从 Oracle 迁移到 DM

原有的系统或开发习惯为 Oracle 的用户，可以参考文档《DM DBA 手记之 ORACLE 移植到 DM》。

DTS 迁移前应注意检查的相关事项

- 迁移前根据实际需求确定好相关的数据库版本；
- 选择合适的 jdk 版本。涉及到更换 jdk 或者 jdbc 包等情况，可能会出现 unsupported major.minor versio n 52.0 的错误提示，这个是由于使用的 jar 编译环境版本更高，而当前 jdk 版本低导致的；
- 注意迁移的顺序。迁移时应先迁移序列、再迁移表、最后迁移视图、自定义类型、类、函数、存储过程、包等；
- 对于数据量大的表单独迁移；
- 对于大字段较多的表，需要修改批量迁移的行数，以免造成迁移工具内存溢出；
- 迁移完成后，一定要进行验证。

如何修改兼容性参数，使达梦对 ORACLE 语法的兼容性更高

【问题解答】：

DM 提供了 COMPATIBLE_MODE 参数来设置数据库的兼容性模式，0：不兼容，1：兼容 SQL92 标准，2：部分兼容 ORACLE，3：部分兼容 MS SQL SERVER，4：部分兼容 MYSQL， 5：兼容 DM6，6：部分兼容 TERA DATA。该参数默认为 0，修改方式有两种：

- 在 dm.ini 文件中修改 COMPATIBLE_MODE 参数的值。
  + 利用 SQL 语句修改 COMPATIBLE_MODE 参数的值。

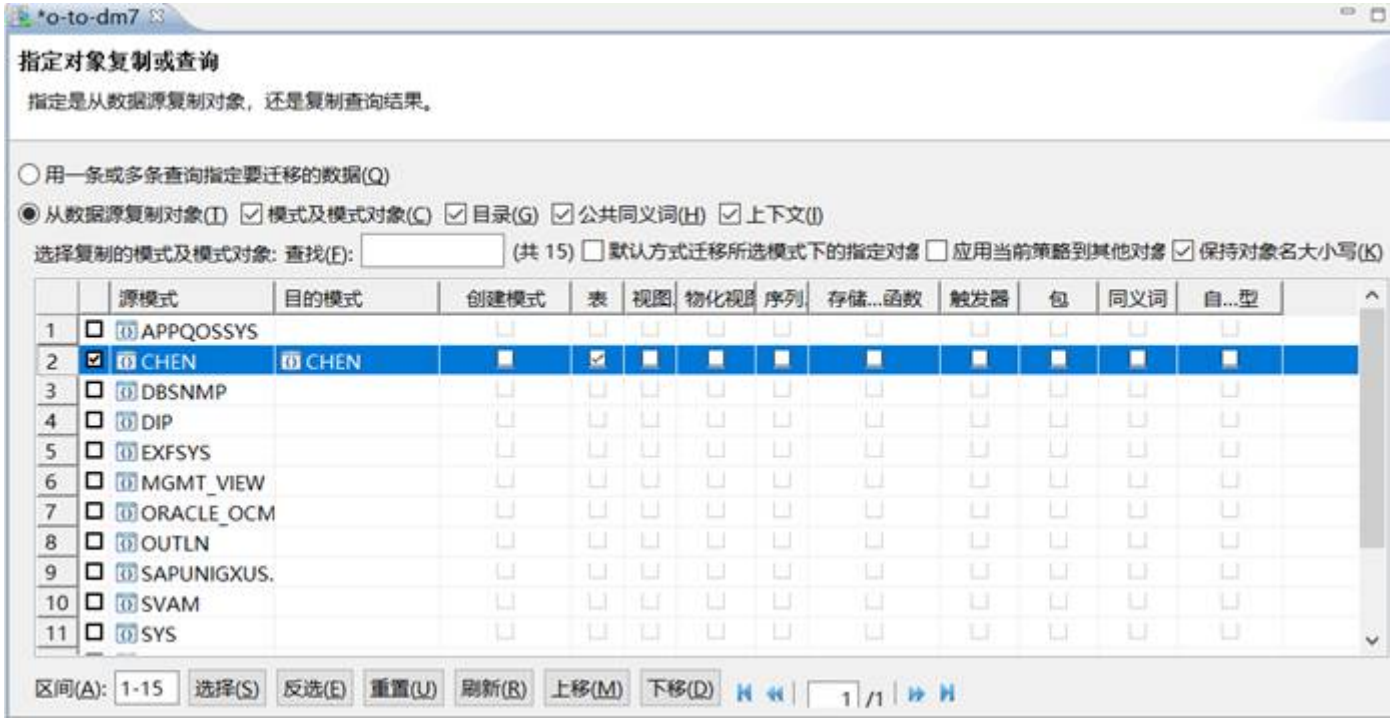
```
SP_SET_PARA_VALUE(2,'COMPATIBLE_MODE',2);
或
```

