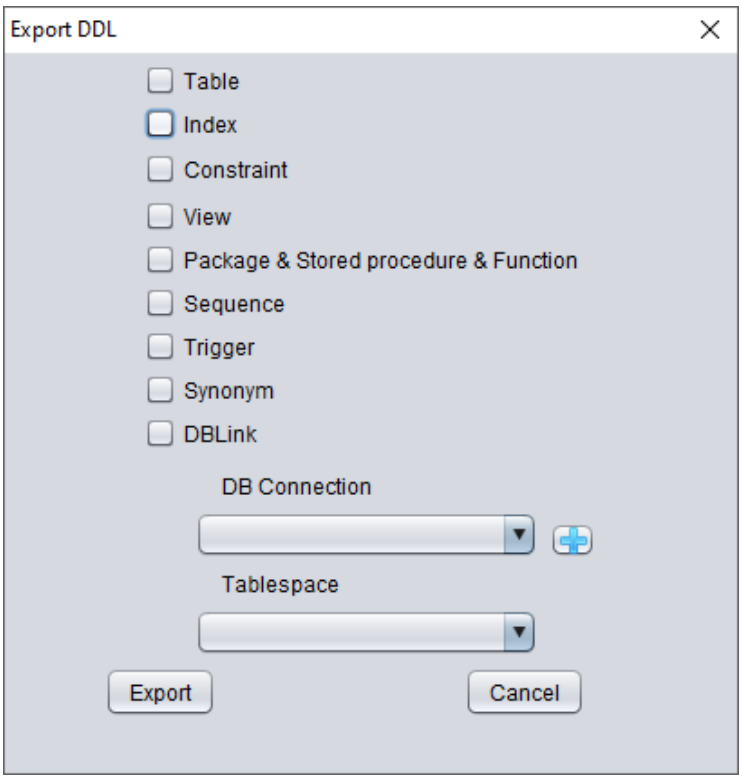
EXPORT DDL功能介绍

以上针对单个SCHEMA的数据表做了恢复操作，其恢复的对象包括：创建了对应的数据表，插入了可恢复的数据。

对于索引、约束、视图、触发器等对象的恢复，可以通过EXPORT DDL功能获得。

选中要恢复的SCHEMA，右键选择EXPORT DDL功能：

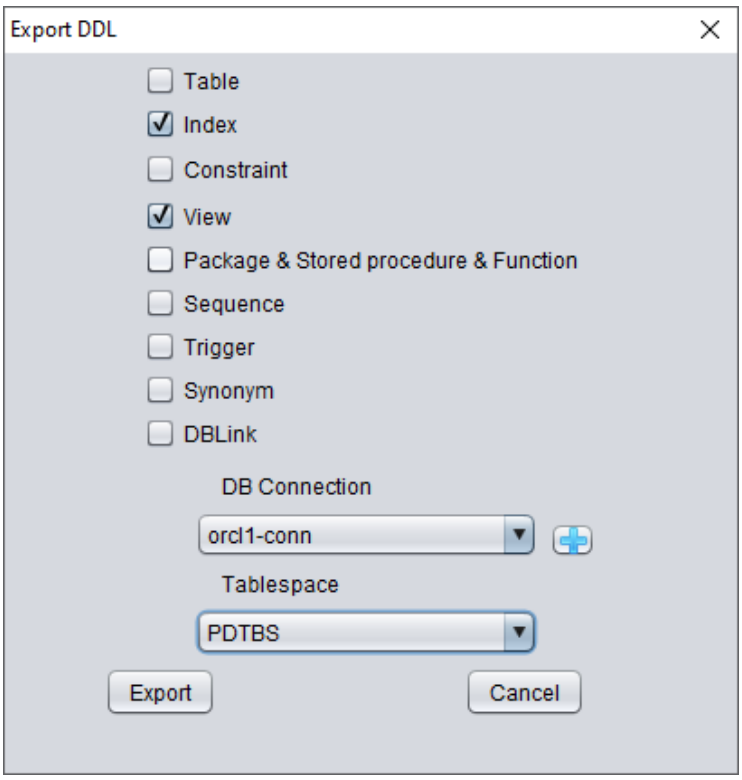

DBRecover for Oracle email: service@parnassusdata.com www.dbrecover.com Professional Oracle Database Disaster Recovery Version 2009
Start Options
Database
Database
DB_20230516115336
Users
GSMROOTUSER
PD
SYS
SYS
Data Bridge
Export DDL
Version 2009 - Community Edition - 2020.09.25
Copyright © 2012 - 2020 ParnassusData Software, Inc.
https://www.dbrecover.com

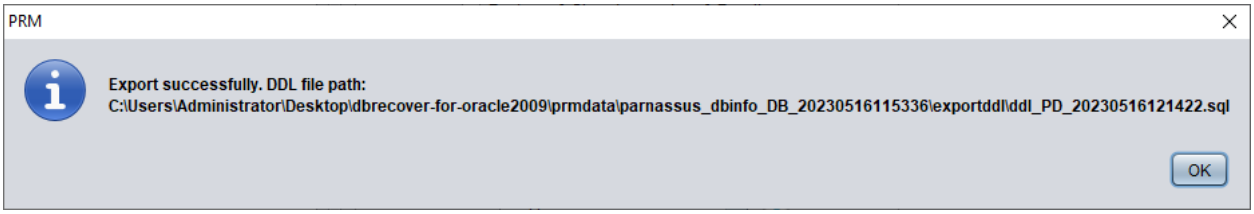


可恢复的对象类型包括：

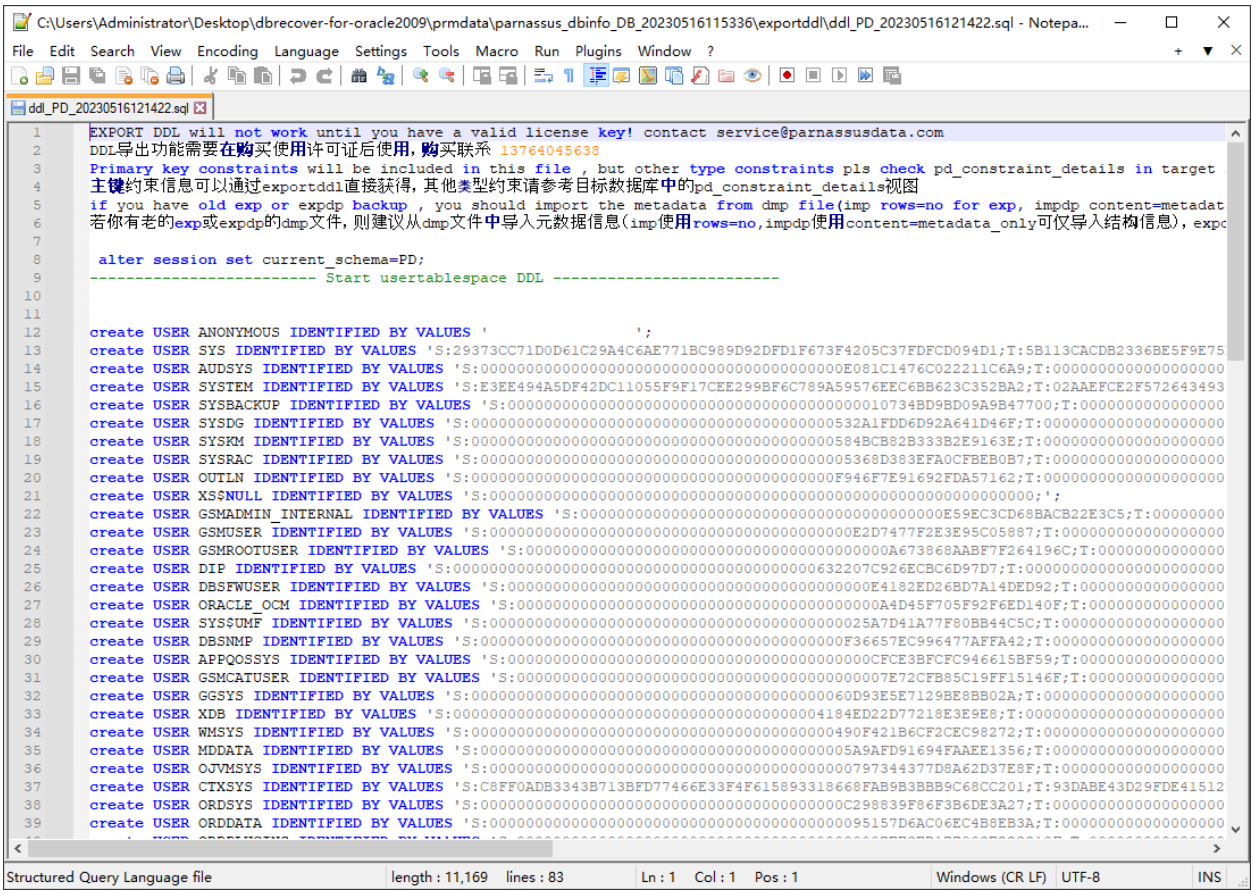
- 建表语句create table statement（注意不包括分区等信息）
- 索引create index statement（注意不包括分区等信息）
- 约束constraint
- 视图view
- 存储过程与函数Package & Stored Procedure & Function
- 序列sequence
- 触发器trigger
- 同义词synonym
- 数据库链接DBlink

此处同样选择之前输入的数据库链接信息，用以临时处理DDL信息。





弹出窗口提示了DDL SQL文件的路径，查看该文件：



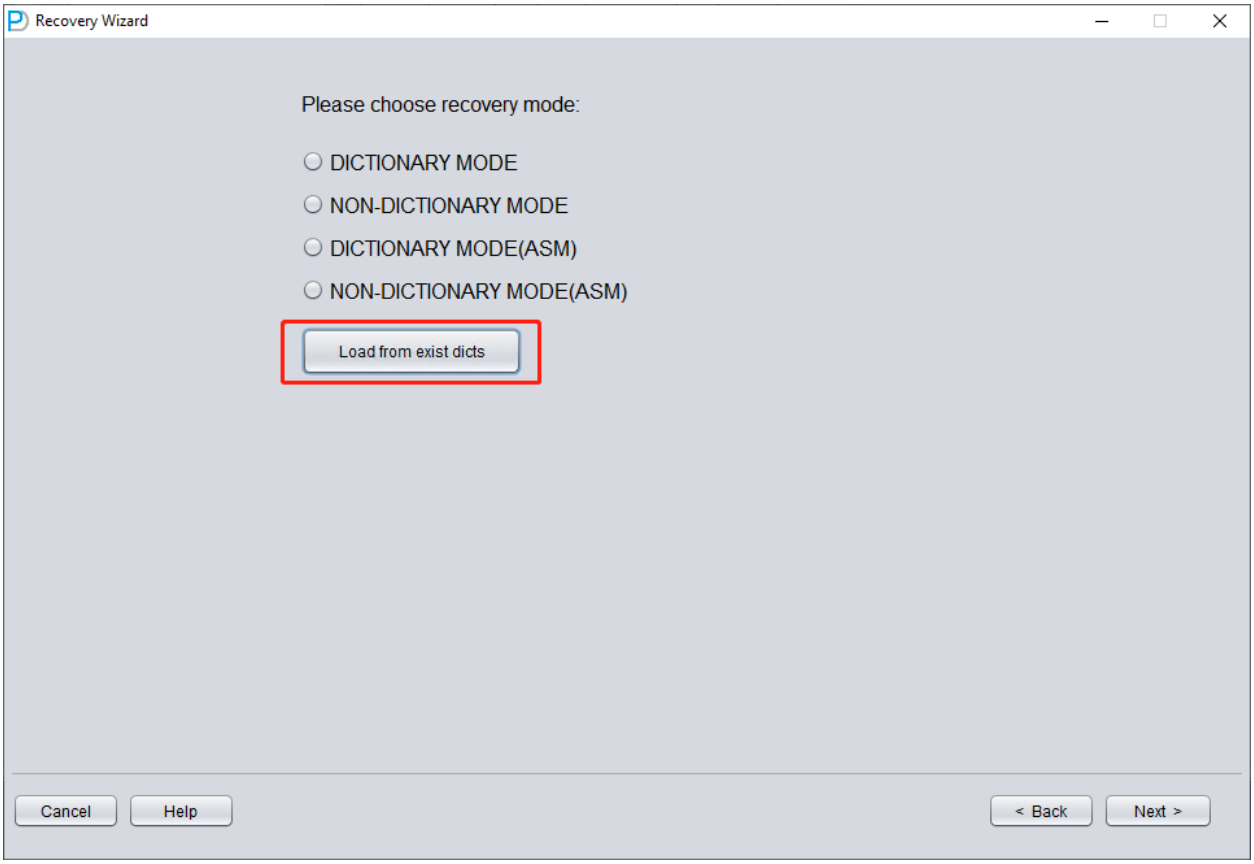
注意：EXPORTDDL功能仅在已经注册有效的企业版许可证(LICENSE KEY)后可以正常使用！

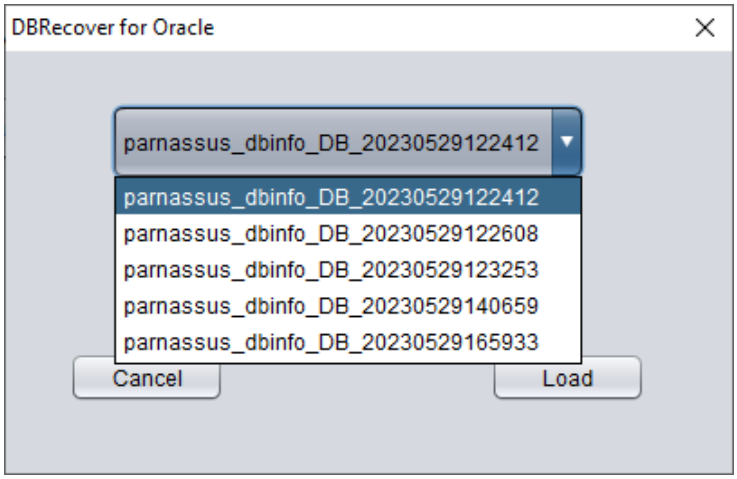
上述DDL SQL文件中的创建索引、视图等对象的语句，需要用户自行复制后到对应数据库的用户下执行。

如果用户有老的exp或expdp的dmp文件，则建议从dmp文件中导入元数据信息（imp使用rows=no,impdp使用content=metadata_only可仅导入结构信息），exportddl功能缺少少量元数据信息，例如对象授权和外键等。

LOAD FROM EXIST DICTS功能介绍：

在实际恢复过程中若遇到程序无响应、卡死、报错等情况，可以在重启DBRECOVER后使用LOAD FROM EXIST DICTS功能直接加载之前的恢复状态。



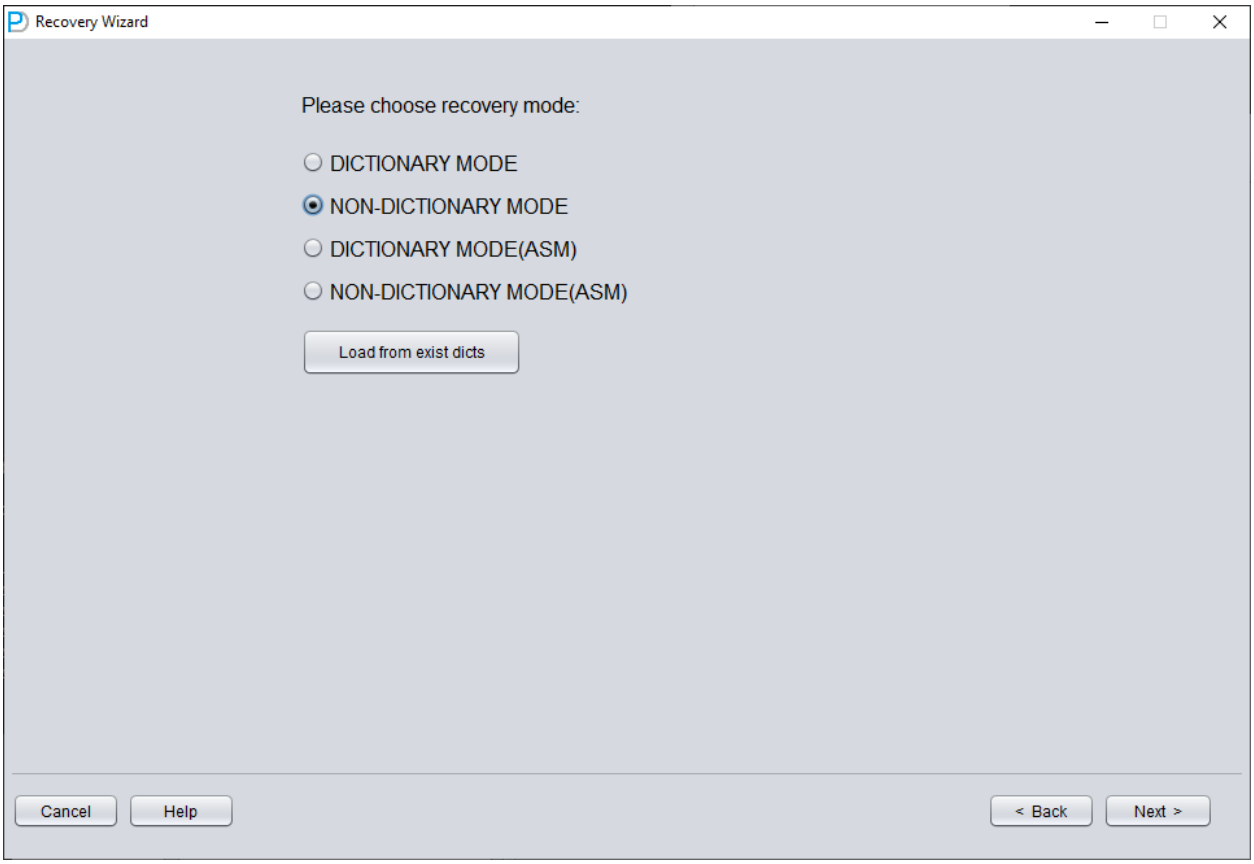


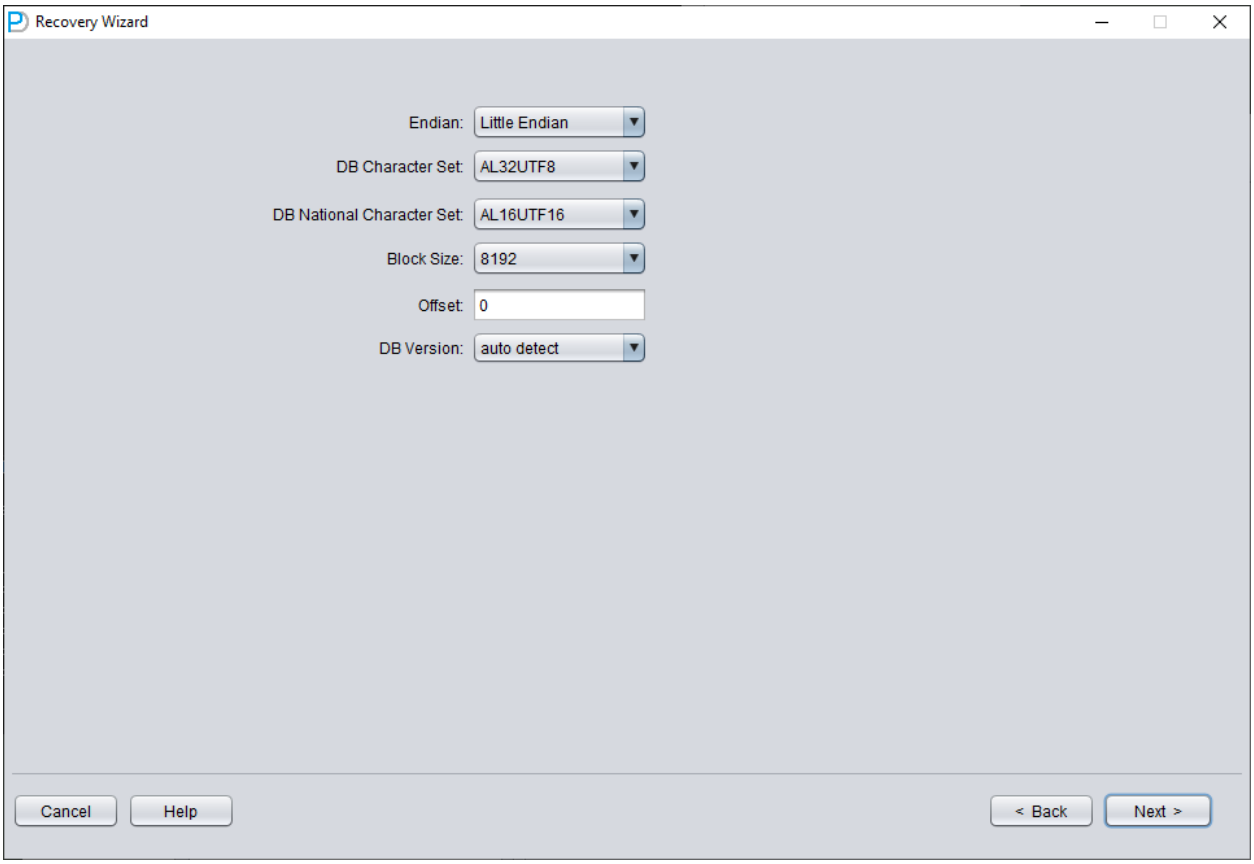
恢复状态按照时间先后排序，适当选择后点击LOAD按钮即可加载。字典模式(DICTIONARY-MODE)和非字典模式（NON-DICTIONARY MODE）均可使用此快速加载功能，以避免重复操作。

恢复场景2 误删除或彻底丢失SYSTEM表空间

D公司的SA系统管理员误删除了某数据库的SYSTEM表空间所在数据文件，这导致数据库完全无法打开，数据无法取出。 在没有备份的情况下，可以利用DBRECOVER挖掘数据。

此场景中启动DBRECOVER后，进入Recovery Wizard后 选择《Non-Dictionary mode》非字典模式:

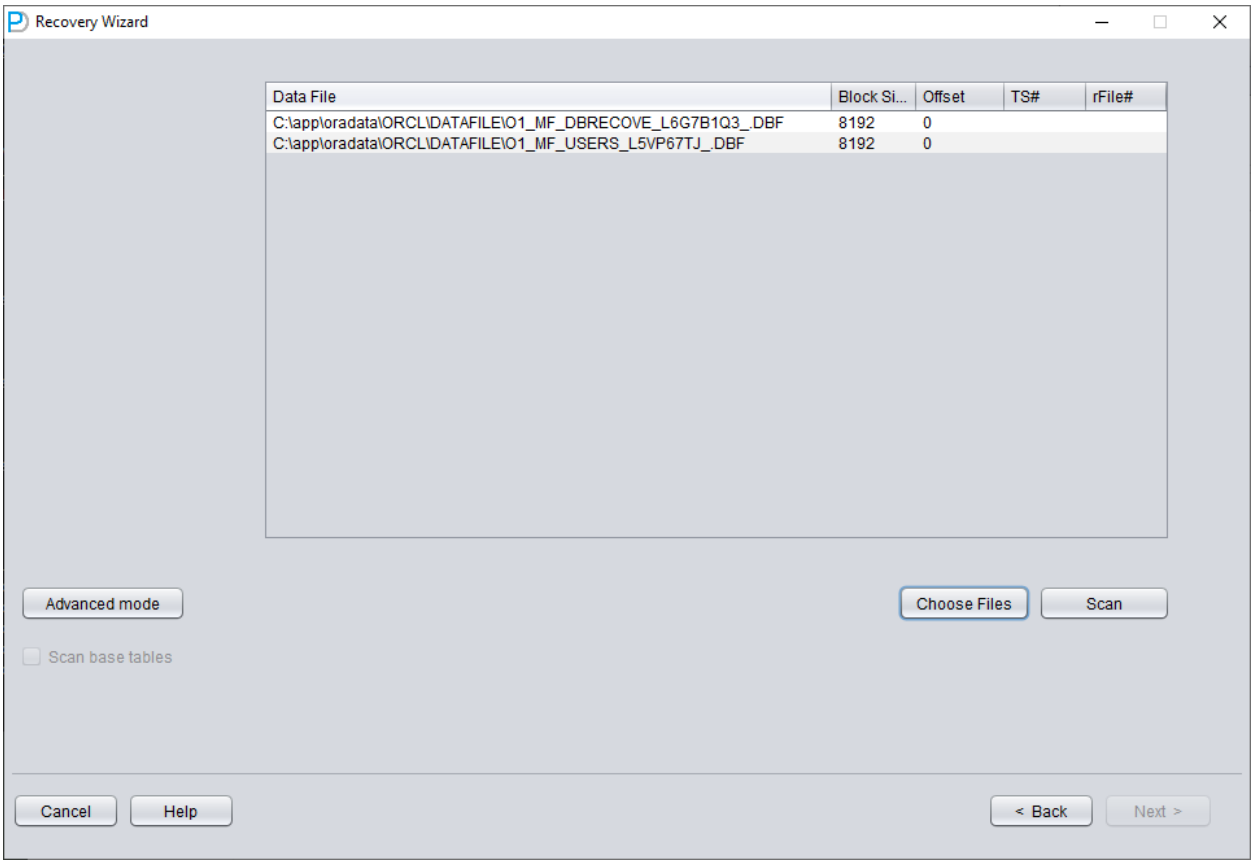




之后要选择正确的字符集，否则后续数据将出现乱码。

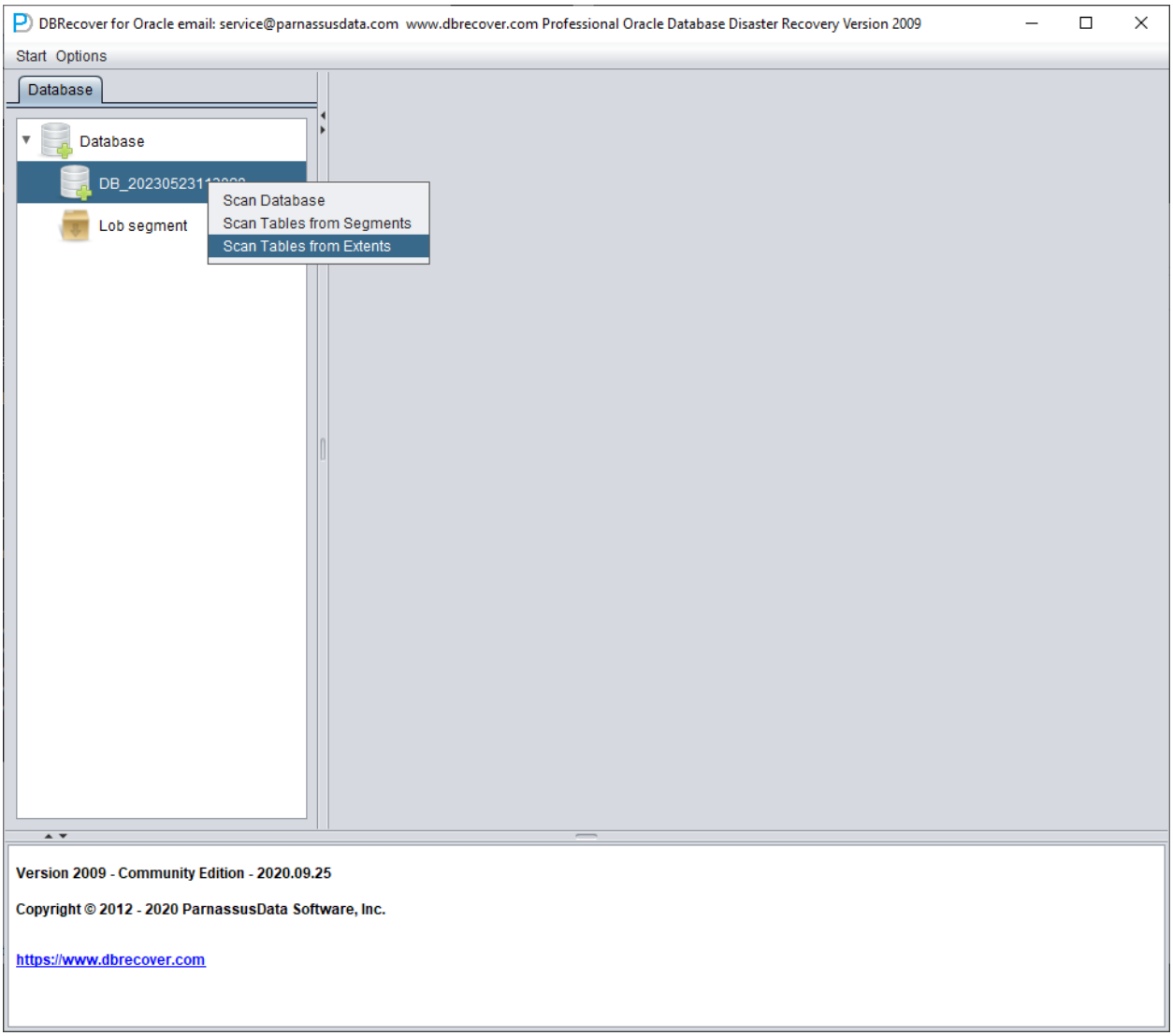
NoN-dictionary模式下需要用户指定字符集和国家字符集，这是因为丢失了SYSTEM表空间后，数据库的字符集信息无法正常获得，所以需要用户的输入。 只有输入正确的字符集设置以及安装了必要的语言包才能保证No-Dictionary模式下正常抽取多国语言。

与场景演示1类似，输入用户目前可得的所有数据文件(不包括临时文件)，并设置正确的Block Size和OFFSET:

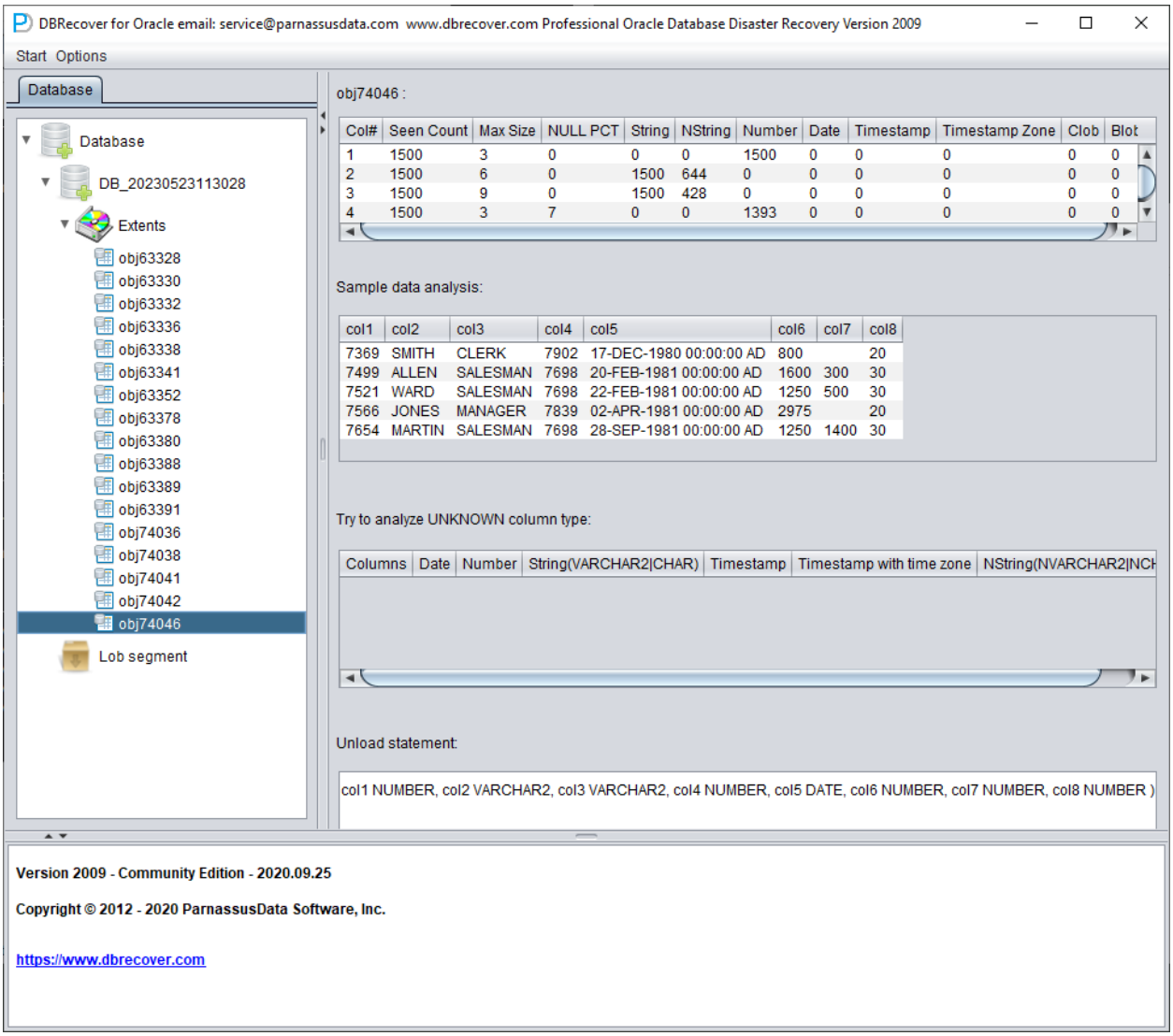


之后点击SCAN，SCAN的作用是扫描所有数据文件上数据信息。

之后选中左侧树形图中的数据库节点右键SCAN EXTENT。仅当可以确认所有数据文件（除了SYSTEM01.DBF）均可用时才使用SCAN TABLE FROM SEGMENTS模式，该模式的优点是速度略微快一些，但对于数据文件不全或存在损坏的情况下其恢复程度要低于SCAN EXTENT模式。

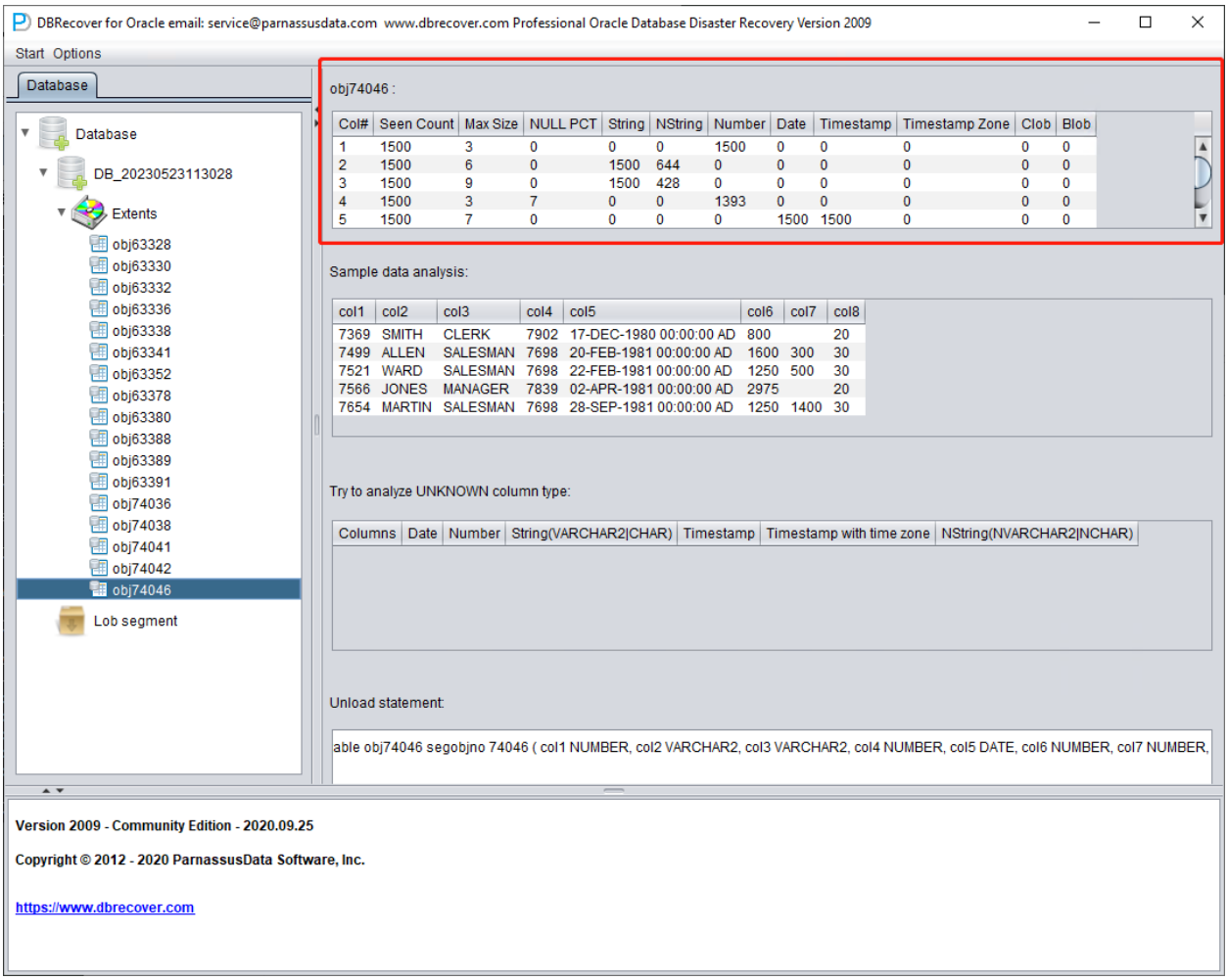


Scan Tables From Extents完成后可以点开主界面左边的树形图：



树形图上每一个节点表示一个普通堆表或分区的数据片段，其名字为obj+ 数据段上记录的DATA OBJECT ID 。

点中一个节点并观察主界面右侧边栏：



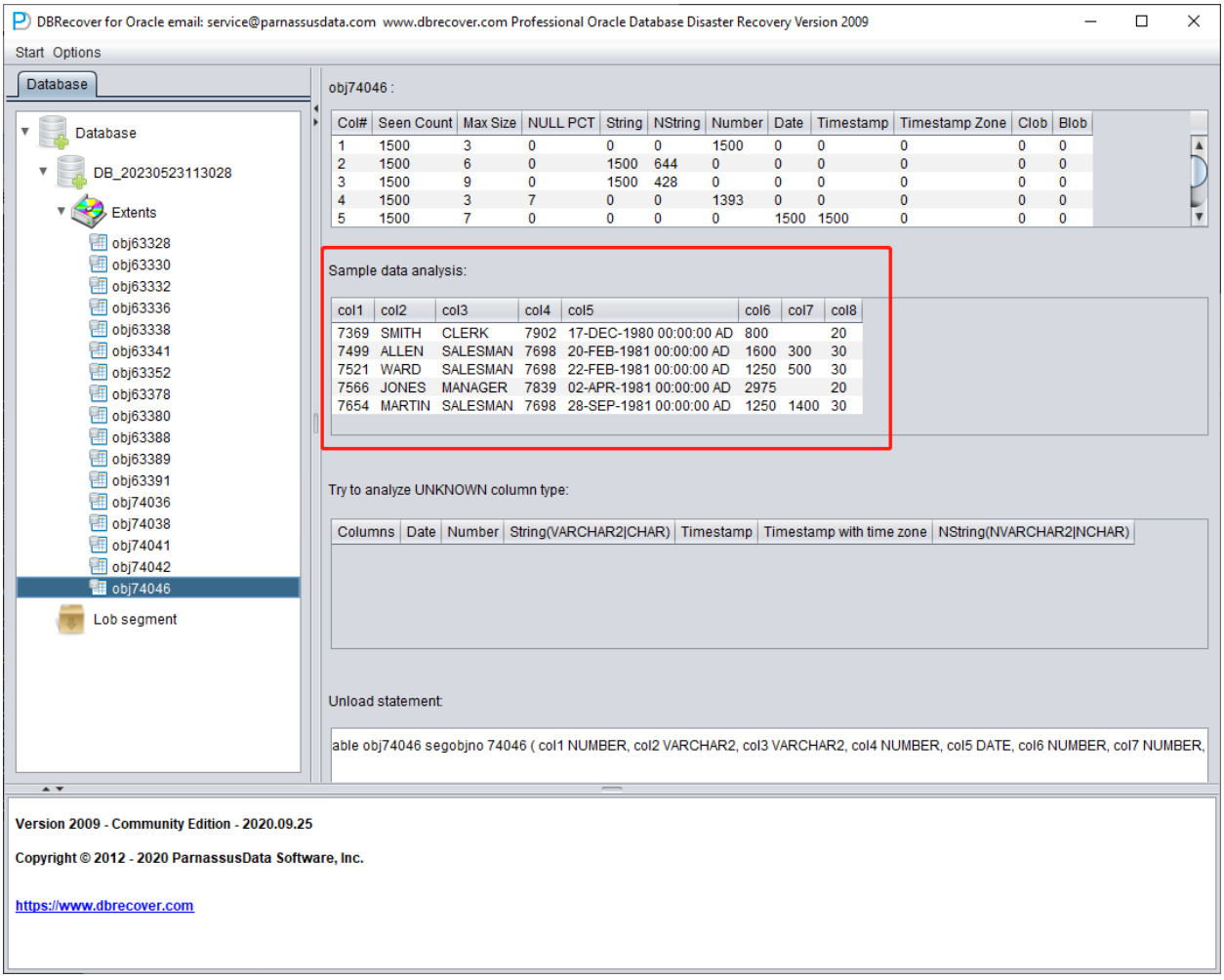
字段类型解析

由于丢失了SYSTEM表空间，故Non-Dictionary模式下缺乏数据表的结构信息，这些结构信息包括表上的字段名字和字段类型，而且在ORACLE中这些信息均只保存为字典信息，不会在数据表上存放。当用户只有应用数据所在表空间时，需要基于数据段上的ROW行数据来猜测每一个字段的类型，这里我们可以解析多达10多种主流数据类型：

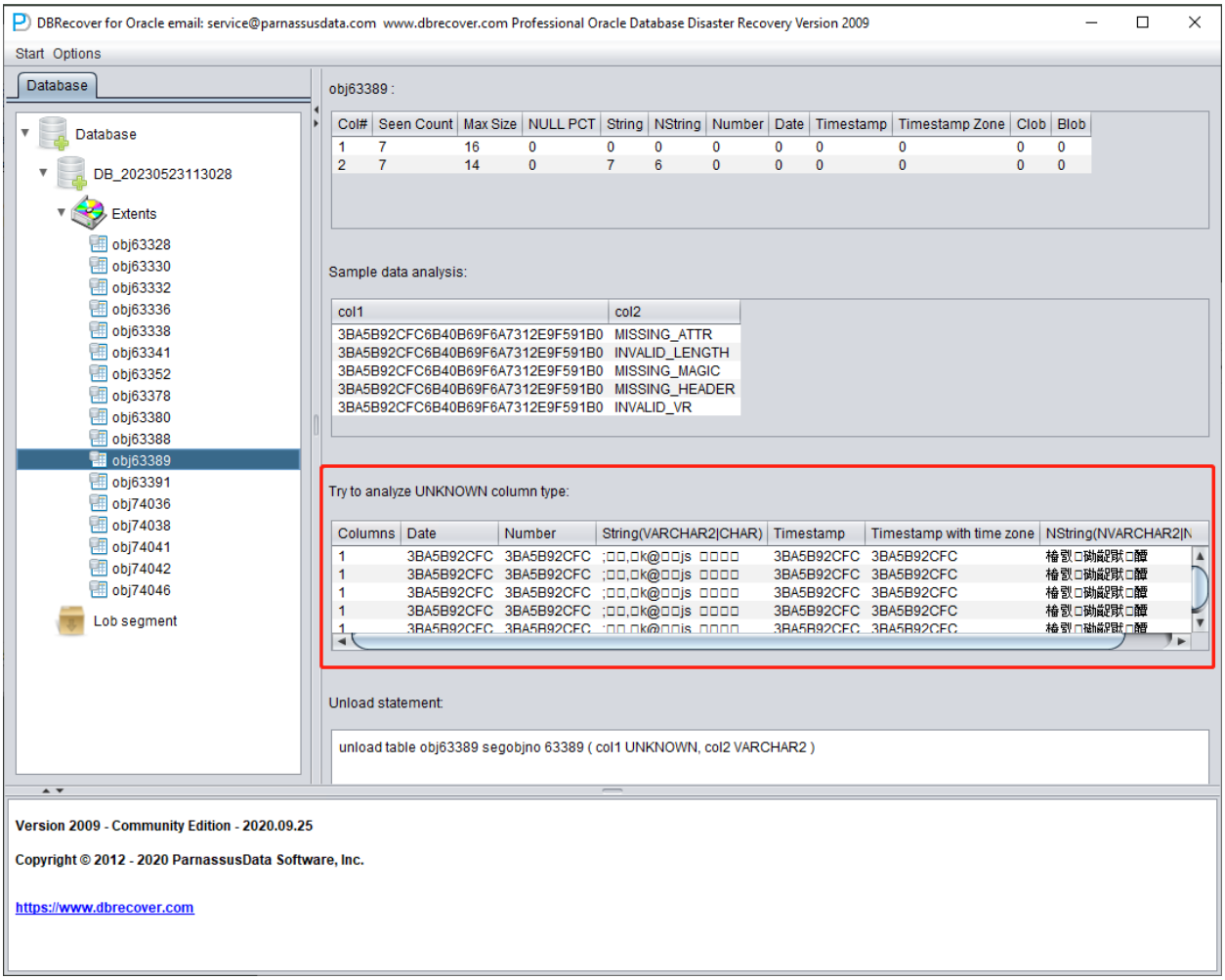
- String字符串:包括char,varchar
- NString国家语言字符串：nchar,nvarchar
- Number数字类型
- Date日期类型

- TimeStamp时间戳类型
- TimeStamp Zone带时区的时间戳类型
- CLOB
- BLOB

示例数据分析Sample Data Analysis:



该部分依据字段类型解析的结果来解析10条数据，并显示解析结果。通过示例数据可以帮助用户了解实际该数据段中存放数据的情况。如果数据段上记录条数不足10条，则显示所有记录。



TRY TO ANALYZE UNKNOWN column type:

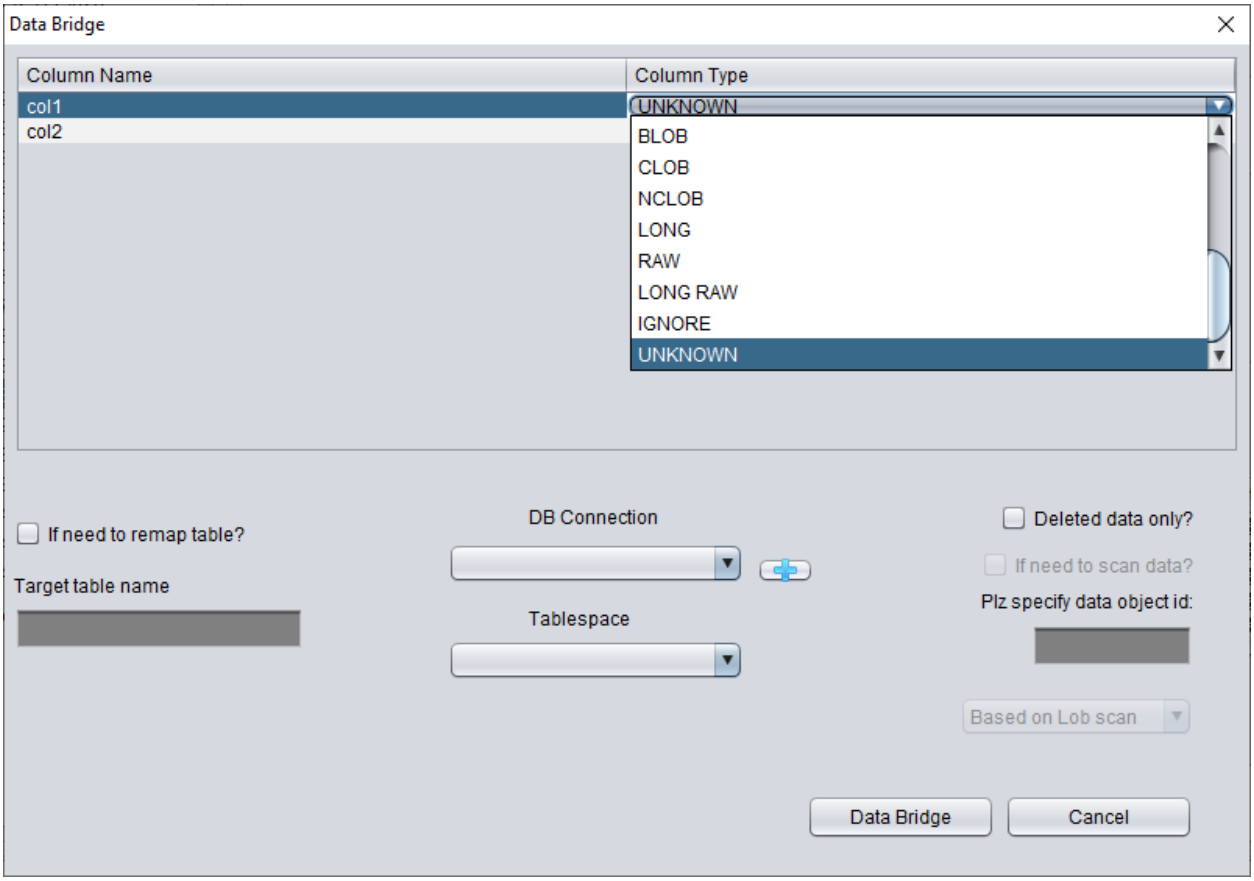
该部分是字段解析功能无法充分确认类型的字段，尝试用各种字段类型来解析，并呈现给用户，以便用户自行判断其究竟是什么类型。

无法确认类型的字段，大致有以下几种情况：

1. RAW或LONG RAW
2. 不支持的数据类型，包括：XDB.XDB$RAW_LIST_T、XMLTYPE、用户自定义类型等
3. 数据块本身严重损坏

在此《Non-Dictionary Mode》非字典模式下同样可以采用常规和数据搭桥模式，与字典模式相比，主要的区别在于在非字典模式下数据搭桥时用户可以自行决定字段的类型，如下图中中部分字段类型为UNKNOWN，即未知的。

如果用户知道这张表设计时的结构（也可以来源于应用开发商的文档），那么可以自行去填选正确的Column Type类型，以便顺利将该表数据搭桥到目标数据库。



恢复场景3 勒索病毒软件加密损坏数据文件的情况

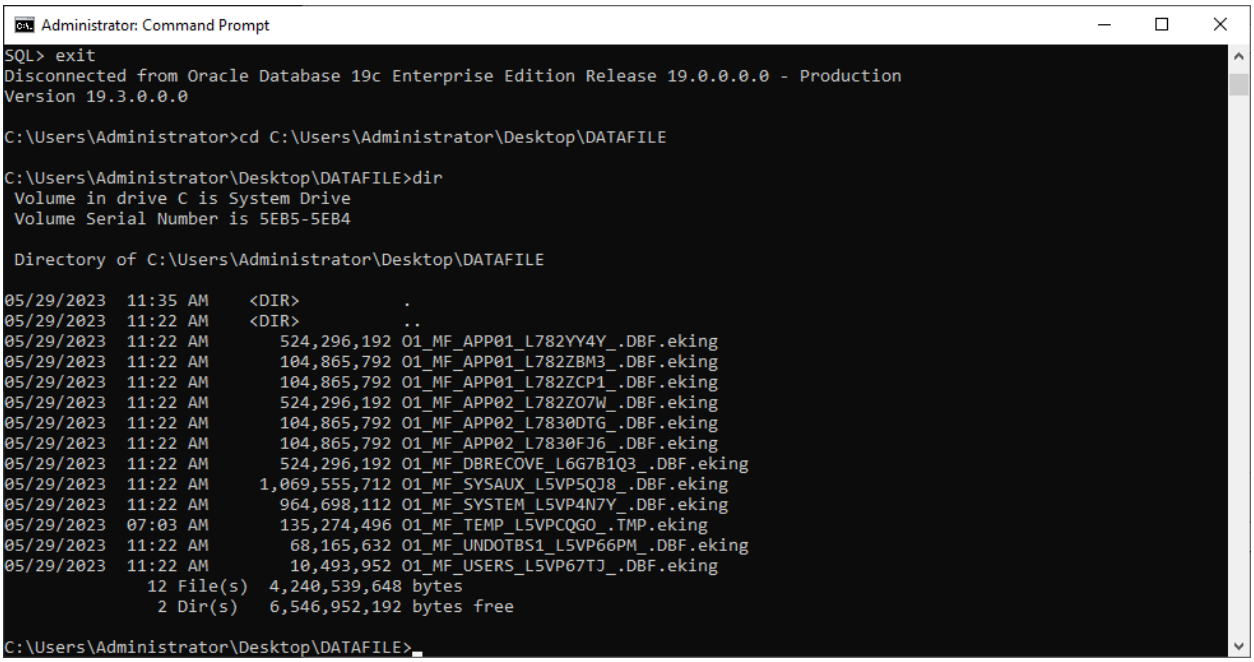
勒索病毒软件（ransomware malware）会将ORACLE数据文件的部分内容或全部内容加密破坏。由于ORACLE的数据文件大小一般较大，全部加密耗时可能会很久，所以部分勒索病毒软件会选择

仅加密ORACLE数据文件头部的连续或随机空间。

对于这种局部的加密破坏，我们可以尝试使用DBRECOVER来恢复其中数据。

由于数据文件头损坏，我们需要通过观察SYSTEM01.DBF的内容来搞清楚，各个数据文件的表空间号(TS#)和相对文件号(RFILE#)等信息。

以下是数据文件清单：



`O1_MF_APP01_L782YY4Y_.DBF.eking`

`O1_MF_APP01_L782ZBM3_.DBF.eking`

`O1_MF_APP01_L782ZCP1_.DBF.eking`

`O1_MF_APP02_L782ZO7W_.DBF.eking`

`O1_MF_APP02_L7830DTG_.DBF.eking`

`O1_MF_APP02_L7830FJ6_.DBF.eking`

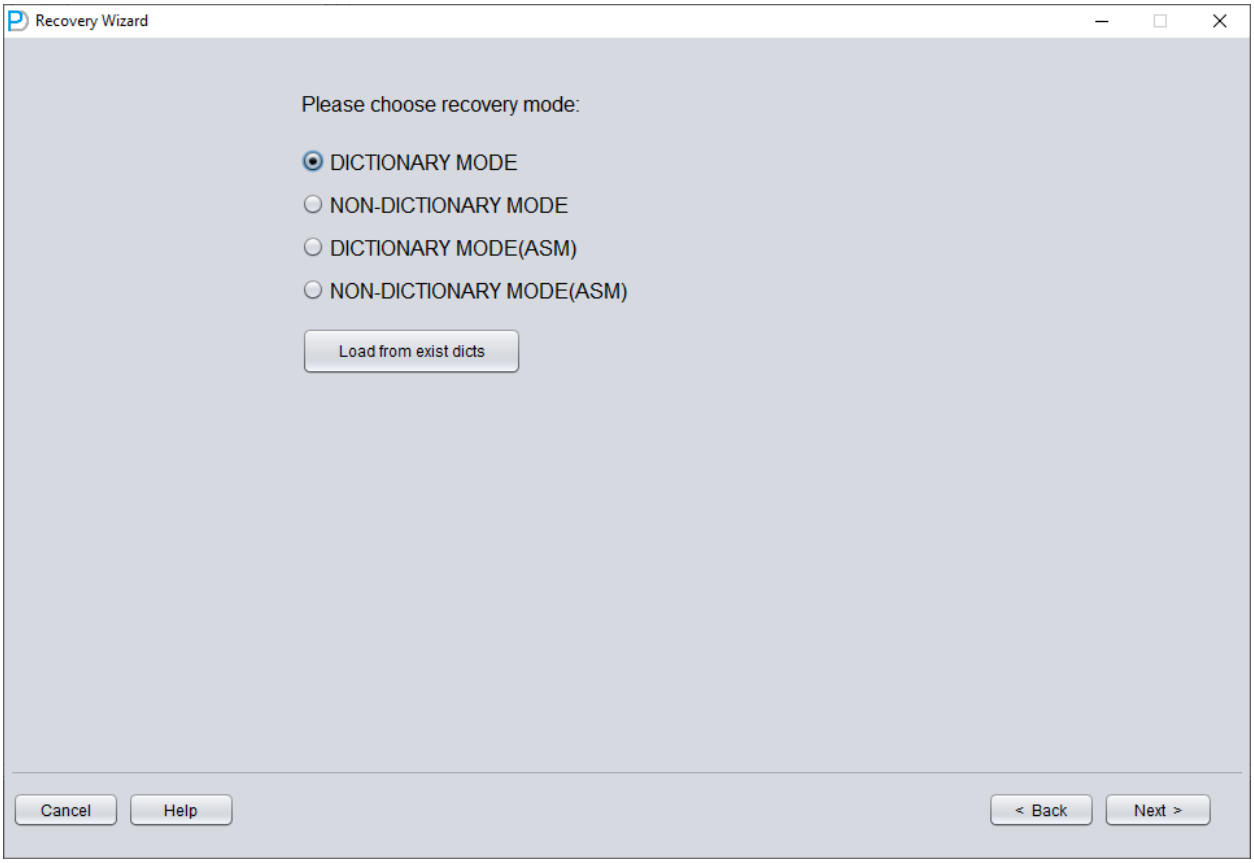
`O1_MF_DBRECOVE_L6G7B1Q3_.DBF.eking`

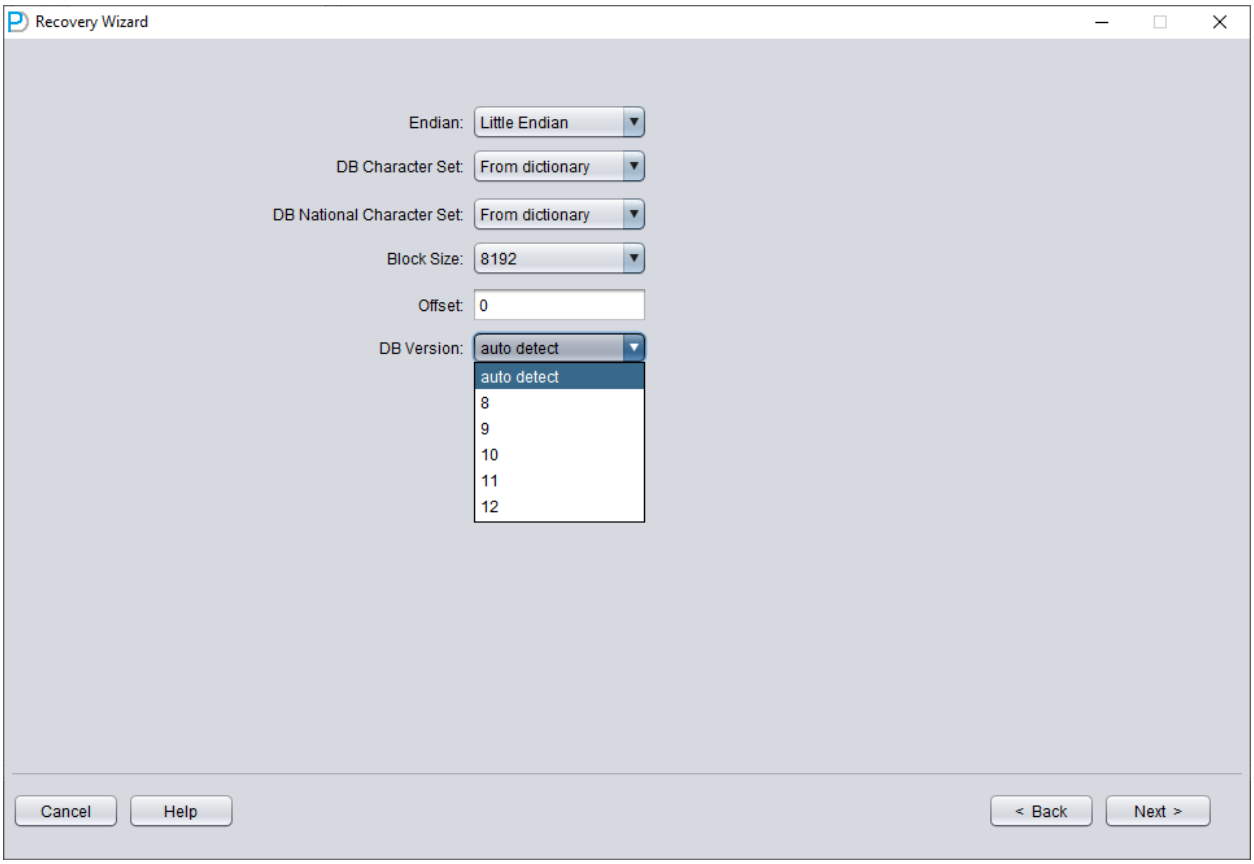
```
O1_MF_SYSAUX_L5VP5QJ8_.DBF.eking
O1_MF_SYSTEM_L5VP4N7Y_.DBF.eking
O1_MF_TEMP_L5VPCQGO_.TMP.eking
O1_MF_UNDOTBS1_L5VP66PM_.DBF.eking
O1_MF_USERS_L5VP67TJ_.DBF.eking
```

以上示例加密后缀为eking

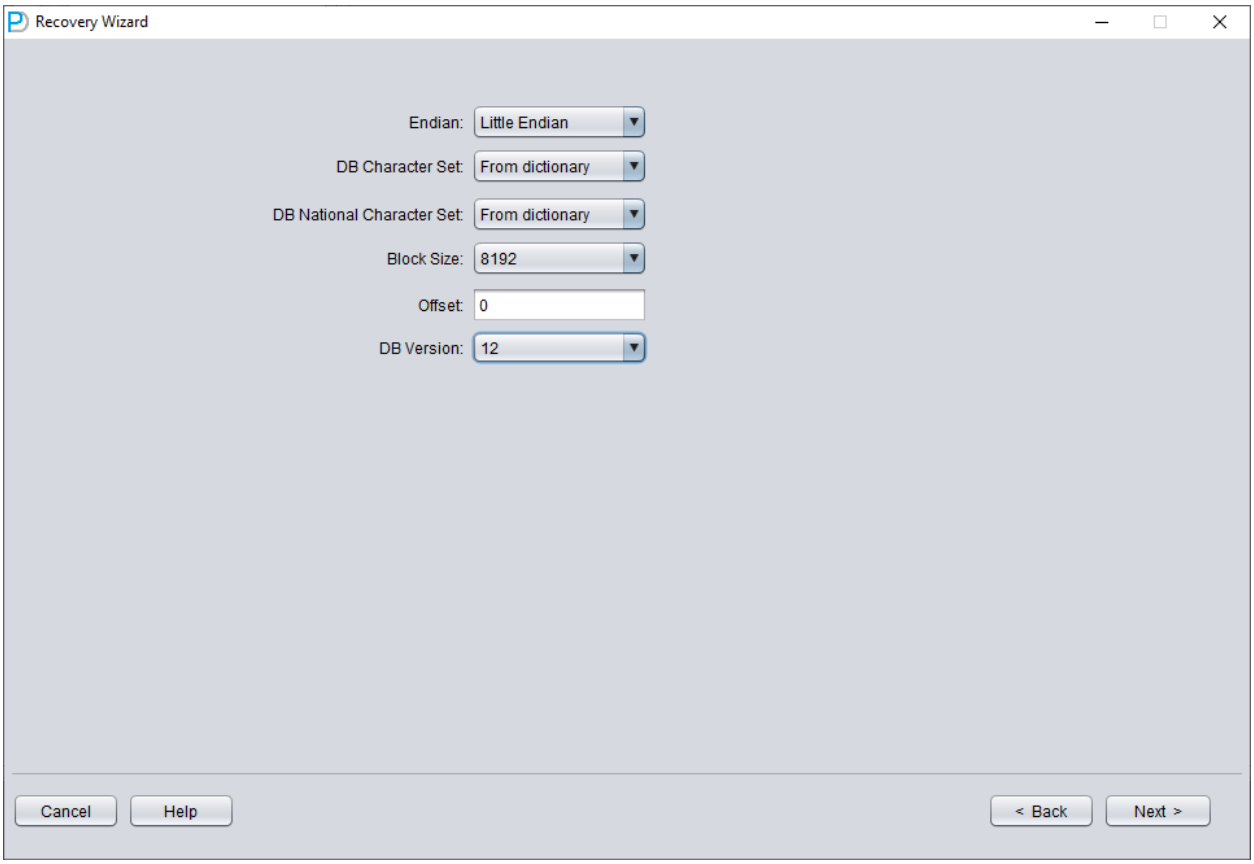
注意其中的TEMP、UNDOTBS1、SYSAUX与我们的恢复作业无关，可以忽略这些文件。