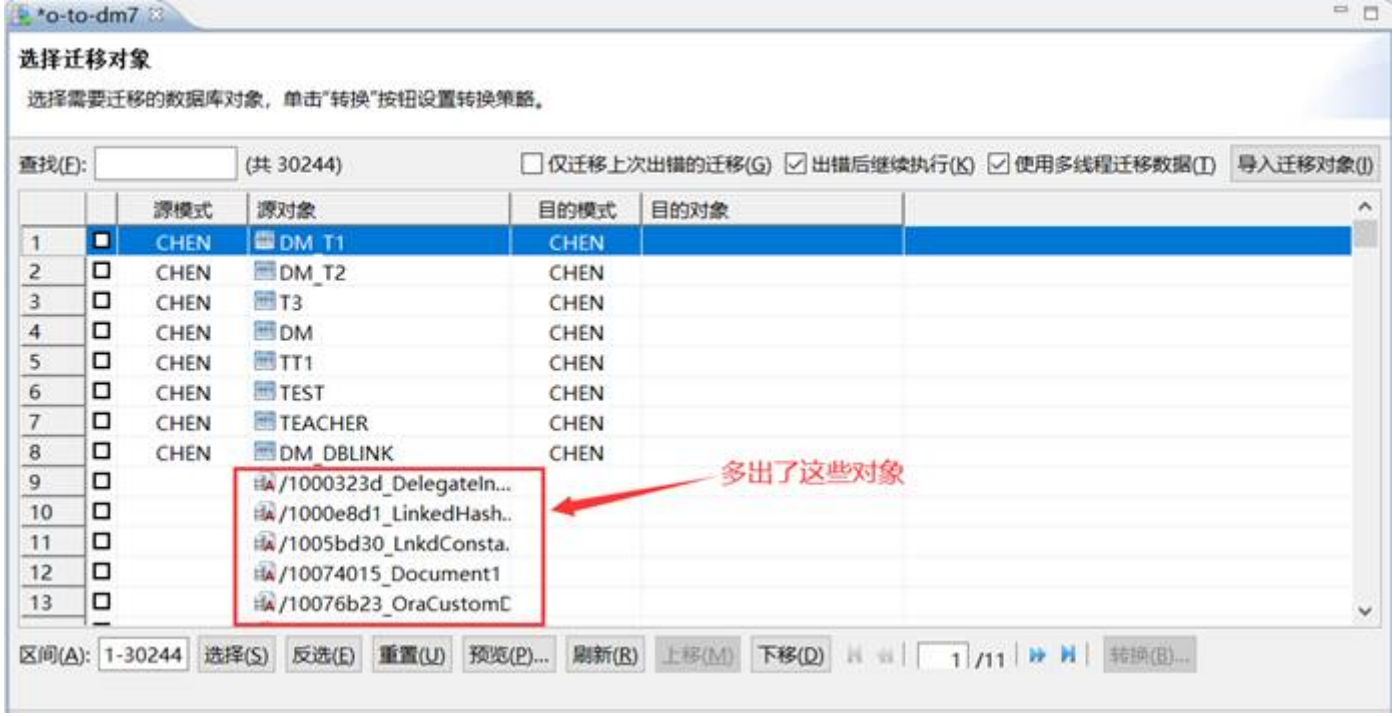
```
ALTER SYSTEM SET 'COMPATIBLE_MODE'=2 SPFILE;
```