我们首先启动DBRECOVER，使用字典模式DICT-MODE：

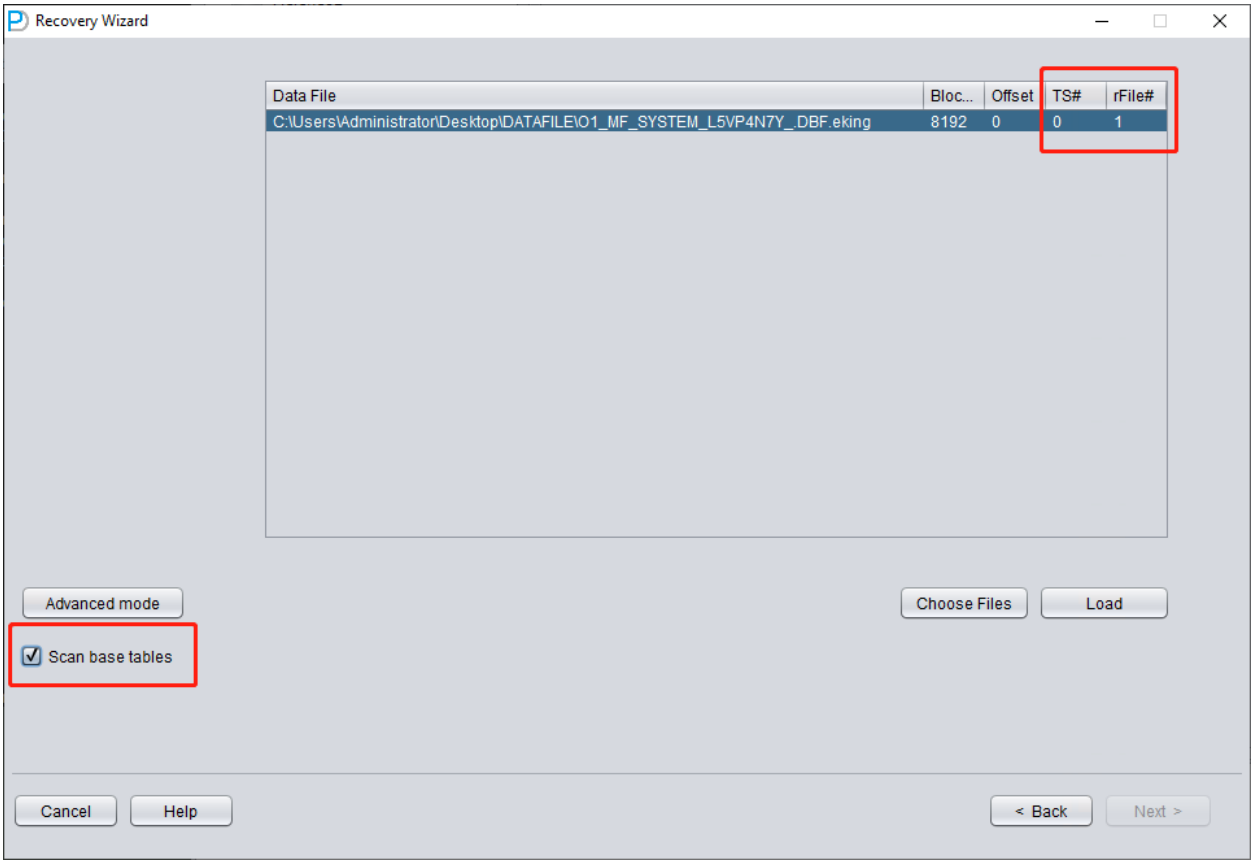




按实际情况选择 DB VERSION，对于版本高于12c的实例，例如18c 19c等均选择12。

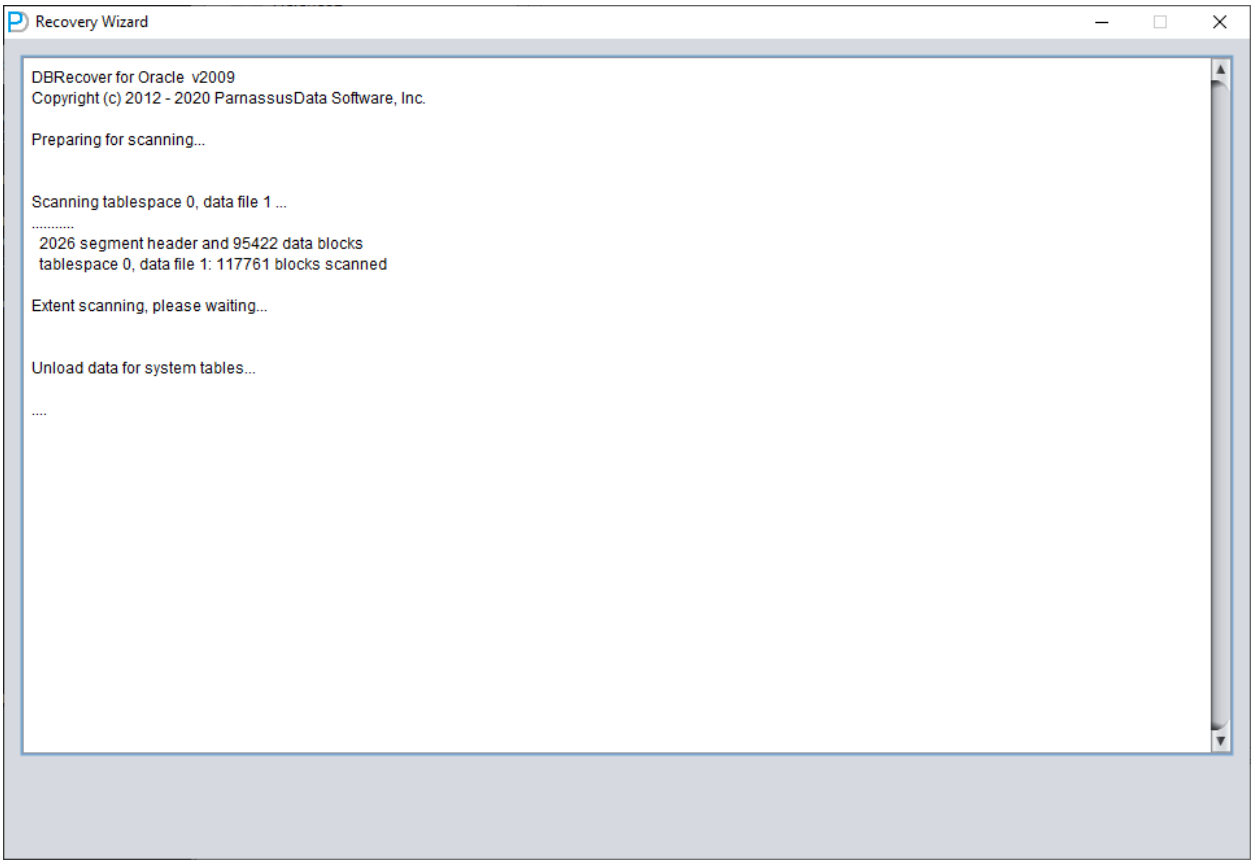


仅仅加入SYSTEM01.DBF，并指定其TS#=0 rFILE#=1（注意这是固定的）。

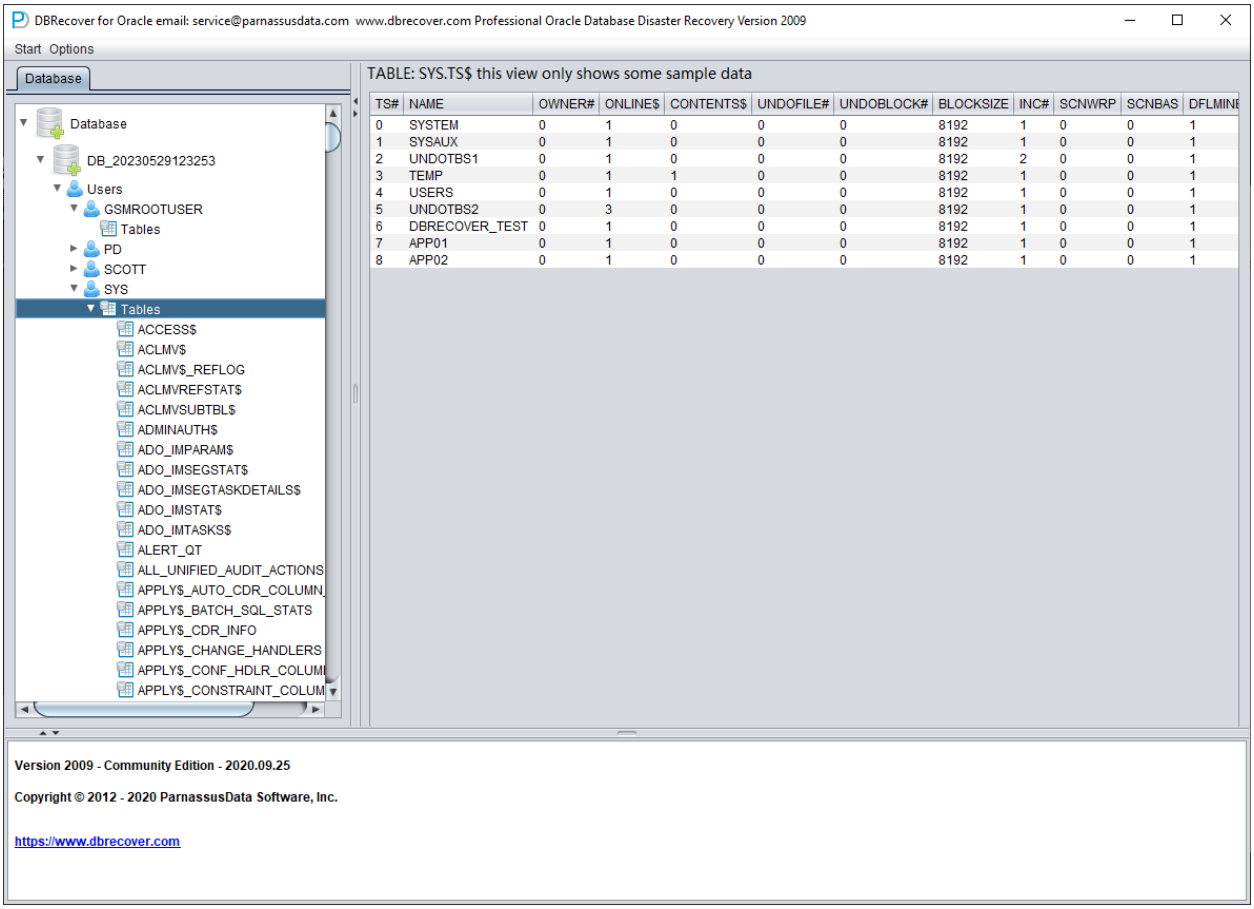


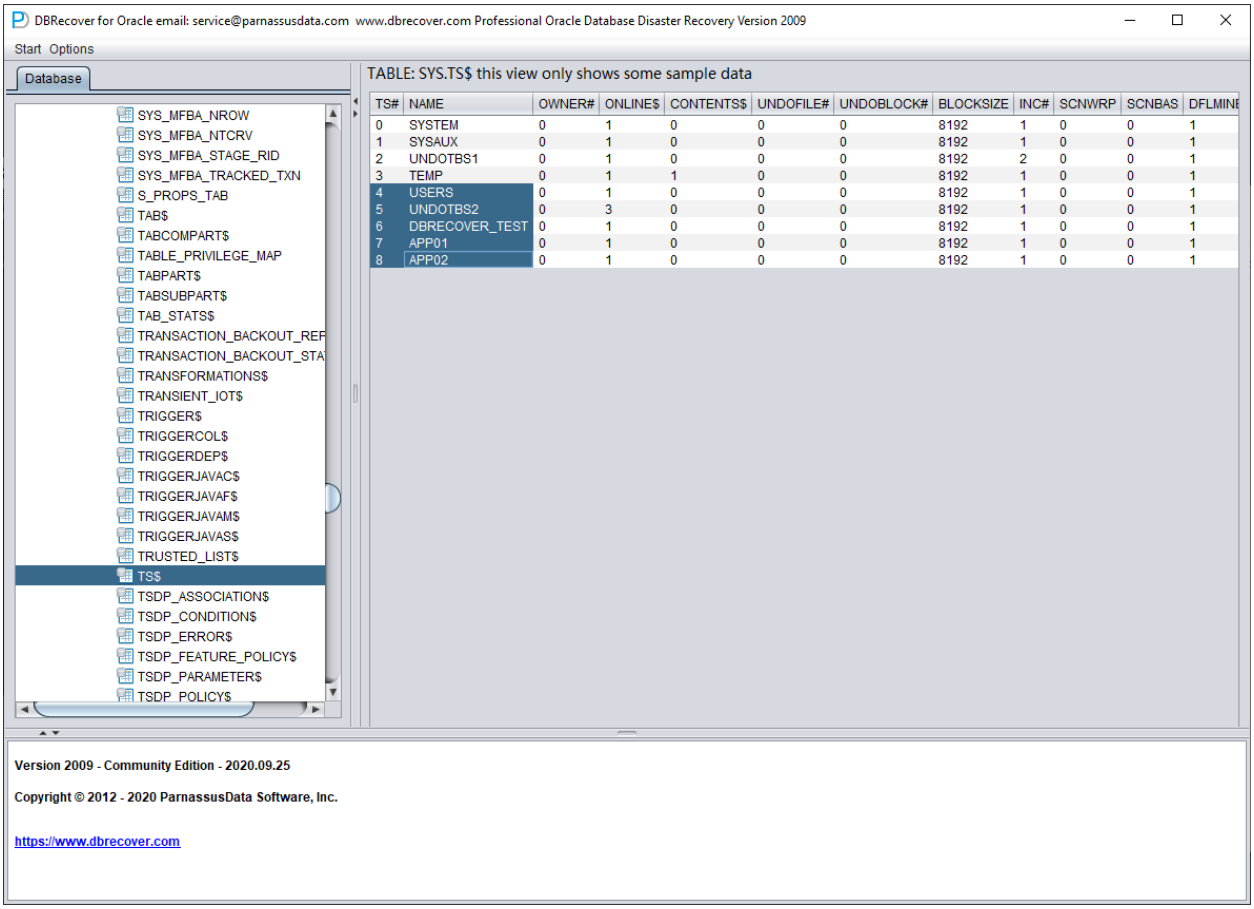
以上勾选”SCAN BASE TABLES”选项可以更强有力地应对损坏情况。

之后点击LOAD按钮，DBRECOVER会整体扫描SYSTEM01.DBF并找出其中的数据字典基表数据。



我们打开SYS用户节点，查找TS$和FILE$ 2个基础表：


DBRecover for Oracle email: service@parnassusdata.com www.dbrecover.com Professional Oracle Database Disaster Recovery Version 2009
Start Options
Database
Database
DB_20230529123253
Users
GSMROOTUSER
Tables
PD
SCOTT
SYS
Tables
ACCESS$
ACLMV$
ACLMV$_REFLOG
ACLMVREFSTAT$
ACLMVSUBTBL$
ADMINAUTH$
ADO_IMPARAM$
ADO_IMSEGSTAT$
ADO_IMSEGTASKDETAILS$
ADO_IMSTAT$
ADO_IMTASKS$
ALERT_QT
ALL_UNIFIED_AUDIT_ACTIONS
APPLY$_AUTO_CDR_COLUMN_
APPLY$_BATCH_SQL_STATS
APPLY$_CDR_INFO
APPLY$_CHANGE_HANDLERS
APPLY$_CONF_HDLR_COLUMI
APPLY$_CONSTRAINT_COLUM
TABLE: SYS.TS$ this view only shows some sample data
TS# NAME OWNER# ONLINE$ CONTENTS$ UNDOFILE# UNDOBLOCK# BLOCKSIZE INC# SCNWRP SCNBAS DFLMINE
0 SYSTEM 0 1 0 0 0 8192 1 0 0 1
1 SYSAUX 0 1 0 0 0 8192 1 0 0 1
2 UNDOTBS1 0 1 0 0 0 8192 2 0 0 1
3 TEMP 0 1 1 0 0 8192 1 0 0 1
4 USERS 0 1 0 0 0 8192 1 0 0 1
5 UNDOTBS2 0 3 0 0 0 8192 1 0 0 1
6 DBRECOVER_TEST 0 1 0 0 0 8192 1 0 0 1
7 APP01 0 1 0 0 0 8192 1 0 0 1
8 APP02 0 1 0 0 0 8192 1 0 0 1
Version 2009 - Community Edition - 2020.09.25
Copyright © 2012 - 2020 ParnassusData Software, Inc.
https://www.dbrecover.com

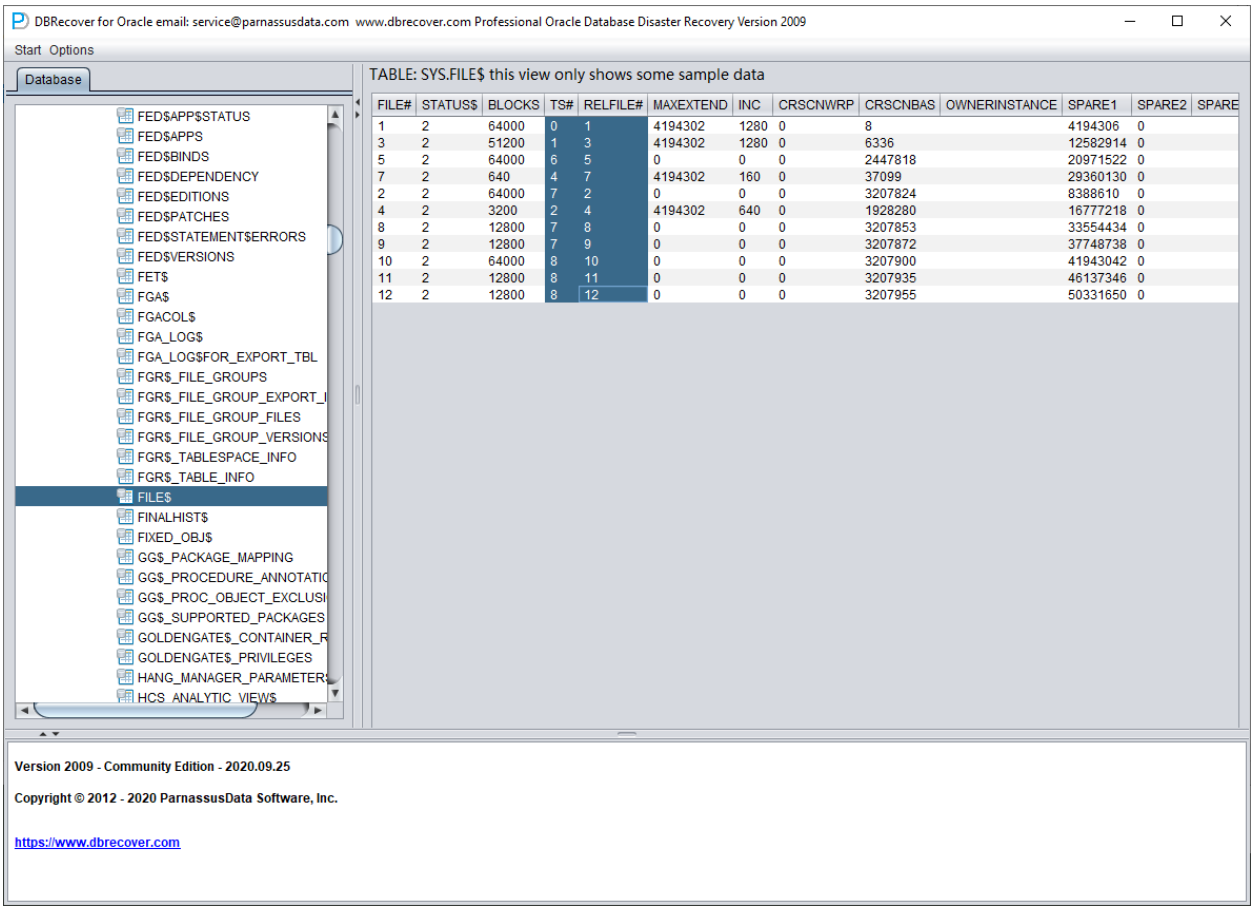


TS$表存放了表空间信息，TS#列为表空间号，可以得出如下信息：

TS#	NAME
0	SYSTEM
1	SYSAUX
2	UNDOTBS1
3	TEMP
4	USERS
5	UNDOTBS2
6	DBRECOVER_TEST
7	APP01
8	APP02

即APP01表空间的TS#=7，而APP02表空间的TS#=8

FILE$表存放了数据文件信息：



FILE#	STATUS$	BLOCKS	TS#	RELFILE#	MAXEXTEND	INC	CRSCNWRP	CRSCNBAS	OWNERINSTANCE	SPARE1	SPARE2	SPARE
1	2	64000	0	1	4194302	1280	0	8		4194306	0	
3	2	51200	1	3	4194302	1280	0	6336		12582914	0	
5	2	64000	6	5	0	0	0	2447818		20971522	0	
7	2	640	4	7	4194302	160	0	37099		29360130	0	
2	2	64000	7	2	0	0	0	3207824		8388610	0	
4	2	3200	2	4	4194302	640	0	1928280		16777218	0	
8	2	12800	7	8	0	0	0	3207853		33554434	0	
9	2	12800	7	9	0	0	0	3207872		37748738	0	
10	2	64000	8	10	0	0	0	3207900		41943042	0	
11	2	12800	8	11	0	0	0	3207935		46137346	0	
12	2	12800	8	12	0	0	0	3207955		50331650	0	

其中我们需要的是TS#和RELFILE#这2列

TS#	RELFILE#
0	1
1	3

6	5
4	7
7	2
2	4
7	8
7	9
8	10
8	11
8	12

通过将两张表格的数据映射合并，可以得到：

TS#	RELFILE#	Tablespace Name
0	1	SYSTEM
1	3	SYSAUX
6	5	DBRECOVER_TEST
4	7	USERS
7	2	APP01
2	4	UNDOTBS1
7	8	APP01
7	9	APP01
8	10	APP02
8	11	APP02
8	12	APP02

删除掉不需要的SYSAUX、UNDOTBS1和已经知道的SYSTEM表空间，则只剩下：

TS#	RELFILE#	Tablespace Name
6	5	DBRECOVER_TEST
4	7	USERS
7	2	APP01

7	8	APP01
7	9	APP01
8	10	APP02
8	11	APP02
8	12	APP02

对应数据文件名字列表：

```
O1_MF_APP01_L782YY4Y_.DBF.eking
O1_MF_APP01_L782ZBM3_.DBF.eking
O1_MF_APP01_L782ZCP1_.DBF.eking
O1_MF_APP02_L782ZO7W_.DBF.eking
O1_MF_APP02_L7830DTG_.DBF.eking
O1_MF_APP02_L7830FJ6_.DBF.eking
O1_MF_DBRECOVE_L6G7B1Q3_.DBF.eking
O1_MF_USERS_L5VP67TJ_.DBF.eking
```

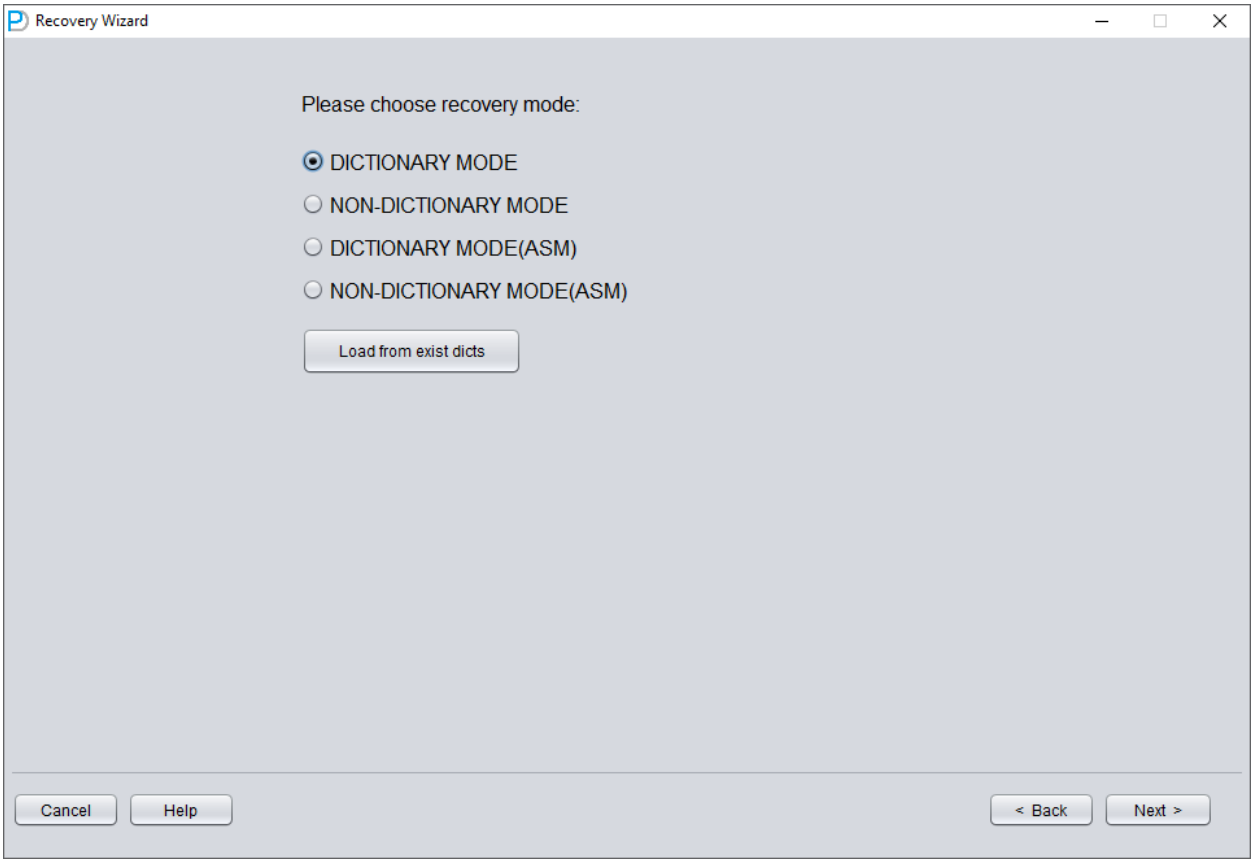
对照上面2个表格，不难发现其中对应关系。对于使用db_create_file_dest OMF文件管理的数据文件，一个表空间下的多个数据文件可按其文件名排序，其顺序与RELFILE#排序一致。对于用户自行管理的文件名（即不使用OMF的情况），一般也会以APP01{XX}(如APP0101、APP0102)之命名方式以便于管理，则也可以获得其对应关系。

以上我们通过猜测获得完整的信息表：

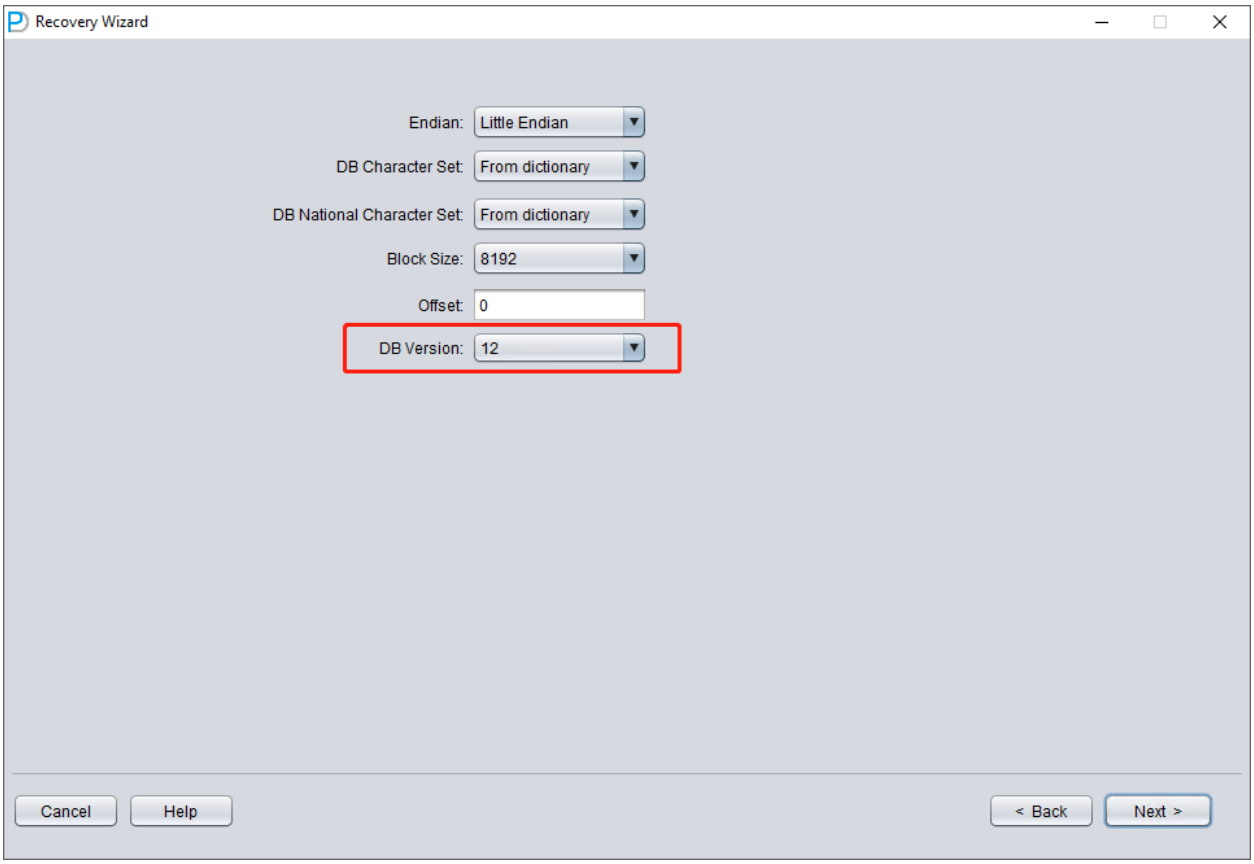
TS#	RFILE#	Tablespace Name	FILE NAME
6	5	DBRECOVER_TEST	O1_MF_DBRECOVE_L6G7B1Q3_.DBF.eking
4	7	USERS	O1_MF_USERS_L5VP67TJ_.DBF.eking
7	2	APP01	O1_MF_APP01_L782YY4Y_.DBF.eking
7	8	APP01	O1_MF_APP01_L782ZBM3_.DBF.eking
7	9	APP01	O1_MF_APP01_L782ZCP1_.DBF.eking
8	10	APP02	O1_MF_APP02_L782ZO7W_.DBF.eking

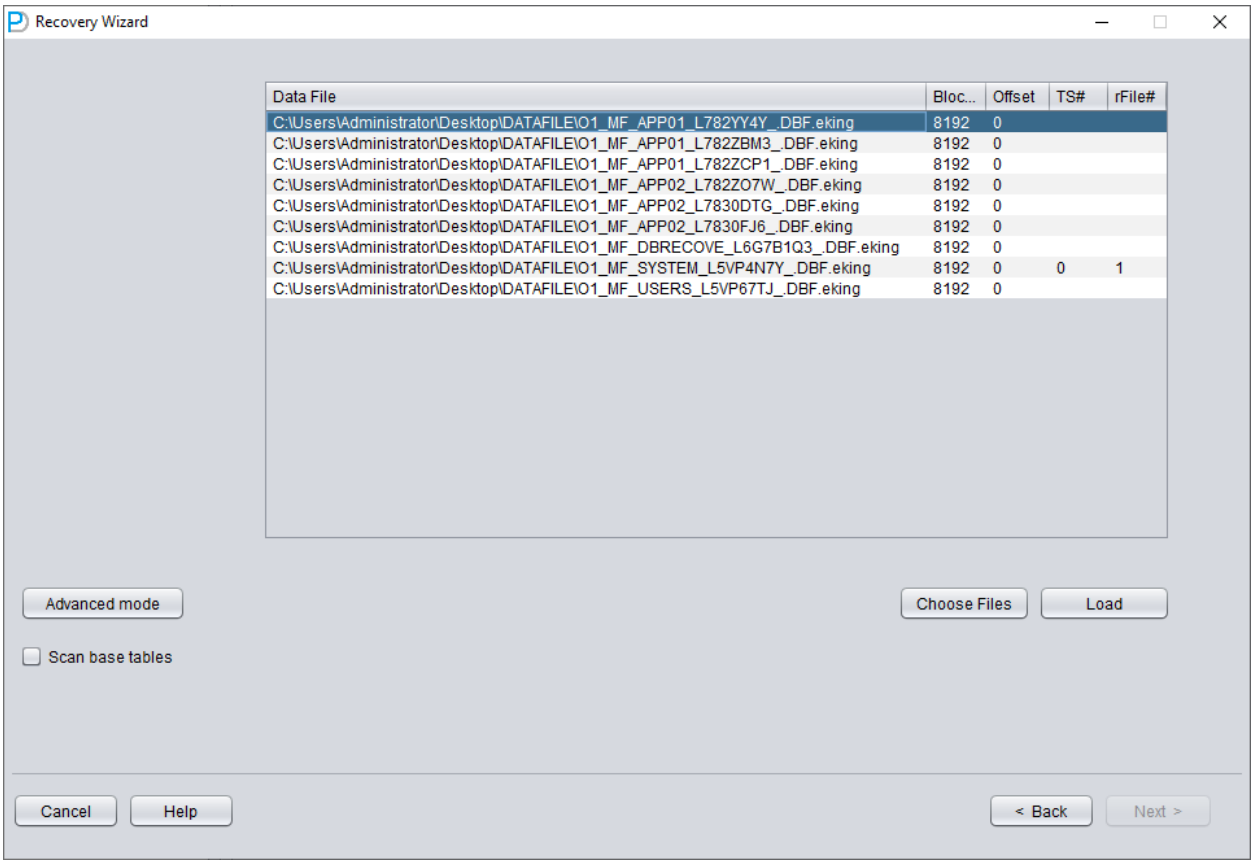
8	11	APP02	O1_MF_APP02_L7830DTG_.DBF.eking
8	12	APP02	O1_MF_APP02_L7830FJ6_.DBF.eking

重新打开DBRECOVER，进入字典模式：



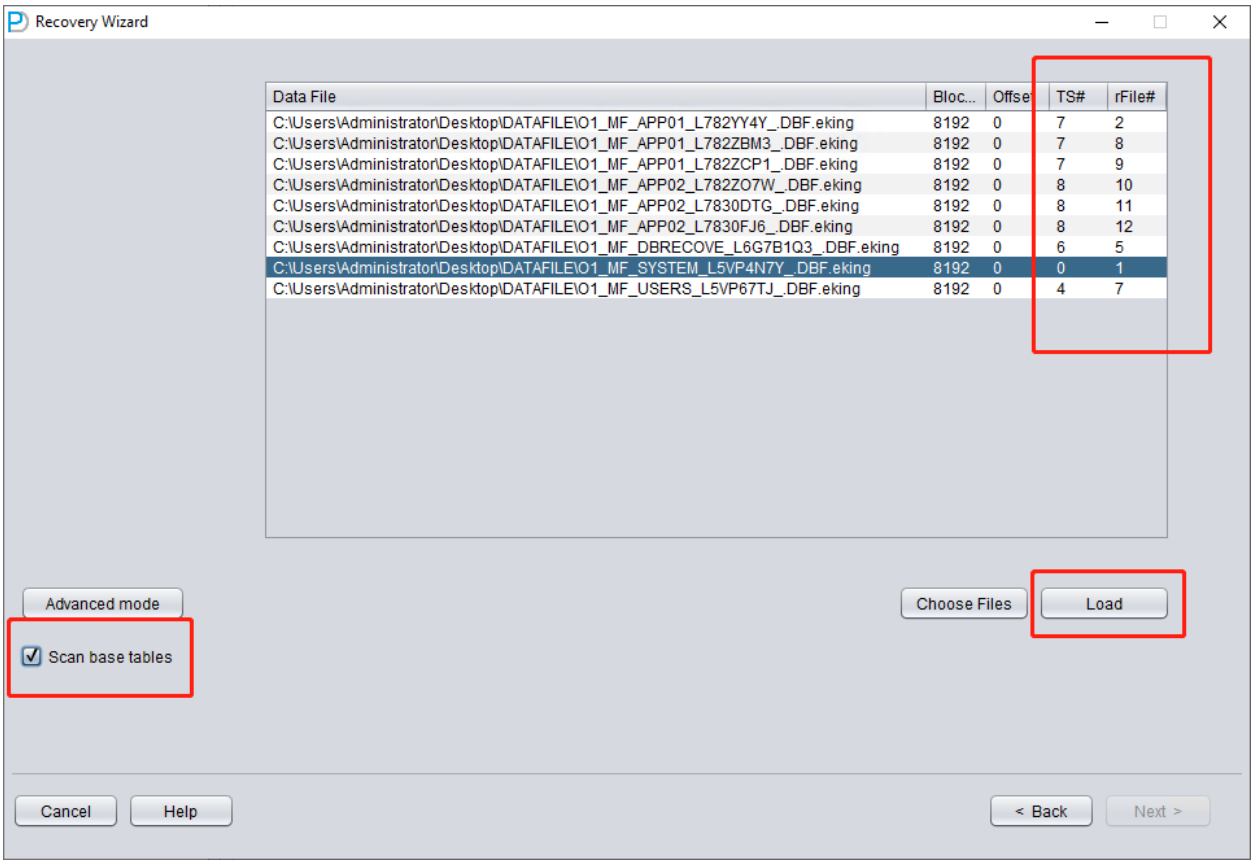
仍需选择数据库版本DB VERSION。


Recovery Wizard
Endian:
Little Endian
DB Character Set:
From dictionary
DB National Character Set:
From dictionary
Block Size:
8192
Offset:
0
DB Version:
12
Cancel
Help
< Back
Next >

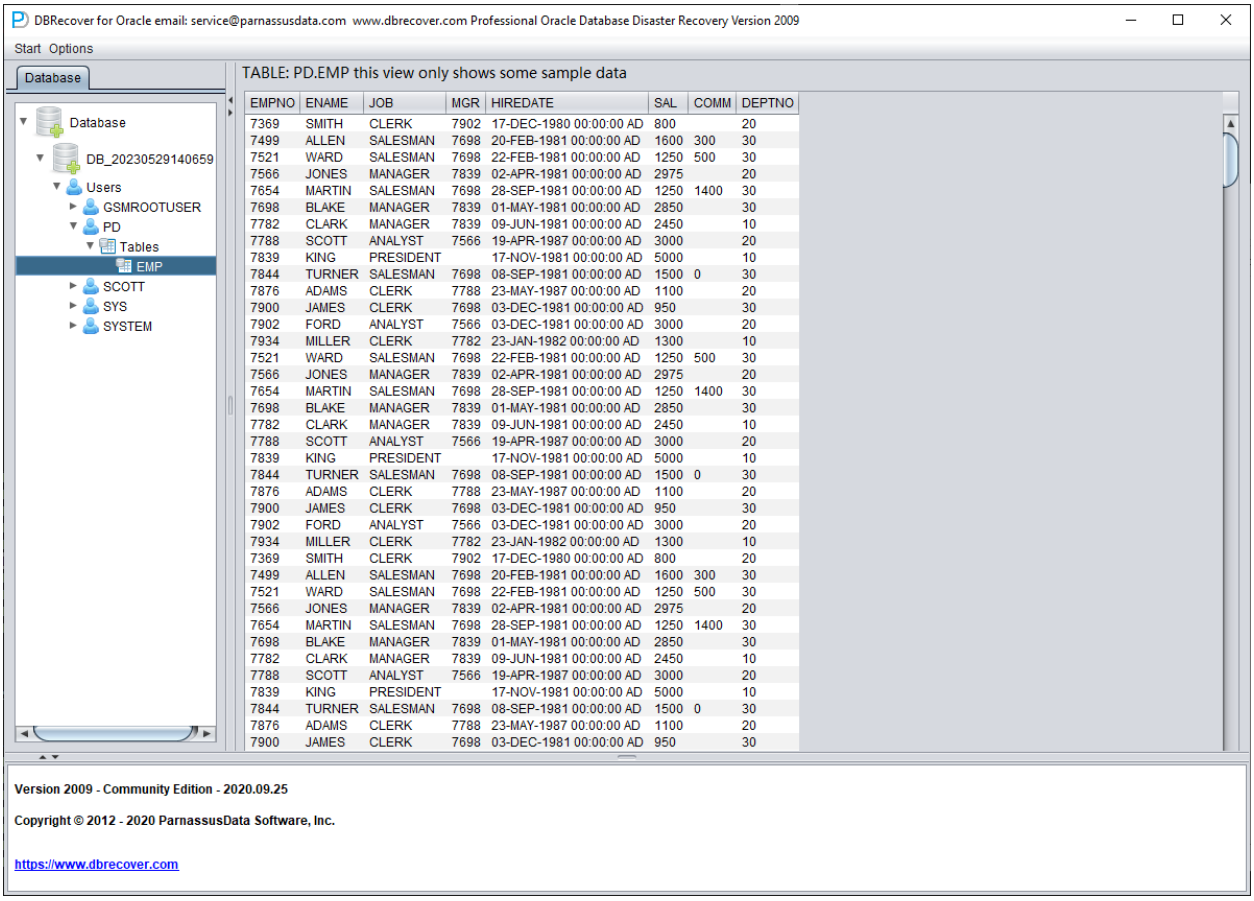


加入所有必要的数据文件（所有可能存放用户数据的文件文件，UNDOTBS1、TEMP、SYSAUX这些不用加入），注意不要漏掉SYSTEM01.DBF(必须加入)。

按照之前整理的表格，填写此处的TS#和RFILE#信息：



若正确填写相关信息，且加密损坏程度不高，则可以直接读取到数据：



由于加密病毒的特征各异，所以实际操作中可能需要更多问题。欢迎通过邮件与我们交流问题：service@parnassusdata.com

恢复场景4 误DELETE FROM TABLE删除数据行的恢复

D公司的开发人员将测试环境的删除数据脚本，误连接产品环境(PROD DATABASE)后执行；将某表上的数据全部DELETE删除了。

以上场景，我们可以通过DBRECOVER挖掘出已DELETE删除的行数据。

但需要用户先执行以下操作，以便最大程序保护数据不受覆盖：

1. 将表所在表空间设置为只读READ ONLY，命令为: ALTER TABLESPACE {TABLESPACE_NAME} READ ONLY
2. 将数据库实例停止：SHUTDOWN IMMEDIATE

以上两个方案，用户可以选择其中一个。

复现场景：

```
SQL> select count(*) from pd.emp;
COUNT(*)
---------
114688
SQL>
```

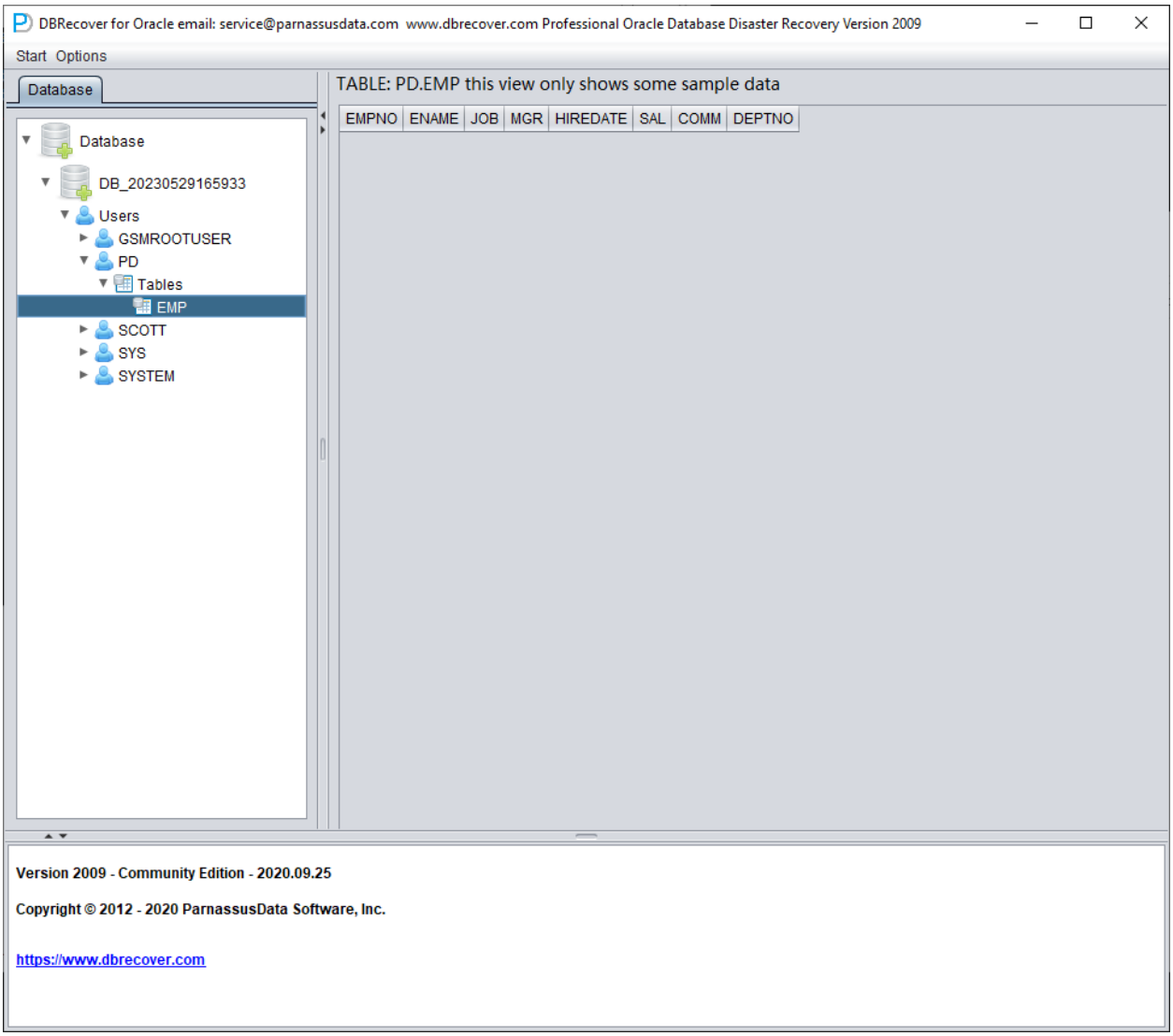
```
SQL> delete from pd.emp;
114688 rows deleted.
SQL> commit;
Commit complete.
SQL> alter system checkpoint;
System altered.
SQL> select count(*) from pd.emp;
COUNT(*)
---------
0
```

开始恢复前，我们先将表空间设置为只读，保护恢复环境：

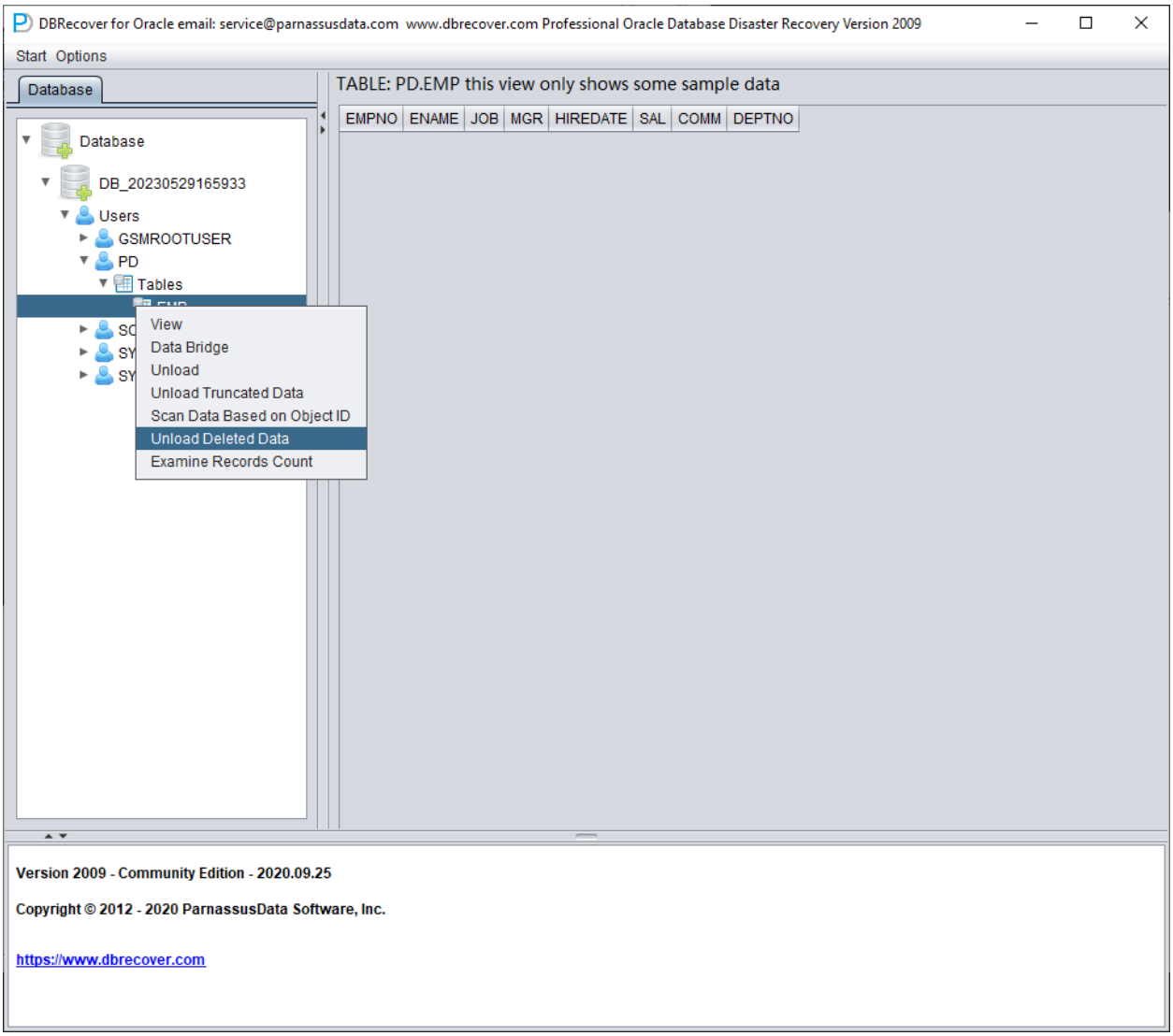
```
SQL> select tablespace_name from dba_segments where owner='PD' and segment_name='EMP';
TABLESPACE_NAME
------------------------------
```

```
DBRECOVER_TEST
SQL> alter tablespace DBRECOVER_TEST read only;
Tablespace altered.
```

启动DBRECOVER，选择字典模式，加入所有可用数据文件：



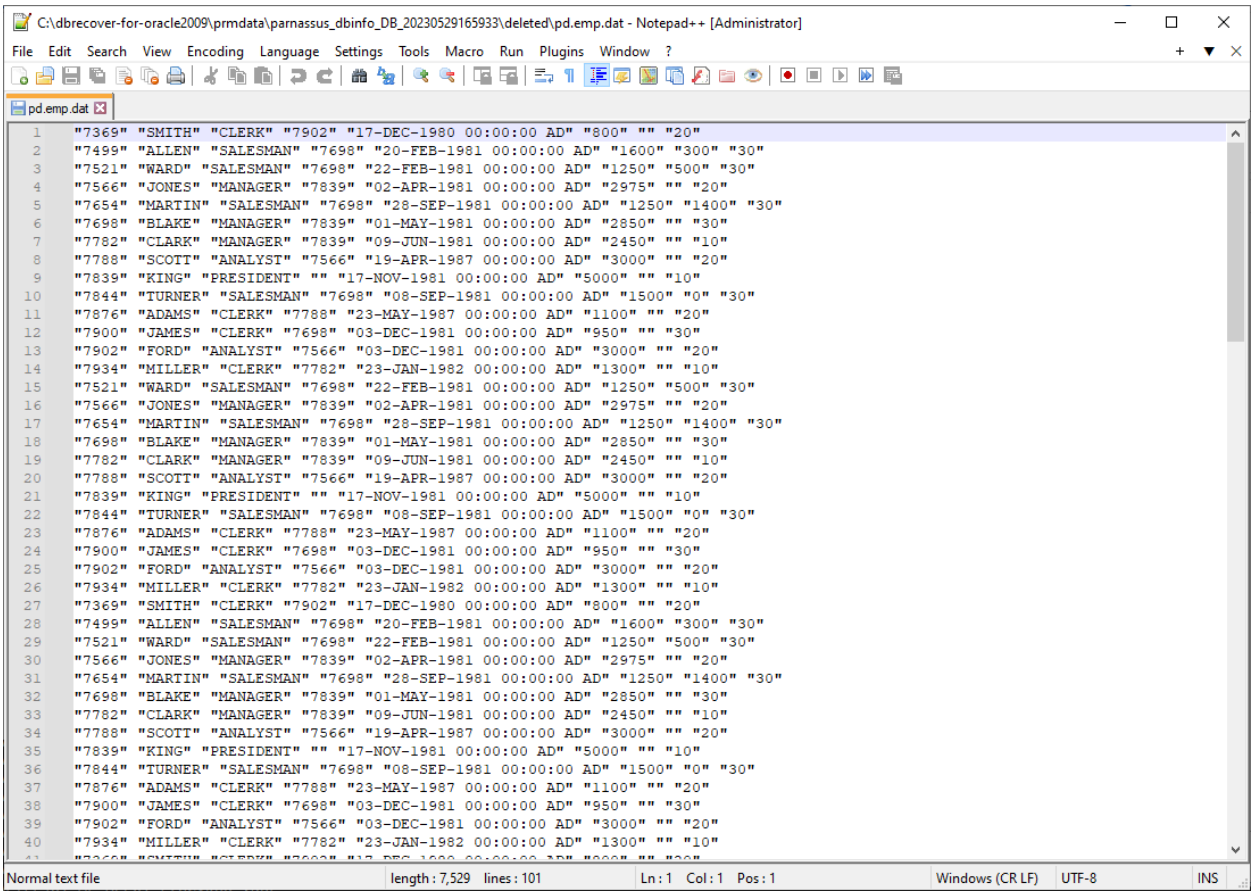
查看示例表的数据可以看到一样是空的。选中表右键Unload Deleted Data