该参数为静态参数，需要重启数据库后生效。

只选择迁移表，多出其他对象

如下图所示：



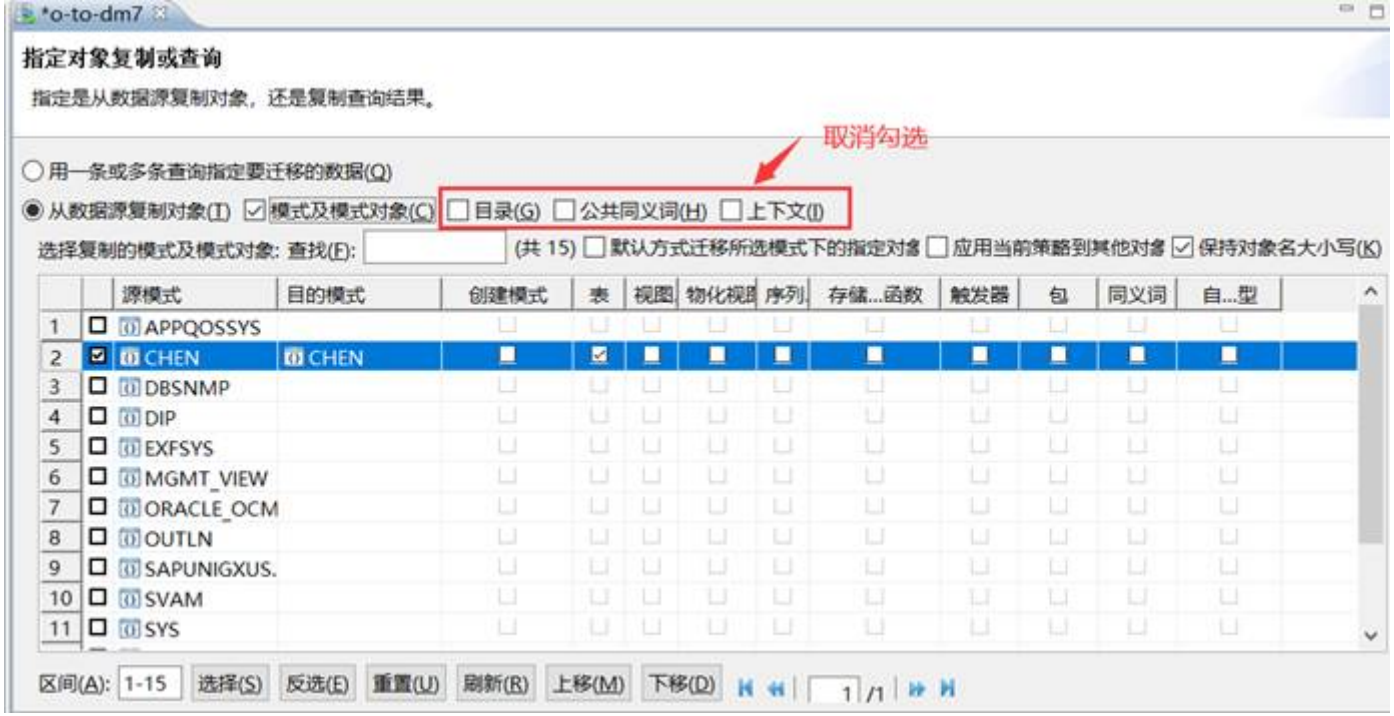


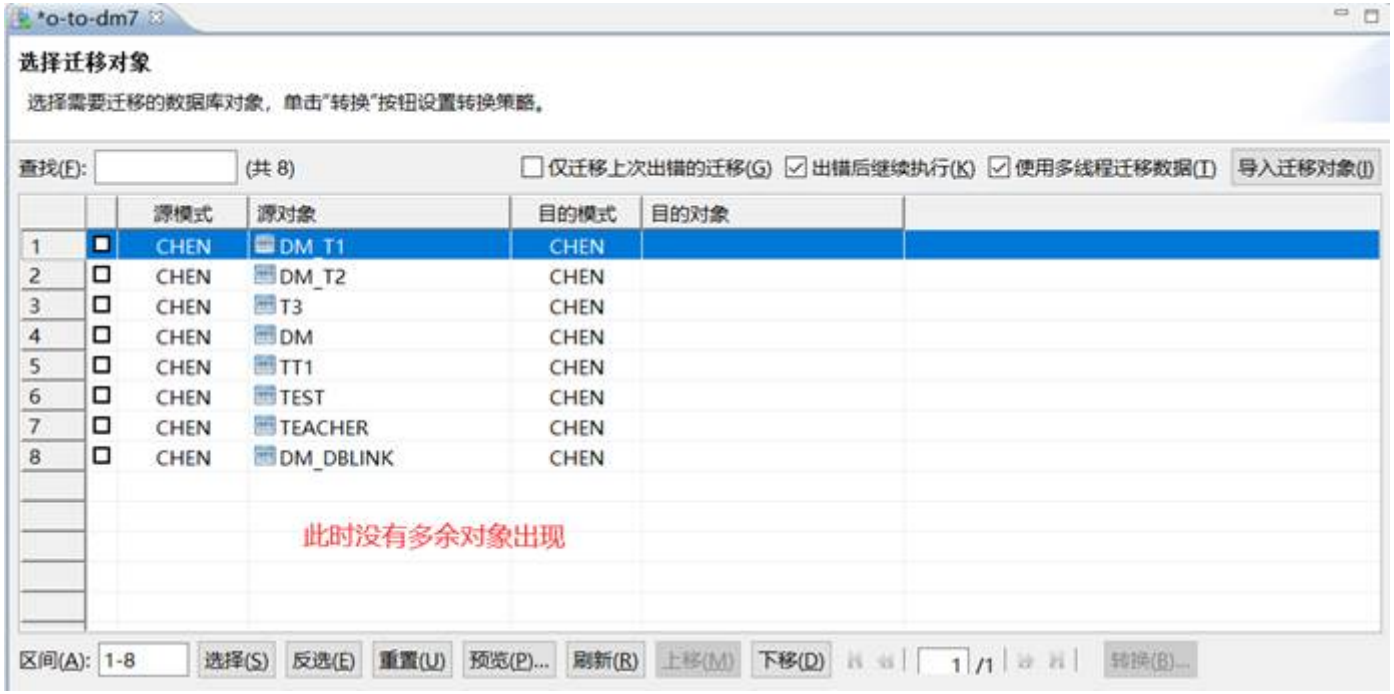
【问题原因】：

勾选了目录、公共同义词和上下文。

【解决方法】：

返回上一步，取消勾选则正常。





varchar2 (4000) 迁移后变成 (3900)

DM 数据库根据页大小，varchar 字段最大长度有限制。具体限制如下：

数据库页面大小	实际最大长度
4K	1900
8K	3900
16K	8000
32K	8188

DTS 内部有字段类型映射，同时也会根据差异情况判断，在 8 kB 页大小情况下，将 Oracle 的 varchar2 (4000) 迁移成 varchar2 (3900)。

Oracle 迁移到 DM 有关数据类型/语法的注意事项

需要注意 Oracle 的 date ， datetime 和 timestamp 都转换成 DM 的 timestamp 。

由于 Oracledate 既可以只存日期，也可以存放日期时间，DTS 迁移表结构过程中无法识别具体字段中内容再确定目的端数据类型，所以统一迁移成了 timestamp，迁移后的数据时间信息为 00:00:00.000，不影响时间字段比较和计算，如果部分字段的确只需要 date 类型则需要在 DM 中将对应字段调整为 date 类型。

DM 数据库和 Oracle 的语法大致相同，大部分都不需要修改，可以把 Oracle 中的语法到 DM 中执行，如果有报错的部分，需要查看系统管理员手册和程序员手册来查看相应的需要修改的语法部分。