DBRecover for Oracle email: service@parnassusdata.com www.dbrecover.com Professional Oracle Database Disaster Recovery Version 2009
Start Options
Database
TABLE: PD.EMP this view only shows some sample data
EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO
Database
DB_20230529165933
Users
GSMROOTUSER
PD
Tables
View
Data Bridge
Unload
Unload Truncated Data
Scan Data Based on Object ID
Unload Deleted Data
Examine Records Count
Version 2009 - Community Edition - 2020.09.25
Copyright © 2012 - 2020 ParnassusData Software, Inc.
https://www.dbrecover.com


DBRecover for Oracle
Unload successfully!
File path: C:\dbrecover-for-oracle2009\prmdata\parnassus_dbinfo_DB_20230529165933\deleted\pd.emp.dat
Unloaded row count: 100
Elapsed time(seconds): 0.0
To recover deleted records for Community Edition, row limitation is 100.
For Enterprise Edition, there is no row limitation.
If you need to recover more data, please contact service@parnassusdata.com
OK

在没有有效的企业版授权的情况下，UNLOAD DELETED DATA功能的限制是每张表100行数据。

挖掘出来的数据存放在弹出窗口所示路径下：



用户需自行检查该恢复的结果，并使用SQLLDR或SQLDEVELOPER等工具将文本数据插入到数据库中。

恢复场景5 误操作Truncate表的恢复

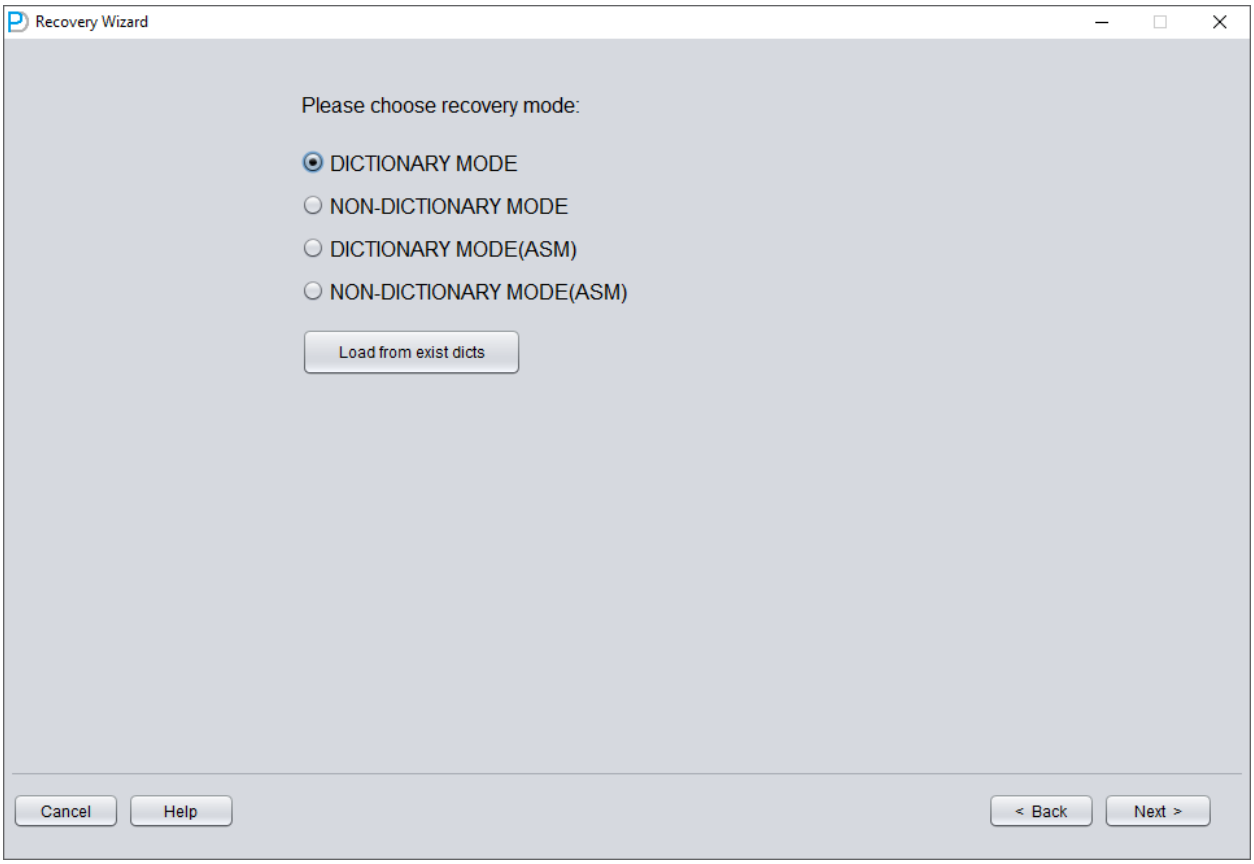
D公司的业务维护人员由于误将产品数据库当作测试环境数据库导致错误地TRUNCATE了一张表上的所有数据，DBA尝试恢复但是发觉最近的备份不可用，导致无法从备份中恢复出该数据表上的记录。 此时DBA决定采用DBRECOVER来恢复已经被TRUNCATE掉的数据。

由于该环境中 所有数据库文件均是可用且健康的，用户仅需要 字典模式下加载SYSTEM表空间的数据文件以及被TRUNCATED表的数据文件即可，例如：

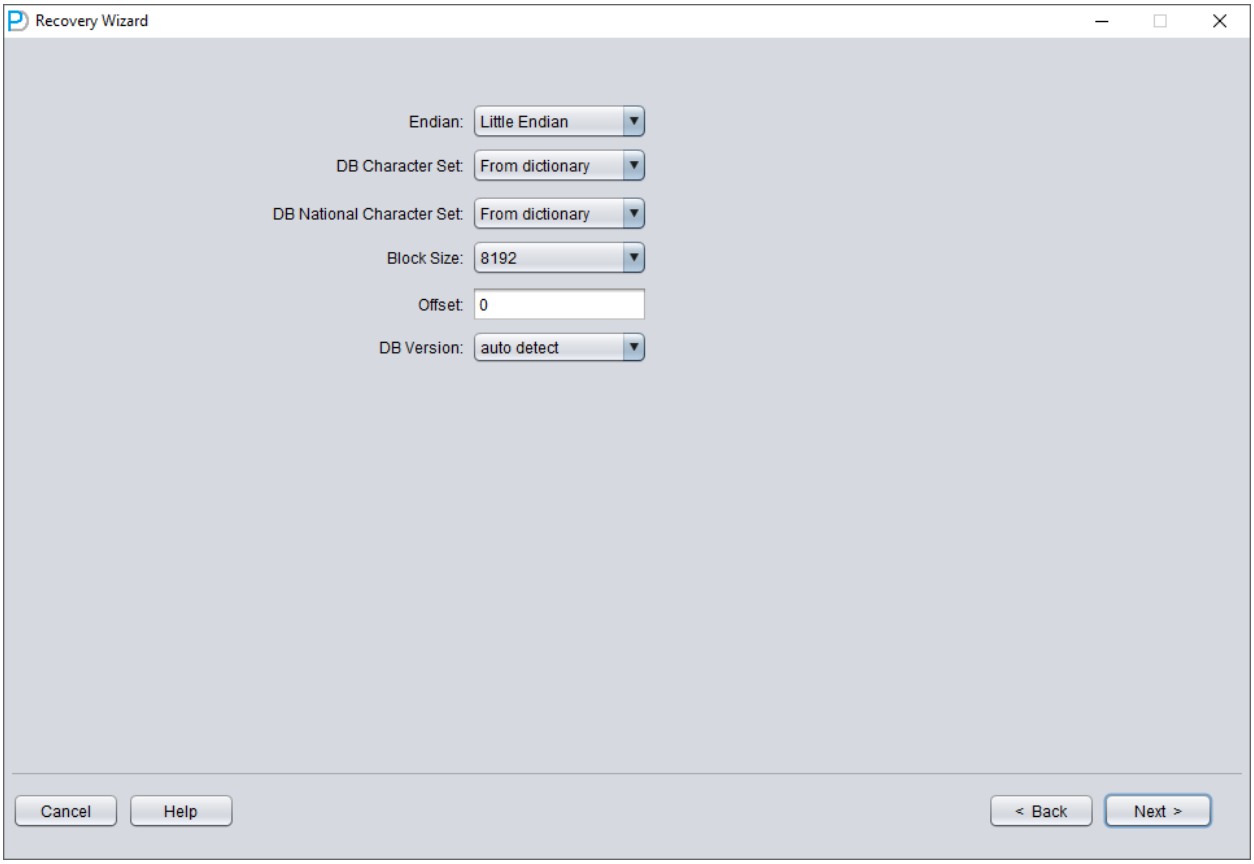
```
SQL> select count(*) From pd.salgrade;
COUNT(*)
---------
655360
SQL> select tablespace_name from dba_segments where owner='PD' and segment_name='SALGRADE';
TABLESPACE_NAME
------------------------------
APP01
SQL> truncate table pd.salgrade;
Table truncated.


SQL>
SQL> alter system checkpoint;
System altered.
SQL> select count(*) from pd.salgrade;
COUNT(*)
---------
0
```

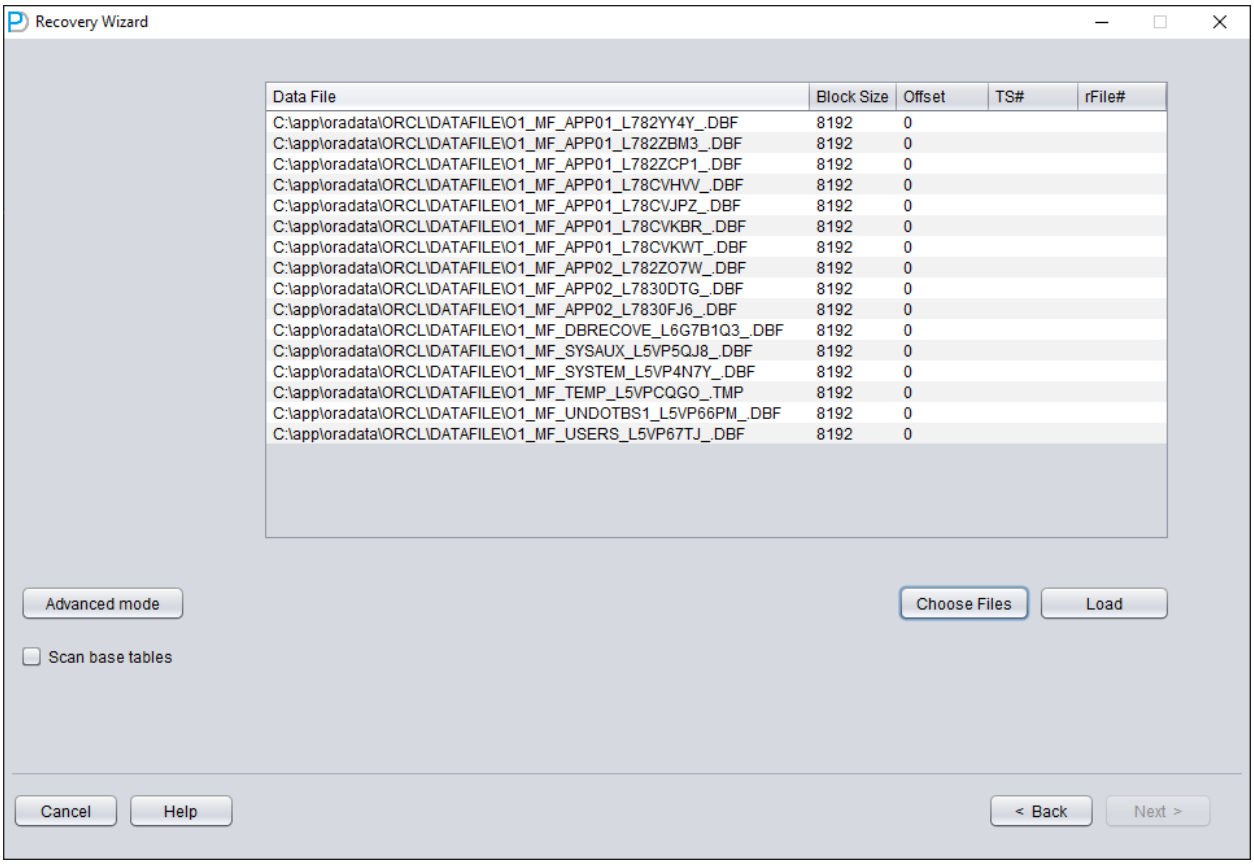
在此TRUNCATE场景中并未采用ASM存储，所以仅需要选择 《Dictionary Mode》字典模式即可：



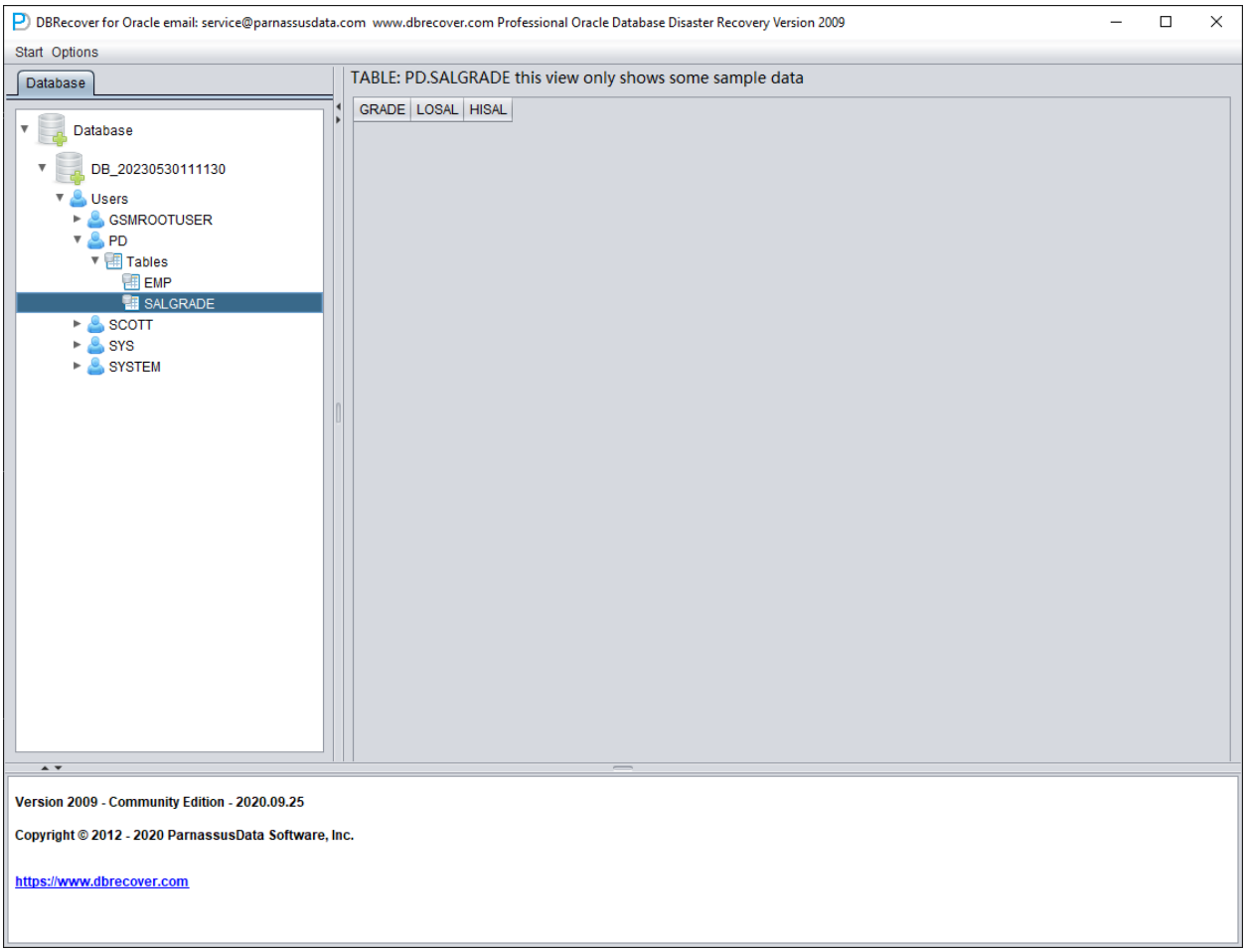
大部分情况下不需要修改任何参数：


Recovery Wizard
Endian:
Little Endian
DB Character Set:
From dictionary
DB National Character Set:
From dictionary
Block Size:
8192
Offset:
0
DB Version:
auto detect
Cancel
Help
< Back
Next >

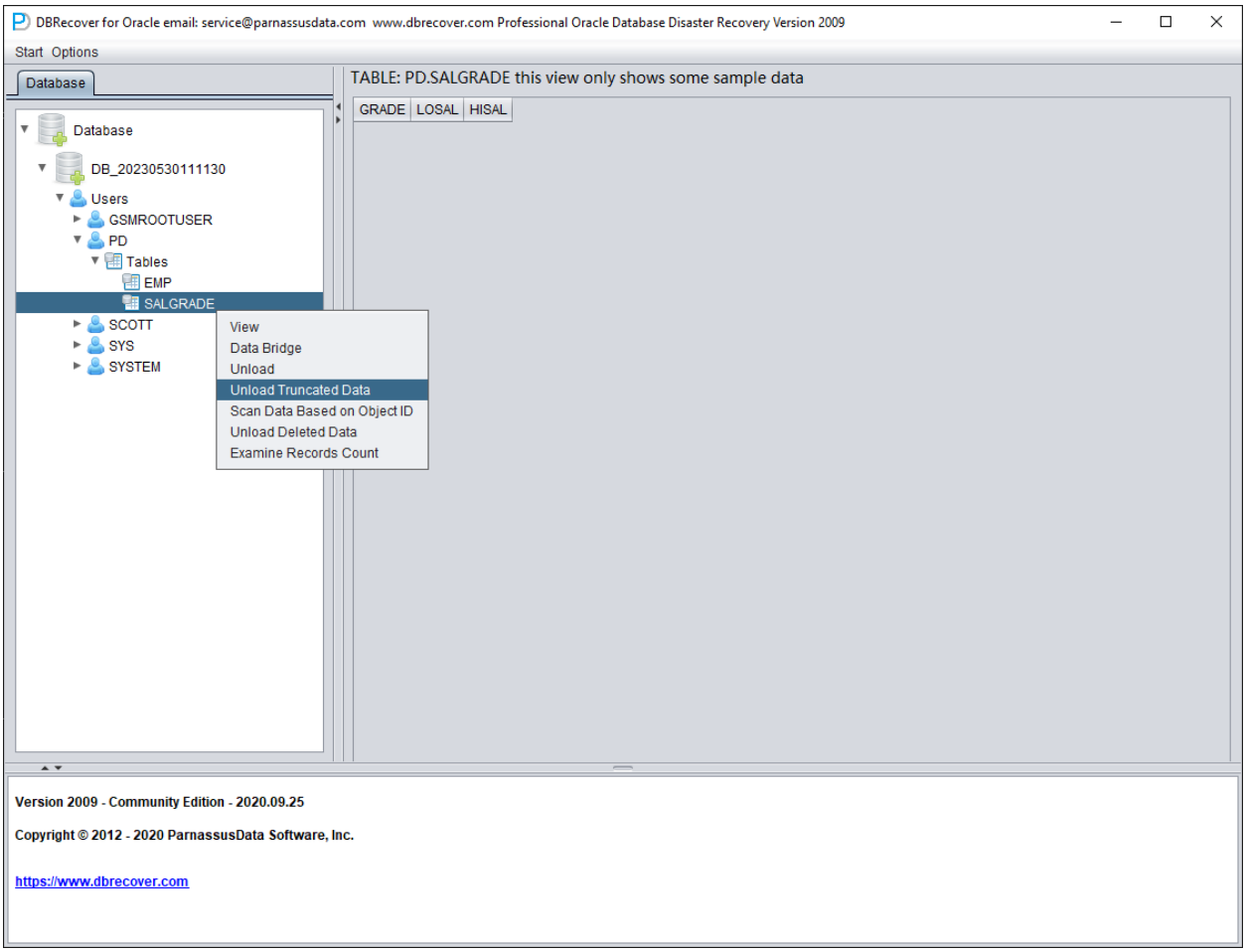
加入所有可用数据文件：

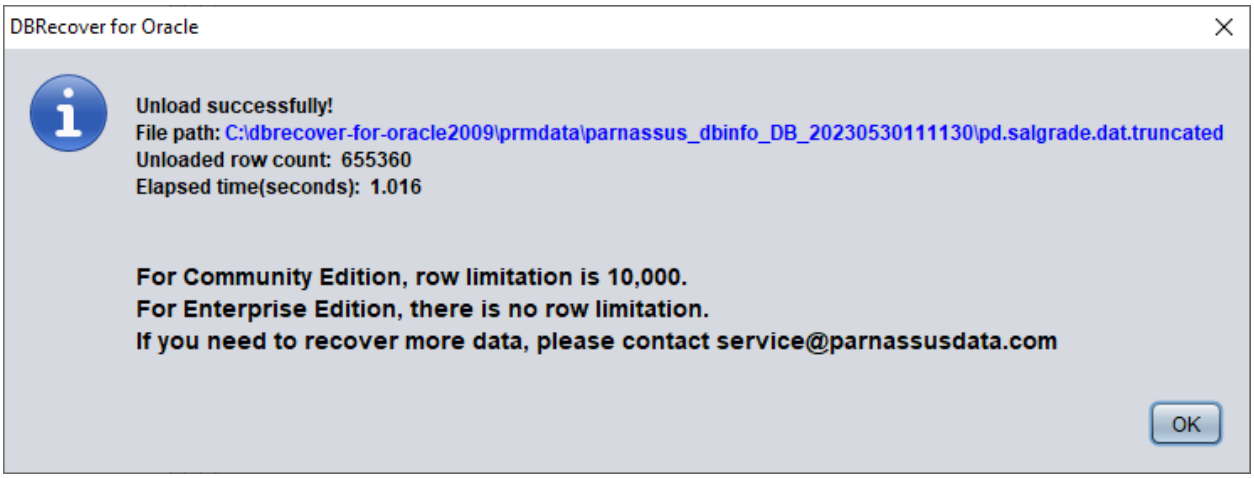


点开USERS，可以看到多个用户名，例如用户需要恢复PD SCHEMA下的一张表，则点开PD，并双击表名：



由于该表之前已经被TRUNCATED掉了，所以双击没有显示有数据，此时在表上右键选择Unload truncated data：





DBRECOVER将尝试扫描该表所在表空间并将已经truncated掉的数据抽取出来：如上图所示从已经被TRUNCATE过的表中抽取出完整的655360条记录，并存放在提示指定的路径下。

用户可以查看该DAT文件以确认恢复结果。

恢复被TRUNCATE掉的数据，其关键的一点是确认表在被TRUNCATE前的DATA_OBJECT_ID，如本例子中：

```
SQL> select object_id ,data_object_id from dba_objects where owner='PD' and object_name='SALGRADE';
OBJECT_ID DATA_OBJECT_ID
--------- --------------
76112 76113
```

在TRUNCATE发生前，该表的OBJECT_ID和DATA_OBJECT_ID均是76612。在TRUNCATE发生后DATA_OBJECT_ID发生了变化。

因此这里的原始DATA_OBJECT_ID是76612；但如果一张表本身被TRUNCATE过很多次，而你需要恢复最后一次TRUNCATE前的数据，则不能单纯使用OBJECT_ID来猜测原始的DATA_OBJECT_ID。

可以利用例如闪回查询、字典检索、日志挖掘等技术来确定DATA_OBJECT_ID；这里举一个闪回查询的例子：

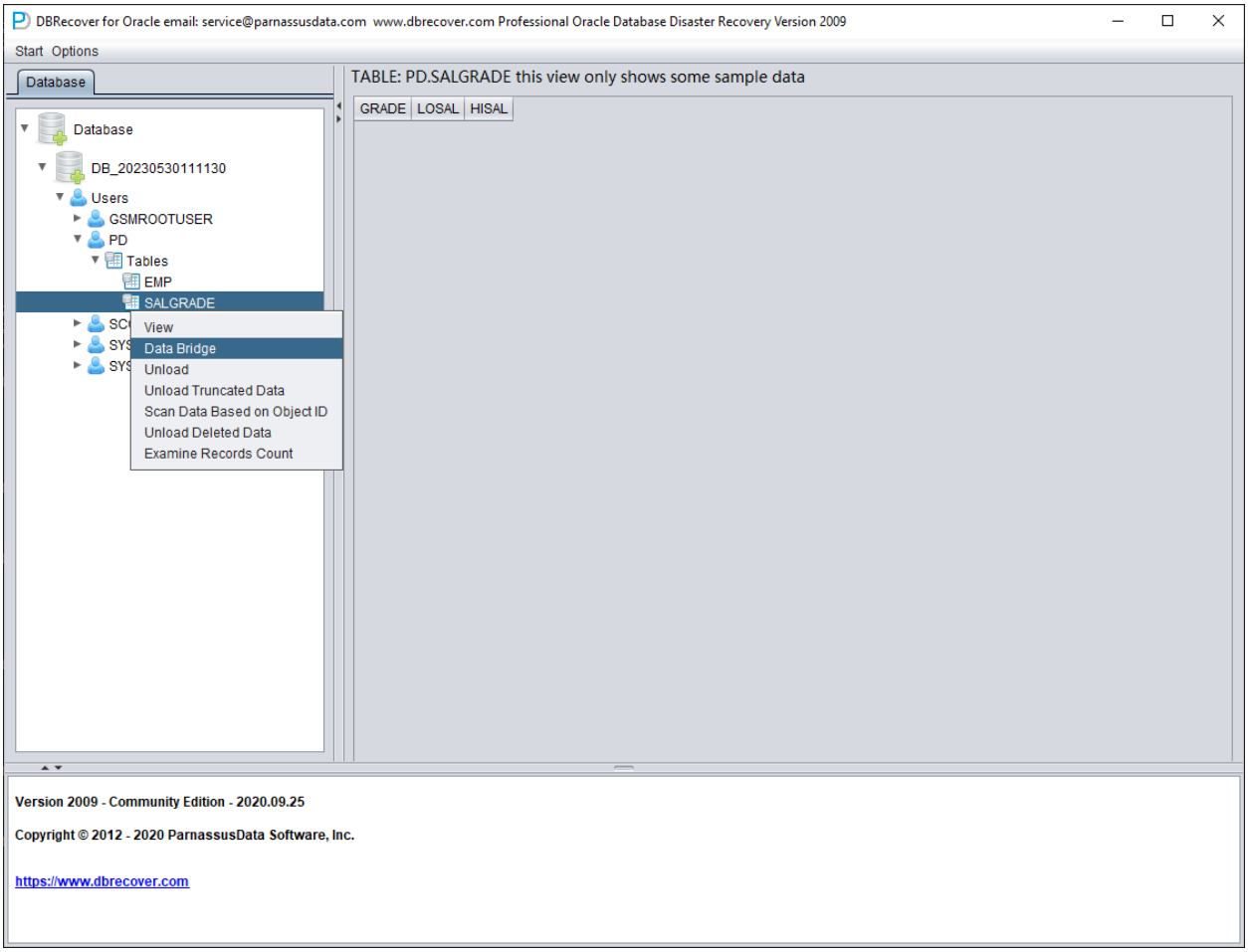
```
SQL> select user# from sys.user$ where name='PD';
USER#
---------
106
SQL> select obj#,dataobj# from sys.obj$ as of timestamp systimestamp -1/24 where name='SALGRADE' and owner#=106;
OBJ# DATAOBJ#
--------- ----------
76112 76112
```

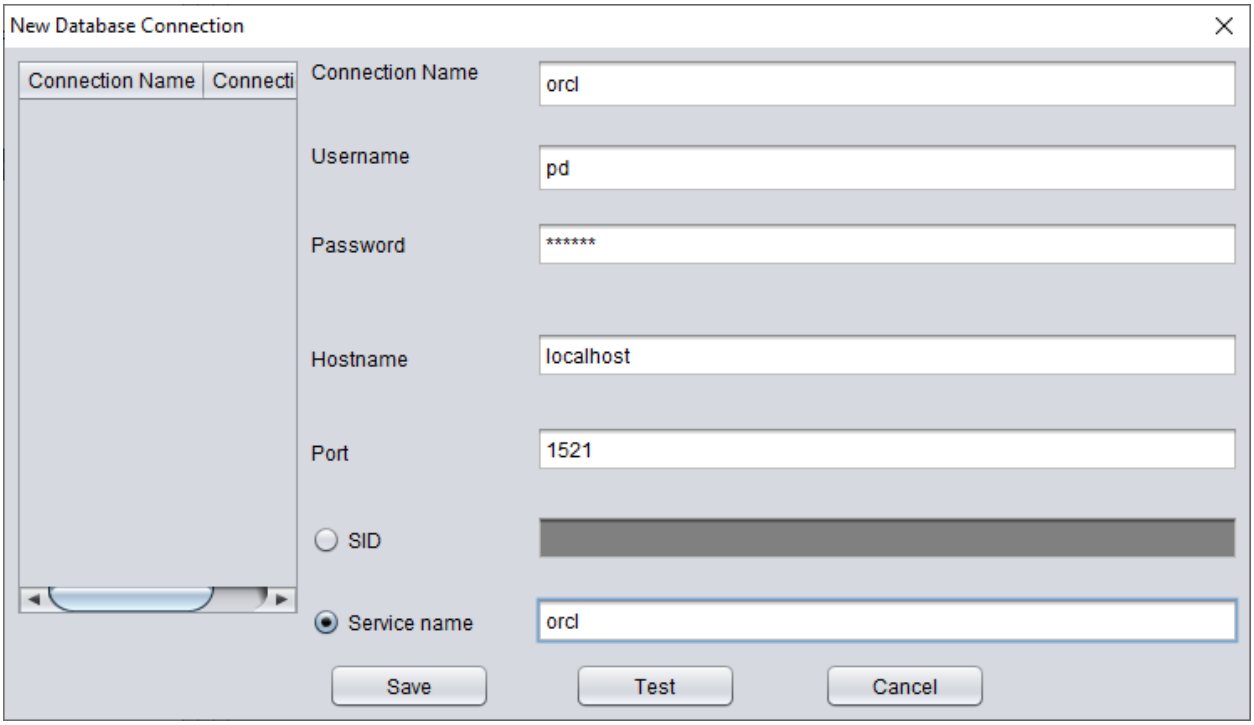
以上利用闪回查询获得了原始DATAOBJ#即DATA_OBJECT_ID。

之后我们需要利用Data Bridge特性将需要恢复的数据插入到目标库中。使用Data Bridge恢复truncate时的注意事项：注意当从源库中恢复出truncate数据时，若使用Data Bridge选项传输数据回到你的源库(如果回传数据不是到源库则没有该问题）时，需要注意 Data Bridge插入到新建表的所在位置应当不是源库中被truncate数据所在的表空间，同时避免插入到源表，否则会出现一边在恢复truncate数据一边我们所需恢复的数据被新数据所覆盖的问题，可能导致该恢复场景中的数据完全无法恢复。故请注意，当使用Data Bridge+恢复数据到源库时，在Data Bridge中指定表空间时千万不要使用需要恢复数据所在的表空间！！！！！！

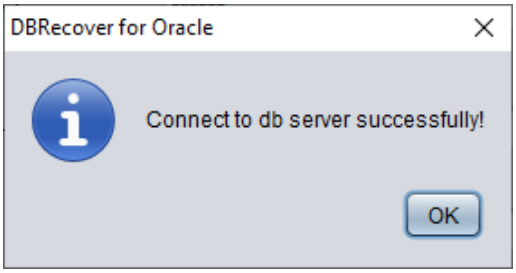
故我们在这里先创建一个新的表空间用以存放恢复的数据表：

```
SQL> create tablespace pd_recover_data datafile size 600M;
Tablespace created.
```

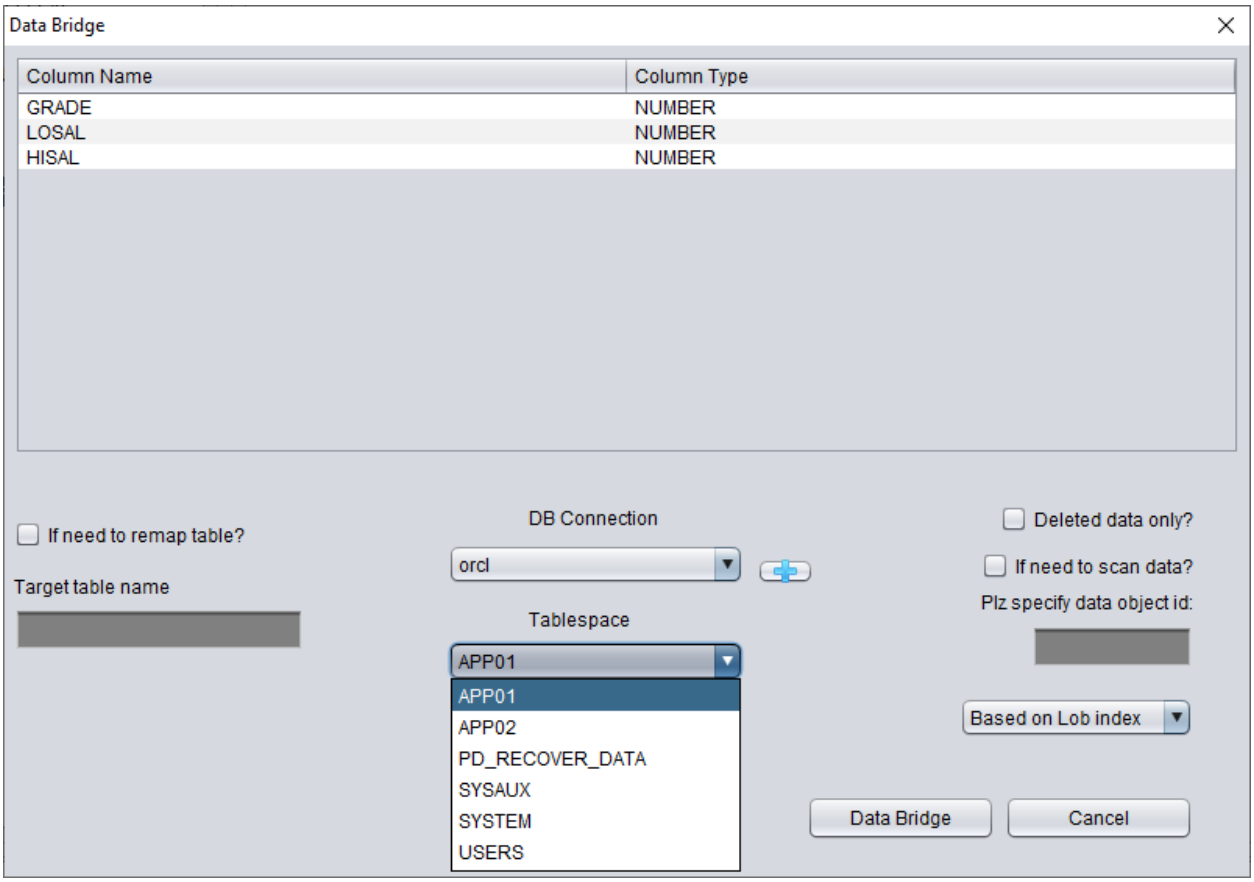




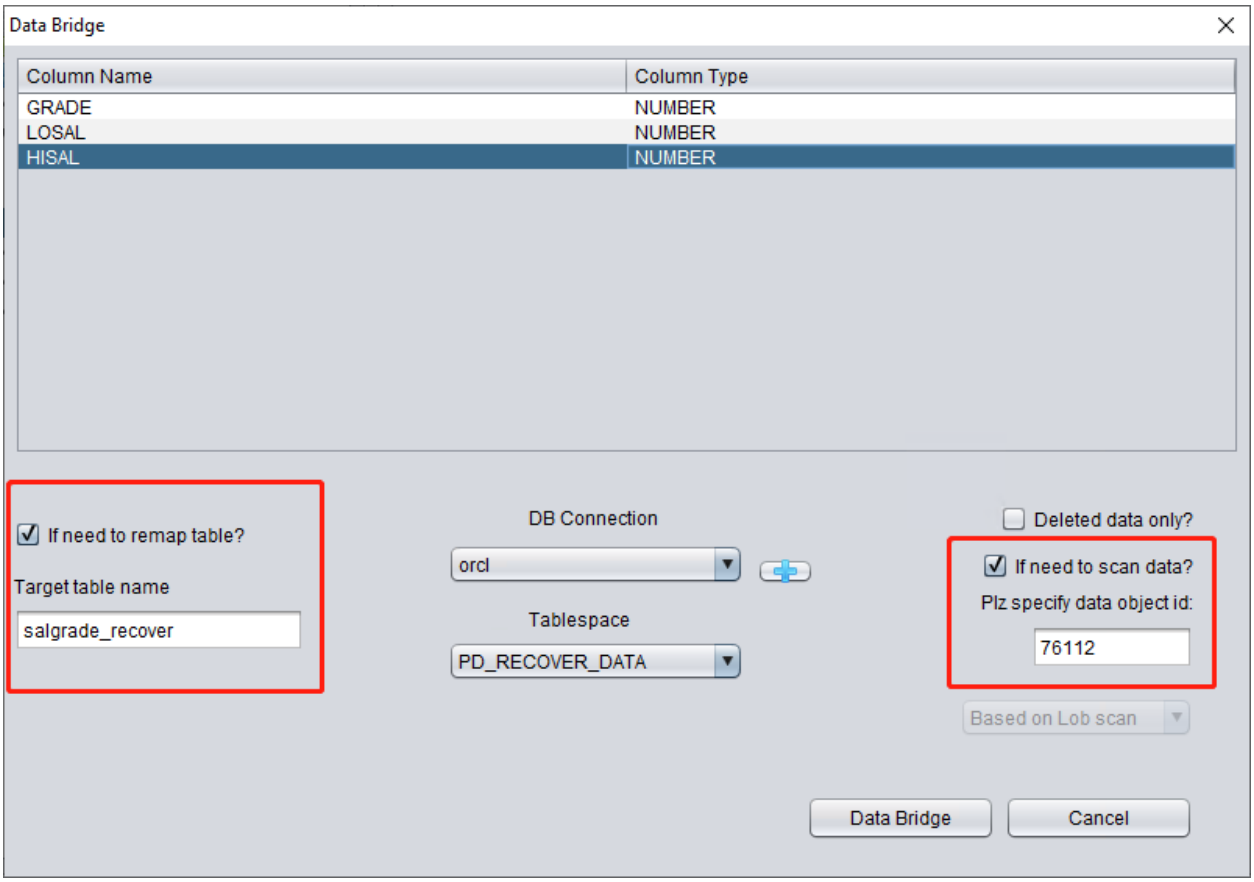
创建必要的登陆信息，注意数据库用户应当有必要的权限（建议授予DBA角色）。



TEST成功后点击SAVE保存。



以上选择用以存放恢复TRUNCATE的数据表的表空间。



这里我们要勾选”if need to scan data”并填入之前获得的原始DATA_OBJECT_ID，这样DBRECOVER会为我们明确扫描该ID对应的数据。

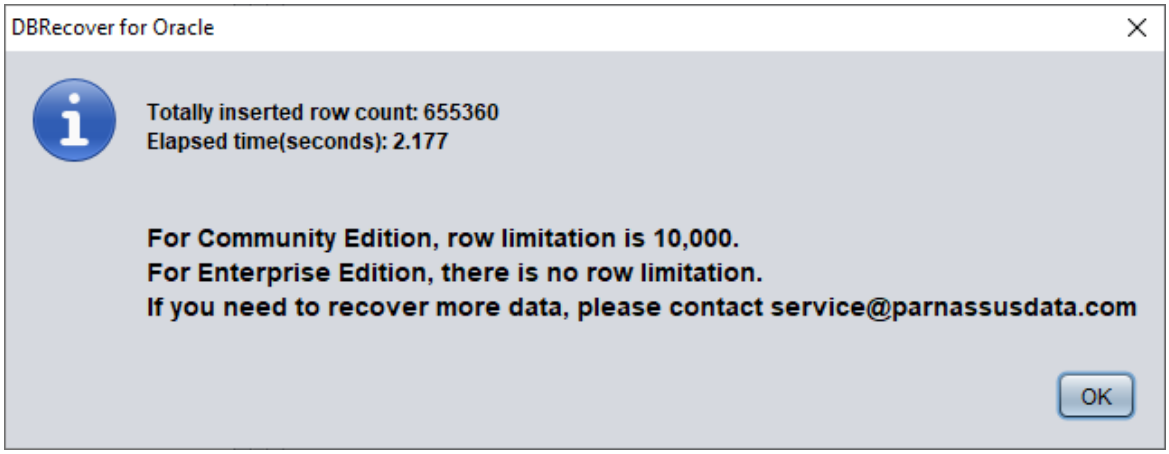
同时我们要勾选”if need to remap table”，并输入一个新表名。以便让数据插入到新表（在新的表空间下），排除任何覆盖数据的可能性。

注意：

1）对于目标库中已经存在对应表名的情况，DBRECOVER不会重建表而是会在现有表的基础上插入所需恢复的数据，由于表已经建立了所以指定的表空间将无效

2）对于目标数据库-SCHEMA中还不存在对应表名的情况，DBRECOVER会尝试在指定表空间上建表并插入恢复数据

完成上述步骤后点击Data Bridge按钮。



确认恢复的行数：

SQL> select count(*) from pd.salgrade_recover;

COUNT(*)

655360

Truncate数据的大致原理是，Truncate发生时ORACLE仅会在数据字典和Segment Header中更新表的Data Object ID，而实际数据部分的块则不会做修改。由于数据字典与段头的DATA_OBJECT_ID与后续的数据块中的并不一致，所以ORACLE服务进程在读取全表数据时不会读取到已经被TRUNCATE但是实际仍未被覆盖的数据。因此DBRECOVER可以通过未被修改覆盖的数据盘区(Data Extent)来恢复其中的数据。

恢复场景6 误操作Drop表的恢复

D公司的应用开发人员在没有任何备份的情况下DROP了系统中一张核心应用表，此时第一时间采用DBRECOVER可以恢复该DROP掉数据表的绝大部分数据。10g以后提供了 recyclebin回收站特性，可以首先通过查询DBA_RECYCLEBINS视图来确定被DROP掉的表是否在回收站中，如果在则优先通过回收站flashback to before drop，如果回收站中也没有了，则第一时间使用DBRECOVER恢复。

与TRUNCATE的恢复类似，DROP表的恢复需要确定原始DATA_OBJECT_ID。

恢复简要流程如下：

1. 首先将被DROP掉的数据表所在的表空间设置为只读模式(ALTER TABLESPACE {TABLESPACE_NAME} READ ONLY；或者第一时间将整个表空间所有数据文件复制一份
2. 通过查询数据字典或者LOGMINER找到被DROP掉数据表的DATA_OBJECT_ID
3. 在NON-DICT非字典模式下启动DBRECOVER，并加入被DROP掉数据表所在的表空间的所有数据文件，之后SCAN DATABASE+SCAN TABLE from Extent MAP
4. 通过DATA_OBJECT_ID定位到展开对象树形图中对应的数据表，采用Data Bridge模式插回到源数据库中

可以通过logminer 得到大致的DATA_OBJECT_ID，使用LOGMINER则大致的脚本如下：

```
EXECUTE DBMS_LOGMNR.ADD_LOGFILE( LOGFILENAME => '/oracle/logs/log1.f', OPTIONS => DBMS_LOGMNR.NEW);
EXECUTE DBMS_LOGMNR.ADD_LOGFILE( LOGFILENAME => '/oracle/logs/log2.f', OPTIONS => DBMS_LOGMNR.ADDFILE);
Execute DBMS_LOGMNR.START_LOGMNR(DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG+DBMS_LOGMNR.COMMITTED_DATA_ONLY);
SELECT * FROM V$LOGMNR_CONTENTS ;
EXECUTE DBMS_LOGMNR.END_LOGMNR;
```

也可以尝试通过挖掘AWR数据来获得DATA_OBJECT_ID：

```
Select * from
(select object_name,object# from DBA_HIST_SQL_PLAN
UNION select object_name,object# from GV$SQL_PLAN) V1 where V1.OBJECT# IS
```

```
NOT NULL minus select name,obj# from sys.obj$;
```

```
select obj#,dataobj#, object_name from WRH$_SEG_STAT_OBJ where object_name
not in (select name from sys.obJ$) order by object_name desc;
```

```
SELECT tab1.SQL_ID,
current_obj#,
tab2.sql_text
FROM DBA_HIST_ACTIVE_SESS_HISTORY tab1,
dba_hist_sqltext tab2
WHERE tab1.current_obj# NOT IN
(SELECT obj# FROM sys.obj$
)
AND current_obj#!=-1
AND tab1.sql_id =tab2.sql_id(+);
```