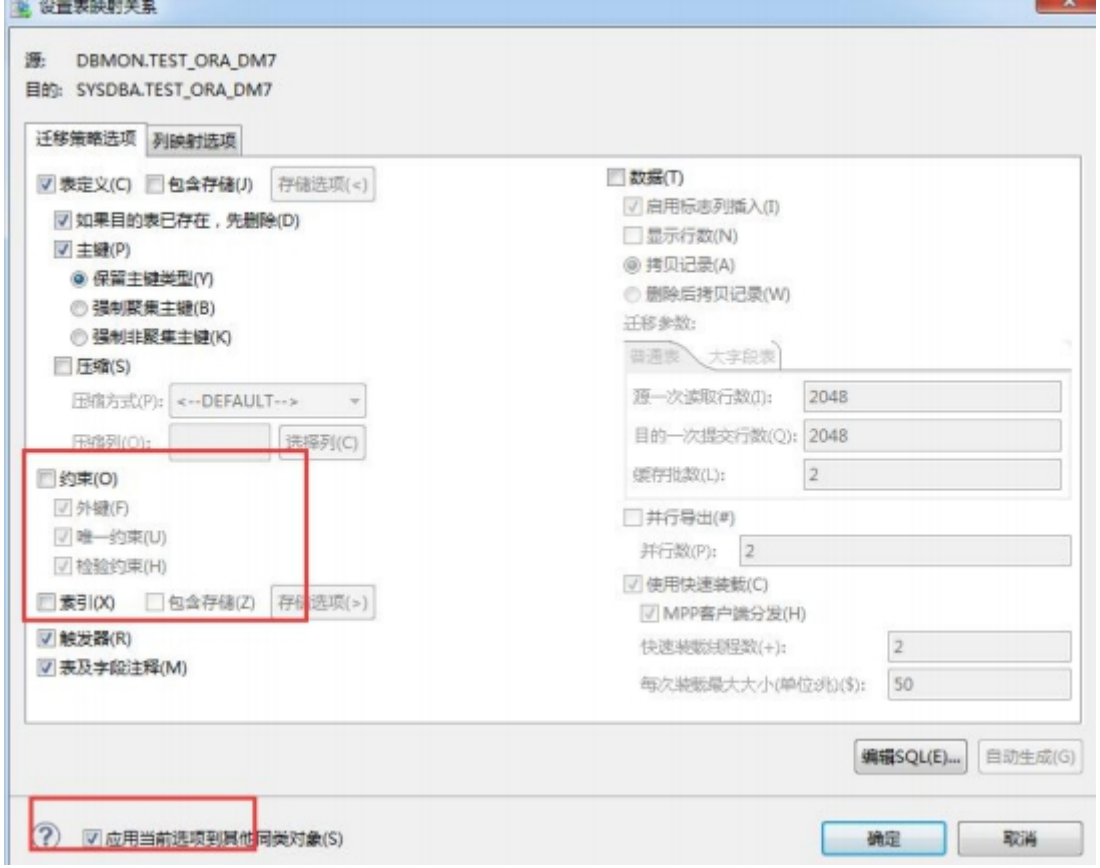
违反引用约束

这种问题主要是由外键约束造成的，父表的数据没有迁移，先迁移了子表的数据，错误如下图所示：

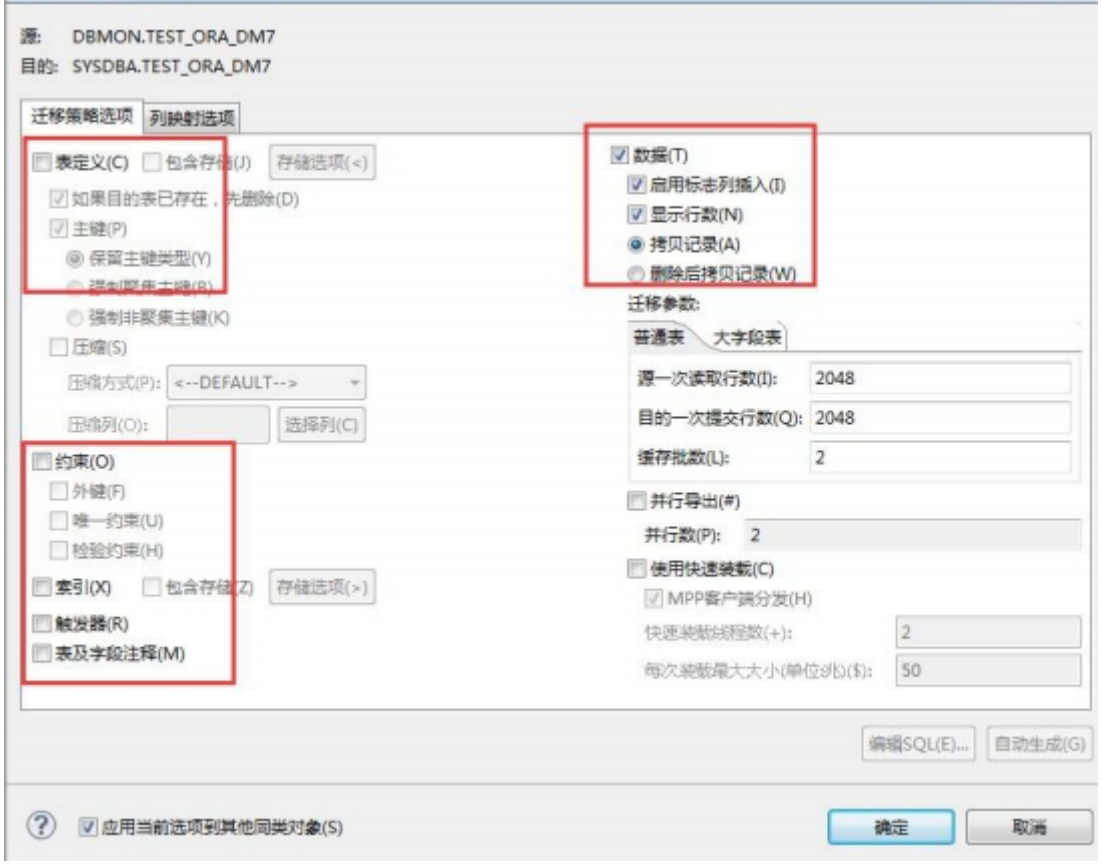
【解决方法】：

迁移的时候先不迁移外键等约束，在选择好要迁移的表时，点击【转换】，按照以下步骤进行迁移。

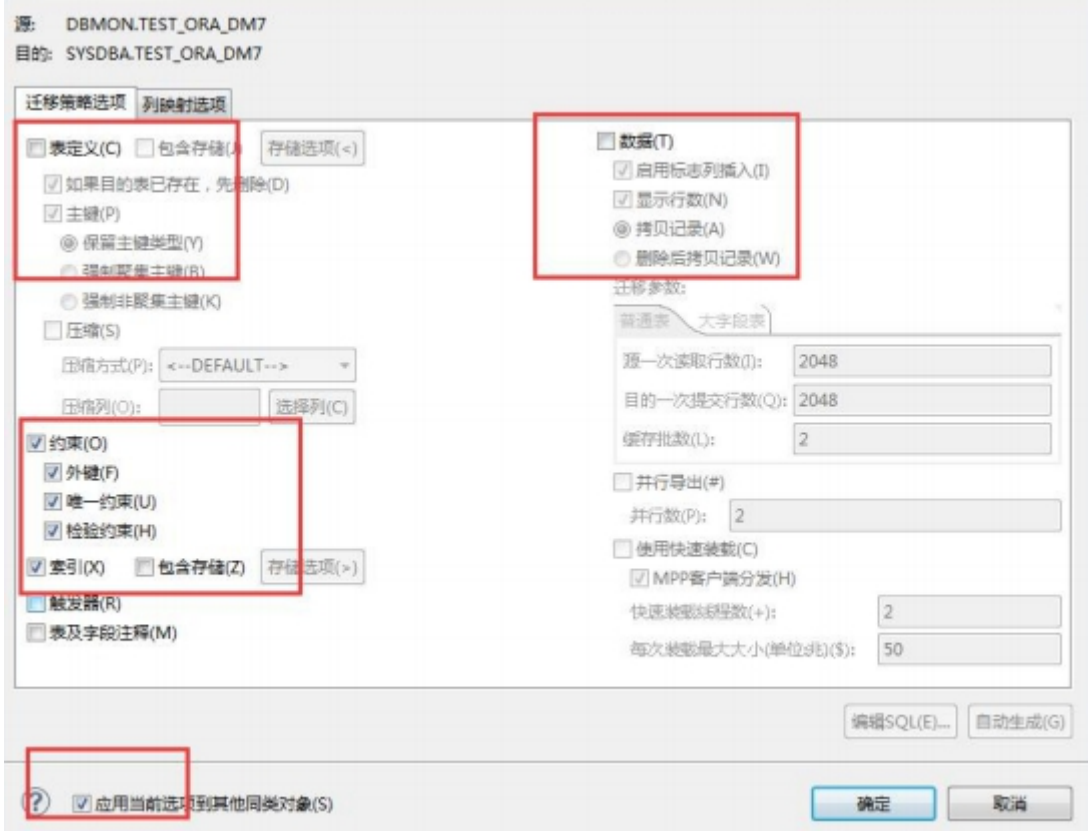
1. 第一次只选择表定义，不选择约束等，如下图所示：



2. 第一次迁移完成后（确保没有错误），第二次只选择数据，如下图所示：



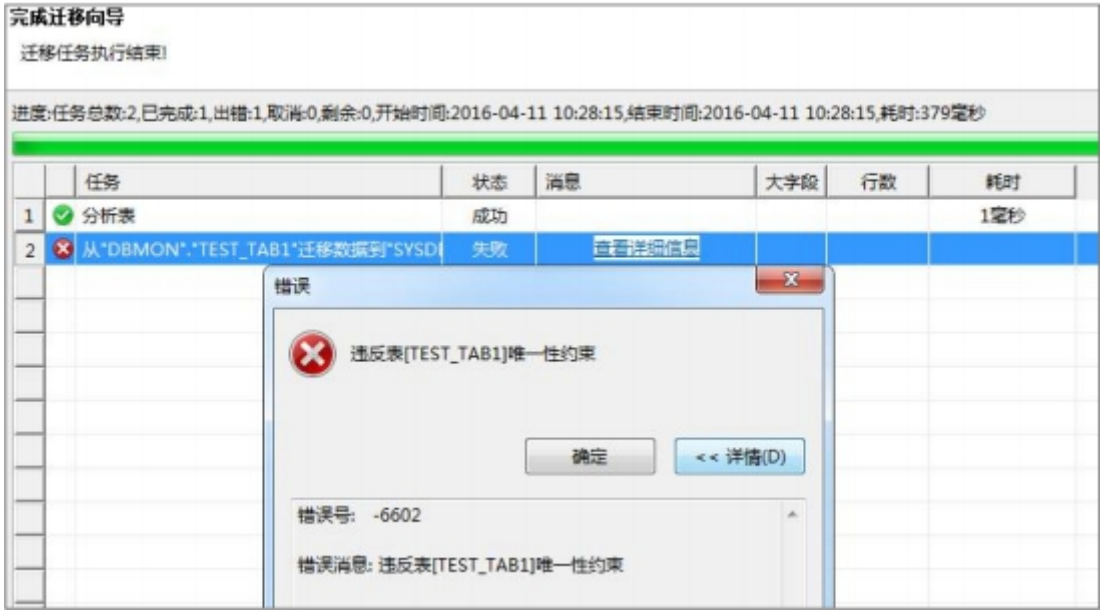
3. 第三步选择约束、索引等，如下图所示：



按照上面三步迁移，基本上都可以顺利迁移成功。

违反唯一性约束

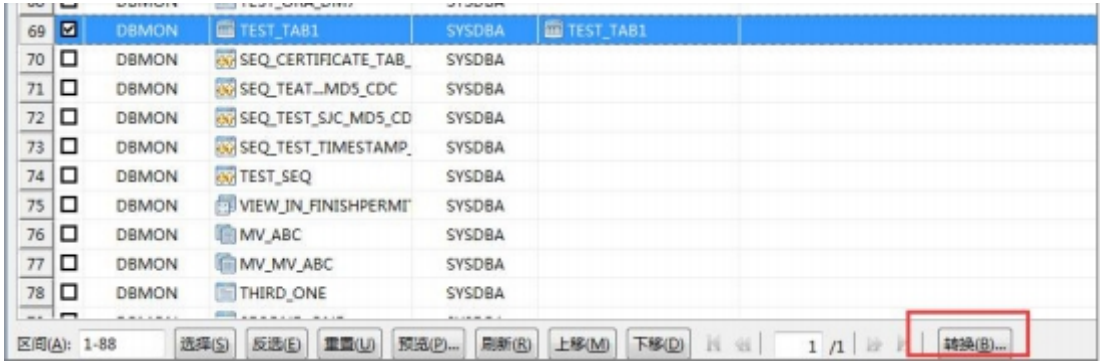
这种情况是因为表中设置了唯一性约束或者主键约束，但是数据中有重复记录。有可能是原始库的约束被禁用了，或者数据重复迁移造成的。