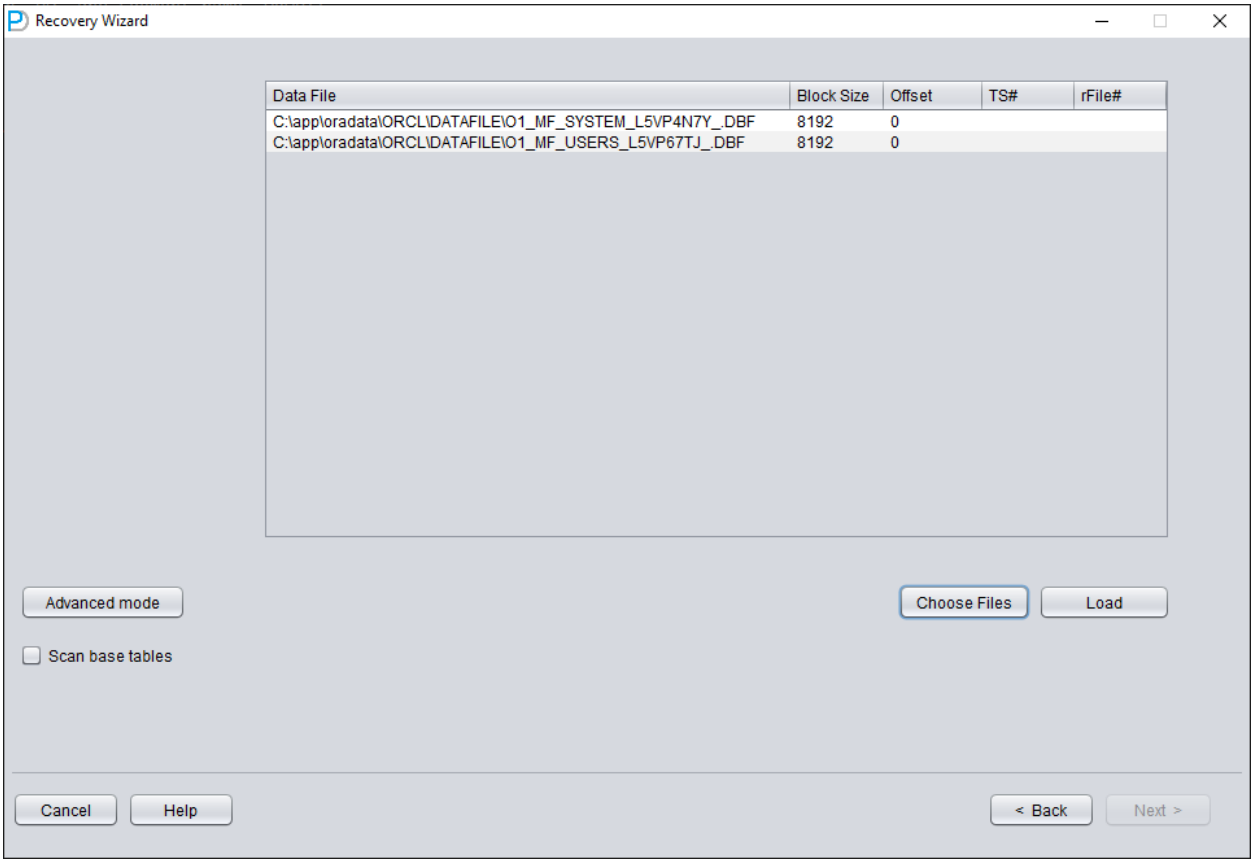
//以上三个查询通过比对AWR数据与OBJ$字典基表，来发现被DROP的表

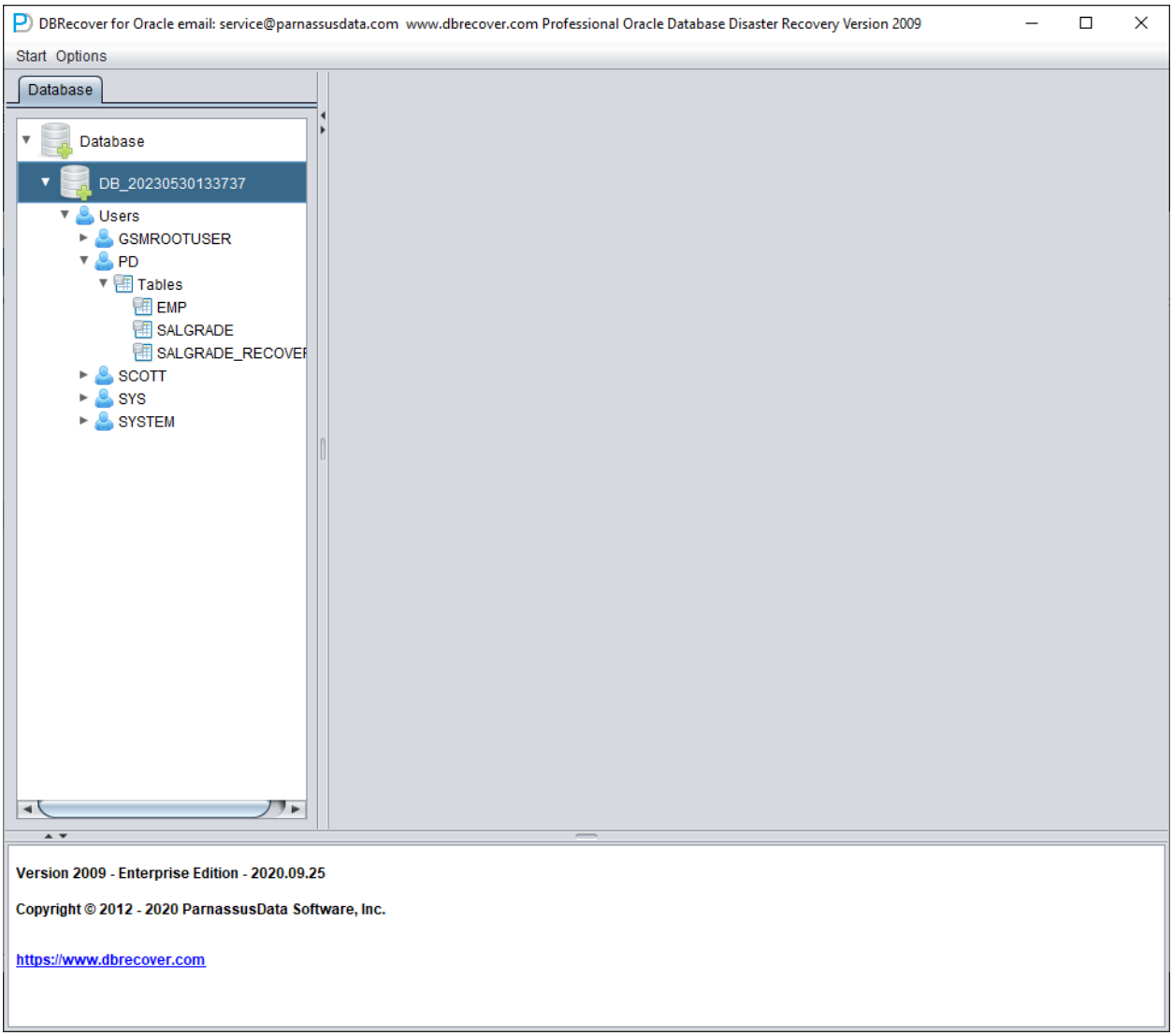
我们来实际演示一遍：

```
SQL> create table dropit as select * from dba_objects;
Table created.
SQL> select count(*) from pd.dropit;
COUNT(*)
---------
73095
SQL> select tablespace_name from dba_segments where owner='PD' and segment_name='DROPIT';
TABLESPACE_NAME
------------------------------
USERS
SQL> select object_id ,data_object_id from dba_objects where owner='PD' and object_name='DROPIT';
OBJECT_ID DATA_OBJECT_ID
--------- --------------
76116 76116
SQL> drop table dropit;
Table dropped.
```

```
SQL> alter system checkpoint;
System altered.
```

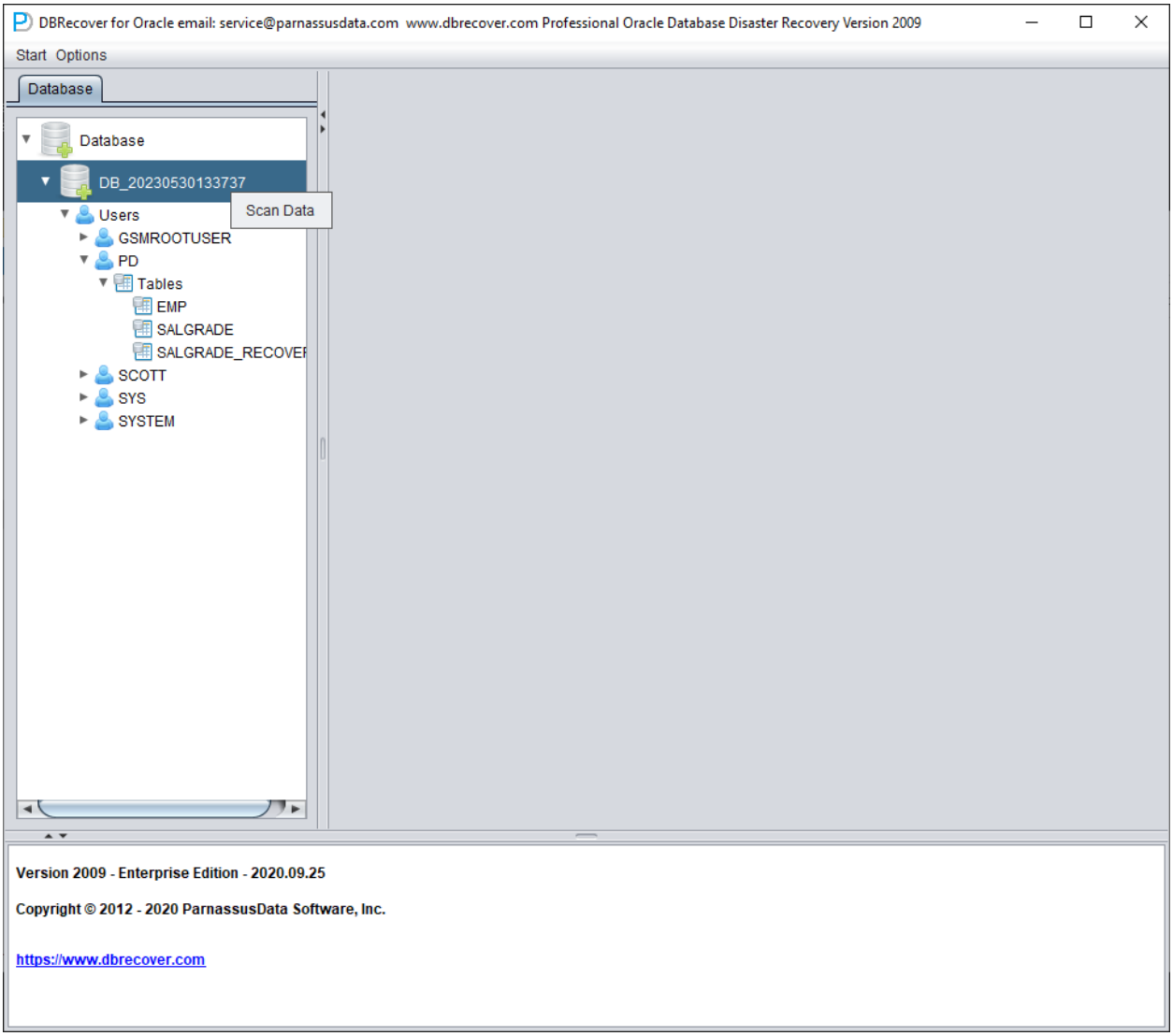
我们将DBRECOVER以字典模式(DICTIONARY-MODE)启动，这里仅需要加入SYSTEM01.DBF和表所在USERS表空间：

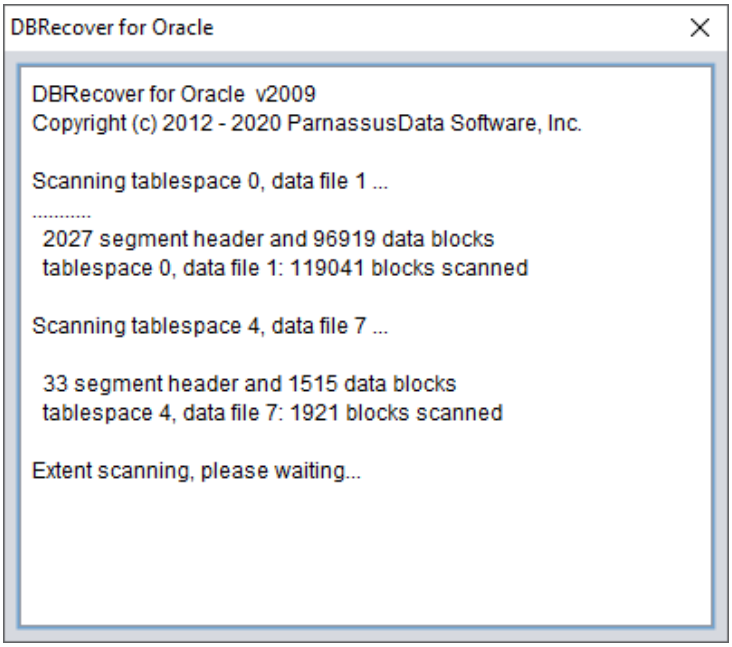


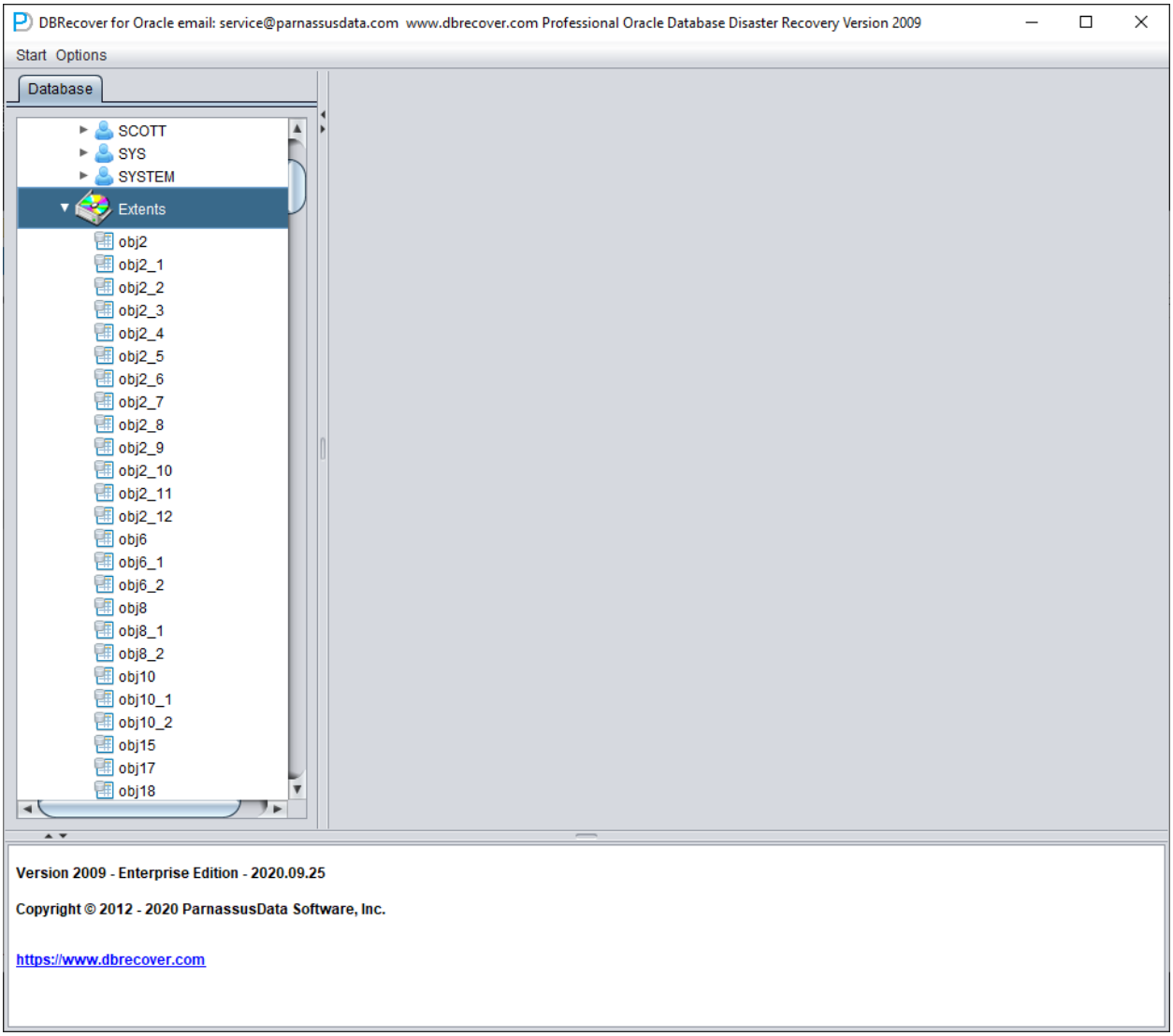


加载完成后可以发现PD SCHEMA下并不存在我们要恢复的表，这是正常的。

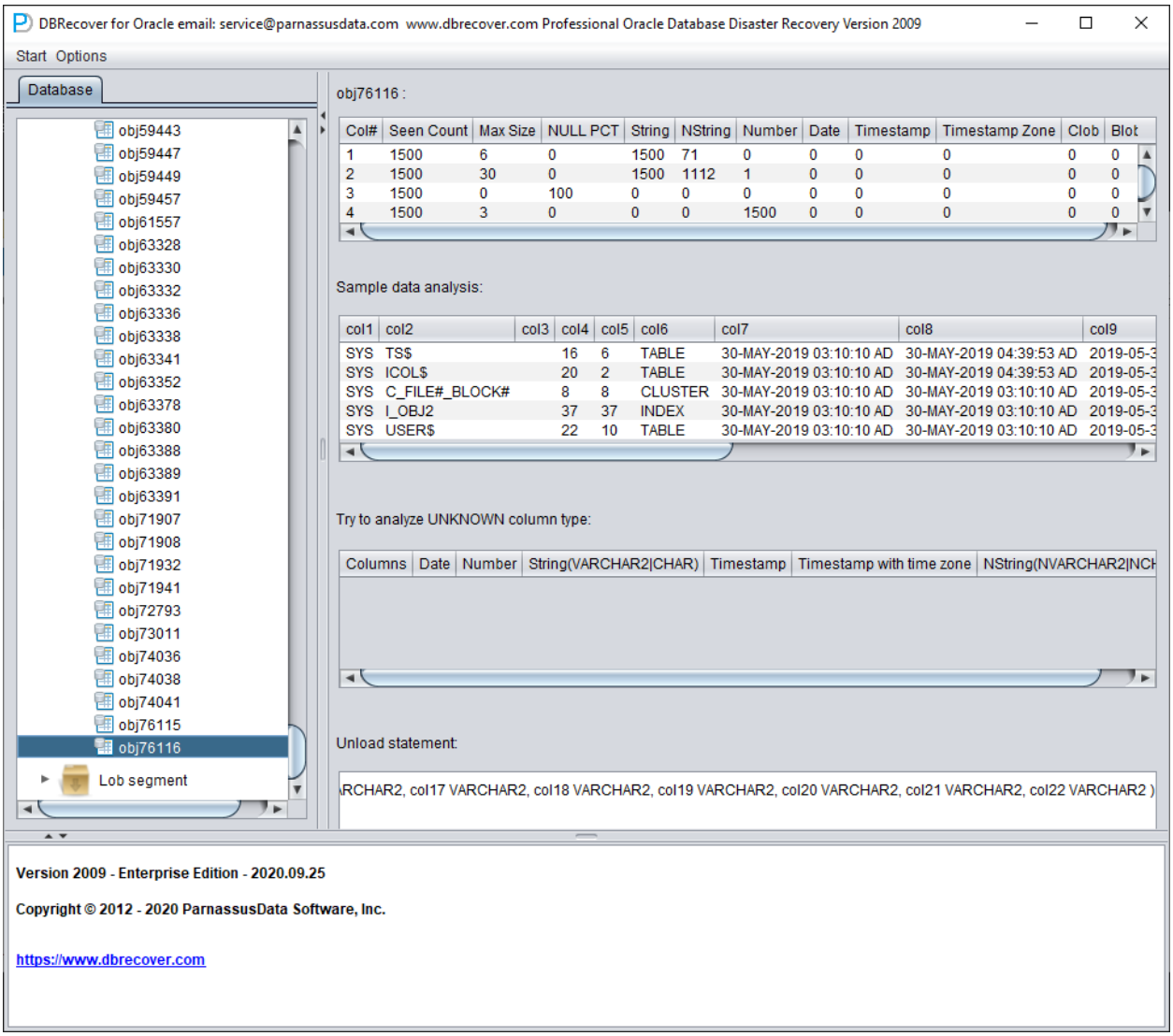
选中数据库节点，右键SCAN Data


DBRecover for Oracle email: service@parnassusdata.com www.dbrecover.com Professional Oracle Database Disaster Recovery Version 2009
Start Options
Database
Database
DB_20230530133737
Scan Data
Users
GSMROOTUSER
PD
Tables
EMP
SALGRADE
SALGRADE_RECOVER
SCOTT
SYS
SYSTEM
Version 2009 - Enterprise Edition - 2020.09.25
Copyright © 2012 - 2020 ParnassusData Software, Inc.
https://www.dbrecover.com


DBRecover for Oracle
DBRecover for Oracle v2009
Copyright (c) 2012 - 2020 ParnassusData Software, Inc.
Scanning tablespace 0, data file 1 ...
.....
2027 segment header and 96919 data blocks
tablespace 0, data file 1: 119041 blocks scanned
Scanning tablespace 4, data file 7 ...
33 segment header and 1515 data blocks
tablespace 4, data file 7: 1921 blocks scanned
Extent scanning, please waiting...



之后会出现一个EXTENTS节点，查找OBJ76116节点：



之后我们再利用Data Bridge特性将其插回源库即可。

恢复场景7 误操作DROP TABLESPACE的恢复

D公司的员工需要删除某个无用的表空间即DROP TABLESPACE INCLUDING CONTENTS操作，但是在操作DROP TABLESPACE后，开发部门反映该被DROP掉的TABLESPACE上其实有一个SCHEMA的数据是有用且重要的，但现在表空间被DROP了，且无任何备份。

此时可以利用DBRECOVER的非字典模式(NON-DICTIONARY)模式去抽取被DROP TABLESPACE的对应的所有数据文件中的数据。 通过这种方式可以恢复大部分数据，但由于是非字典模式所以需要将恢复出来的表与应用数据表一一对应起来，此时一般需要应用开发维护人员介

入，通过人工识别来分辨哪些数据属于哪张表。由于DROP TABLESPACE操作修改了数据字典，并在OBJ$中删除了对应表空间上的对象，所以无法从OBJ$上获得DATA_OBJECT_ID与OBJECT_NAME之间的对应关系。此时我们可以使用DROP TABLE场景中介绍的方法，尽可能多得获取DATA_OBJECT_ID与OBJECT_NAME之间的对应关系。

其大致流程如下：

若DROP TABLESPACE时数据文件也被物理删除了，则需要先恢复数据文件。可以通过文件系统级别的恢复软件尝试，或者使用PRMSCAN软件来从ORACLE数据块级别扫描重组数据文件。

PRMSCAN是ORACLE数据块碎片扫描合并工具，其适用于以下的场景：

1. 误手动删除了文件系统（任意文件系统 NTFS、FAT、EXT、UFS、JFS等）或ASM上的数据文件
2. 文件系统损坏，导致数据文件大小变成0 bytes即数据文件被清零
3. 文件系统损坏，导致文件系统无法MOUNT加载
4. ASM存储元数据损坏，导致diskgroup无法mount加载
5. 文件系统或ASM其中的LV或PV被物理破坏或丢失
6. 以上场景均可以利用prmscan直接扫描文件系统或ASM对应的 PV、LV 中的残余未被覆盖的oracle block，来实现对这些oracle数据块的合并重组，以达到数据恢复的目的。

PRMSCAN是基于JAVA语言开发的，可以跨一切支持JDK 1.6以后操作系统，包括Windows、Linux、Solaris、AIX、HP-UX。

目前该产品不提供零售，可以联系我们以提供恢复服务。

例如下面的例子中/dev/sdb1为ext4文件系统的分区，但是由于ext4文件系统损坏，导致SDB1无法被MOUNT，但该文件系统上存放了一套oracle数据库的数据文件，若无法MOUNT文件系统则oracle数据库也将无法使用。

这里我们使用prmscan的扫描oracle数据文件块和合并功能，从损坏的文件系统中直接将数据文件都重组出来。

扫描整个磁盘:

```
[oracle@dbdao01 ~]$ java -jar PRMScan.jar –scan /dev/sdb1 –guess 8k
–scan 选项代表扫描 /dev/sdb1 设备，并指定Oracle blocksize 为8k
[oracle@dbdao01 ~]$ java -jar PRMScan.jar –outputsh ./8kfull.txt
–outputsh 代表写出一个可以合并已扫描到信息的SHELL文件 即这里的8kfull.txt
[oracle@dbdao01 ~]$ sh 8kfull.txt
执行8kfull.txt即可以 在当前目录下生成所有需要合并的数据文件
如下
[oracle@dbdao01 ~]$ ls -ll PD*
rw-r–r– 1 oracle oinstall 295428096 Jul 28 00:37 PD_DBF1.dbf
rw-r–r– 1 oracle oinstall 83427328 Jul 28 00:37 PD_DBF2.dbf
rw-r–r– 1 oracle oinstall 220266496 Jul 28 00:37 PD_DBF3.dbf
rw-r–r– 1 oracle oinstall 1324482560 Jul 28 00:38 PD_DBF4.dbf
```

若数据文件未被物理删除，则可以直接以非字典模式(NON-DICTIONARY MODE)加入DBRECOVER后扫描其中数据。

后续步骤可以参考上文DROP TABLE的操作，区别在于DROP TABLESPACE的恢复对象会是很多张表。