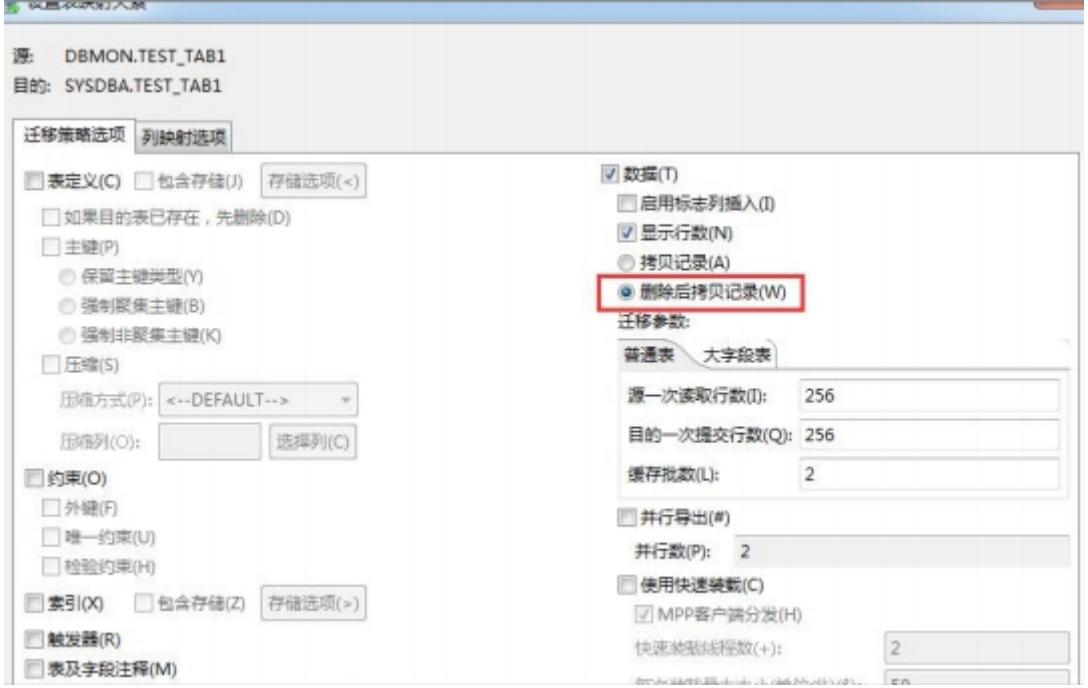
【解决方法】：

在确定源数据没有问题的情况下，迁移的时候选择删除后再拷贝，如下图所示：

在迁移界面中，选中要迁移的表，然后点击【转换】。



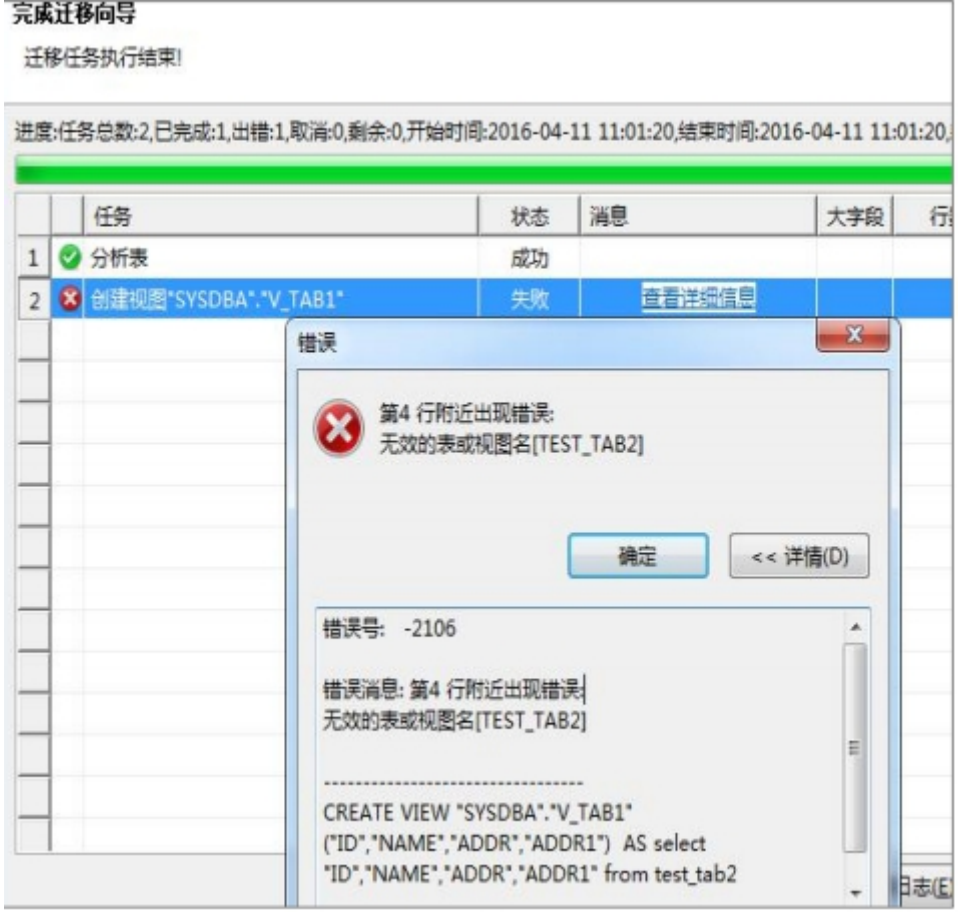
在弹出的窗口中选择删除后拷贝，如下图所示：



按照上述操作即可迁移成功。

迁移视图提示：无效的用户对象

这个问题一般是因为在迁移视图之前，没有将视图依赖的表迁移过去，如下图所示：



【解决方法】：

严格按照迁移的顺序，先迁移表，然后再迁移视图、存储过程、函数的顺序迁移即可。

连接尚未建立或已经关闭

【问题描述】：

从 Oracle 迁移数据到 DM 数据库，数据迁移时提示连接尚未建立或已经关闭。

【解决方法】：

- 原因一：这个问题有可能是因为 Oracle 库中存在非法的数据，例如： -5486-12-31 00:00:00 这样的非法的时间，在批量绑定插入的时候 JDBC 未作校验，服务器端检测到就会把这个连接剔

除，就会报这个错误。新版的 JDBC 驱动（2019 年 7 月以后）已经对此类问题进行了处理，增加了校验，碰到非法的数据会直接报错。碰到这种问题建议使用最新的 JDBC 驱动，替换掉迁移工具使用的 JDBC 驱动即可。

- 原因二：迁移连接用户设置了会话时间限制，放开会话连接限制即可，数据库 参数 max_sessions 设置。
- 原因三：迁移过程中数据库连接断开了重新迁移。

报错：精度必须大于标度

【问题描述】：

从 Oracle 迁移到 DM 的时候有一个报错：精度必须大于标度。

【解决方法】：

Oracle 中 number (m,n) 允许 n>m，但是在 DM 中是不允许的，DM 中 m>=n，m 表示精度，n 表示标度。精度是一个无符号整数，定义了总的数字数；标度定义了小数点右边的数字位数。

碰到这种问题一方面要思考一下 Oracle 里面列定义是否弄错了，如果是特意这样设计的，要搞清楚这个列到底需要存放什么样的数据，单独迁移这张表，对 DM 数据类型进行修改。

Oracle 中 raw 类型在 DM 中可以用哪种类型

Oracle 中 raw 类型在 DM 中可以使用 varbinary 代替。

如果您第一次使用 DM 数据库，可以参考从 Oracle 迁移到 DM。

Oracle 的 date 类型是对应 DM 哪种类型

【问题描述】：

DM 的 DATE 类型不能存放时分秒吗？Oracle 的 date 日期类型是对应 DM 的 datetime 类型吗？

【解决方法】：

如果设置兼容参数 COMPATIBLE_MODE = 2，date 类型默认会建成 timestamp 类型。

> **说明**
>
> Oracle 的 DATE 类型中包括年、月、日、时、分、秒，DM 中 `DATE` 类型只包括年、月、日，而时、分、秒在 `TIME` 类型中。DM 中包含年、月、日、时、分、秒的数据类型是 `TIMESTAMP` 类型。从 Oracle 迁移到 DM 时要注意 DATE 类型引起的日期比较问题，可以**将 DATE 换成 TIMESTAMP 类型**。

Oracle 的 dmp 文件导入 DM 报错：终止导入请使用更高级别的工具

Oracle 数据库的 dmp 文件不能导入到达梦数据库，每种数据库导出的 dmp 文件，都具备自身数据库特有的加密格式，其它架构的数据库肯定是无法识别的。

Oracle 数据库的数据信息可以通过达梦数据库自带的数据库迁移工具 DTS 迁移到 DM 数据库。

迁移过程中出现的错误：JAVA HEAP SPACE

dts 所在机器内存不足导致/本机内存不足。

若是 dts 机器内存不足，找到客户端修改 dts.ini 配置文件，加个内存参数 -Xmx=2048，重新迁移即可。

迁移过程中出现的错误：违反协议

一般是 Oracle 的 jdbc 驱动问题导致，尝试更换驱动包。

报错：序列最大值超出达梦范围

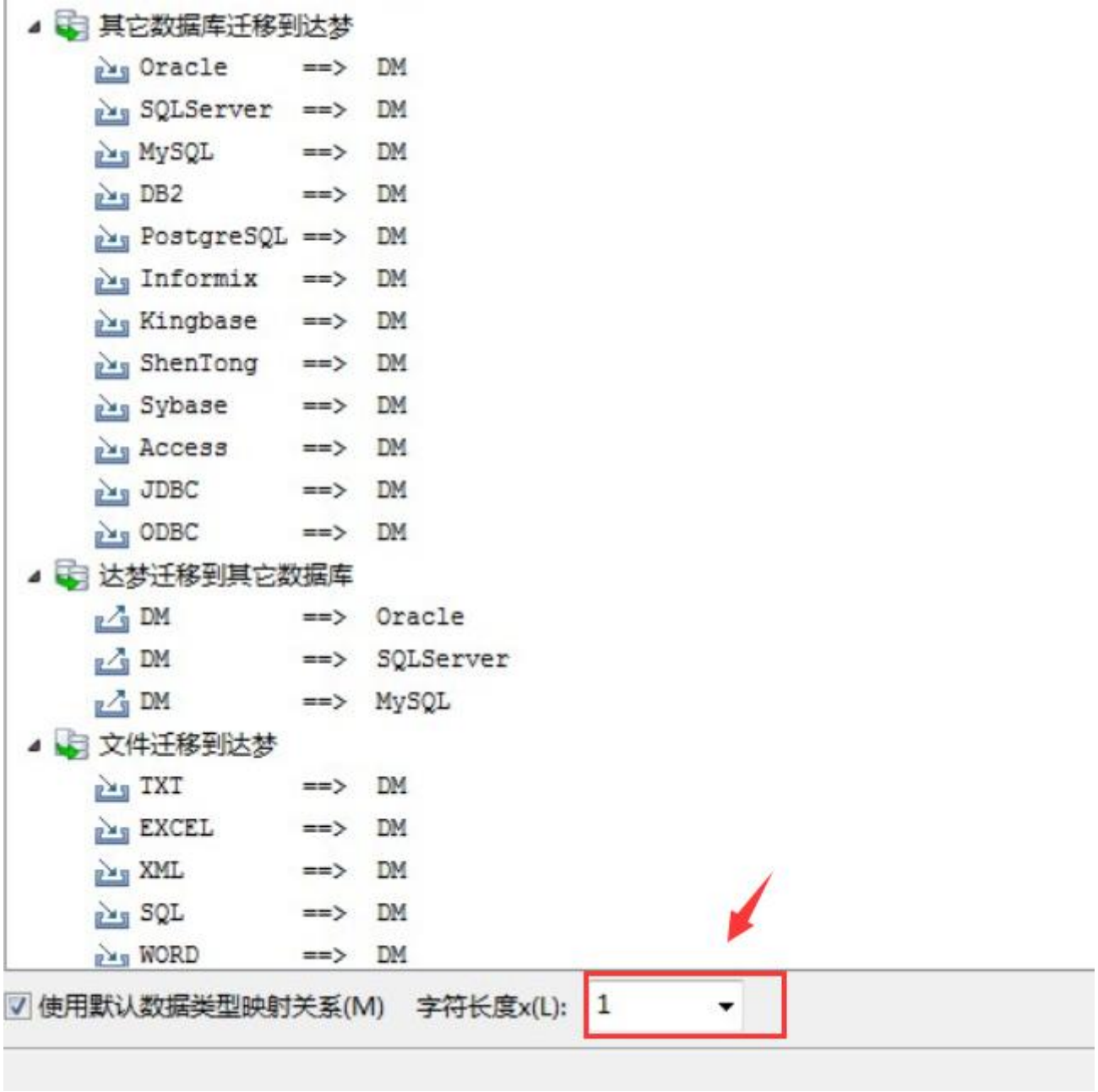
分析一：< 最大值 > 指定序列能生成的最大值，如果忽略 MAXVALUE 子句，则降序序列的最大值缺省为-1，升序序列的最大值为 9223372036854775807(0x7FFFFFFFFFFFFFFF) 。

非循环序列在到达最大值之后，将不能继续生成序列数；但是 Oracle 中最大值 28 个 9，迁移到 DM7 时报序列最大值超出达梦范围的错误（最新的 dts 版本已经将超过 DM7 最大值的序列的最大值直接转换成 9223372036854775807）

分析二：对于这类报错，需要分析源库序列用途，目的库范围是否能够满足使用情境。如可以满足，则按照目的 DM 的可定义范围设置，如存在风险，则考虑在应用层面修改或者采取其他措施规避风险。

Oracle 迁移到 DM 报错：精度超出范围

- Oracle 迁移到 DM 数据库，数据类型的映射关系，可以参考 DTS 迁移工具的帮助菜单-帮助主题--数据类型映射--默认数据类型映射关系。
- 可能是由于 Oracle 和 DM 数据库字符集不一样的原因：比如 gbk 和 utf-8 中文占用的字节数不一样。 在迁移的时候，可以将字符长度映射调正大。



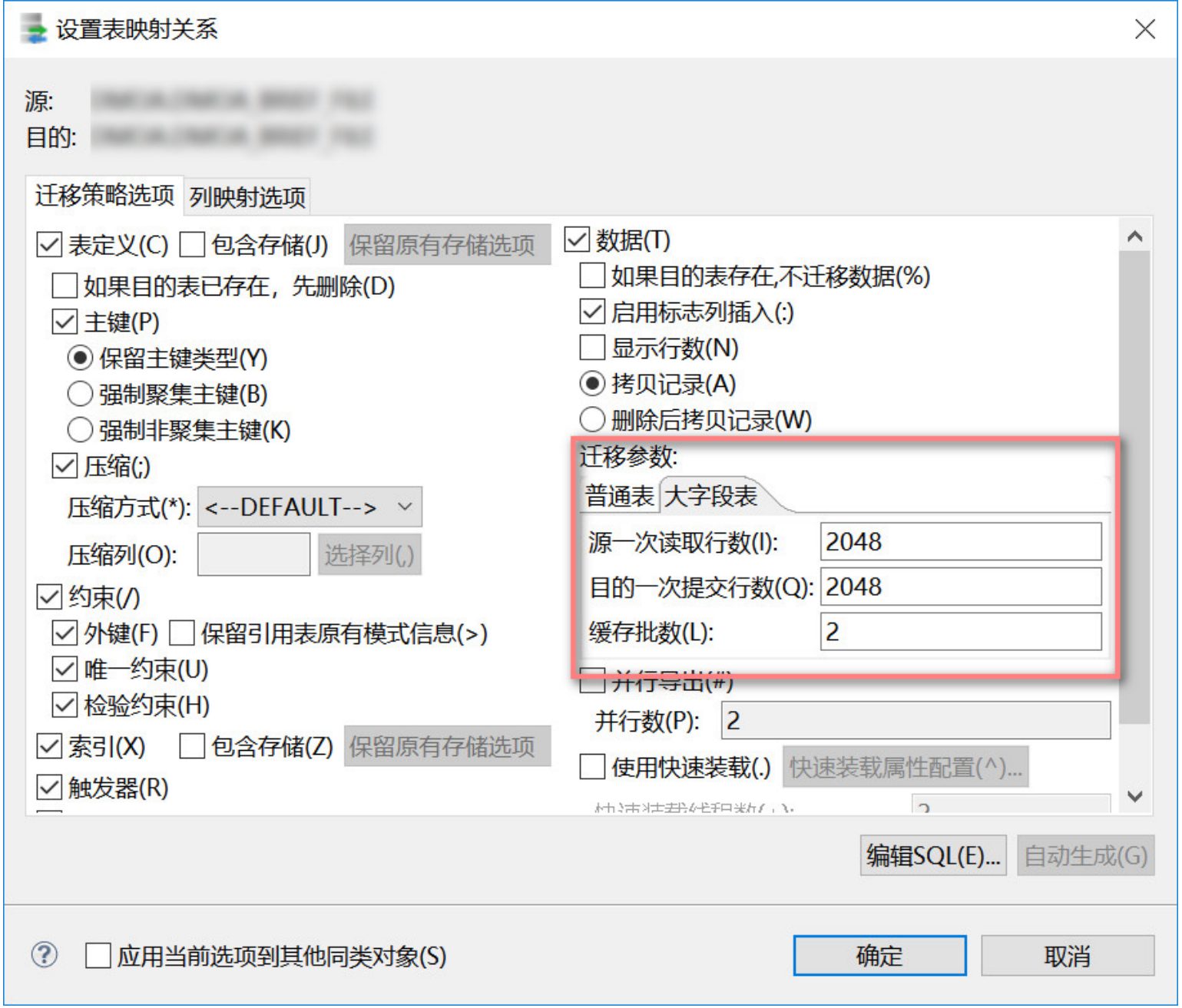
- 如果碰到中文对应到 DM 都需要三倍长度，可以检查初始化数据库的时候，达梦数据库选择字符串是否以字符为单位。（length_in_char=1 试试）

Oracle 迁移过程中出现的错误:试图在 blob 或者 clob 列上排序或比较

达梦数据库中是不允许对大字段（clob blob）类型的列进行排序操作的，部分新版本可以修改配置 ini 参数 ENABLE_BLOB_CMP_FLAG=1。

包含 blob 类型的数据表都迁移不动

需要调小 dts 里面大字段表每次读取跟提交行数。



DTS 使用时候选择删除后再拷贝记录，是否会删除源库的数据

不会，删除的是目的端的数据。

迁移的时候报错：ORA-00942

缺少查询字典权限导致。可以授予用户相应权限，语句如下：

```
grant select_catalog_role to user_name;
```

DTS 迁移工具未响应/卡住

1. 可能与客户端机器性能有关
2. 建议一次少选一些表；在进行重新迁移时，迁移时勾选清空数据库迁移，避免数据重复
3. 数据库迁移的顺序，参考达梦云适配中心的迁移手记
4. 低版本的客户端将低版本的数据库中的数据迁移到高版本中，比如：使用 DM7 迁移工具，迁移 DM7 至 DM8，迁移卡住。这时请更换成 DM8 的 DTS 工具进行迁移。

迁移报错，提示“-3236 此列列表已索引”

出现这个提示，证明列表已经有索引了，可以到目标数据库查一下，有的话就忽略这个报错。

当前对象被占用

【问题描述】：

oracle 迁移 dm 之后，想把迁移的模式删了，重新迁移一份，删除不了，一直提示错误号: -6509 错误消息: 第 1 行附近出现错误: 当前对象被占用。

【问题解答】：

查看是否开了多个窗口的管理工具，把无用的界面关闭了。是否有未结束的应用连接在操作。

删除模式时加上级联删除选项，例如：

```
DROP SCHEMA <模式名> CASCADE;
```

迁移过程中出现的错误：被引用列未添加索引

一般出现在添加外键，建议先添加唯一约束然后添加外键约束。

迁移过程中出现的错误：对象不再存在

一般是导入的表是临时表，重新导入即可。

修改兼容性参数 COMPATIBLE_MODE=2 后，SQL 走索引查询 is null 无法查询出空字符串数据

【问题说明】：

在修改兼容性参数 COMPATIBLE_MODE=2 后，SQL 查询 where 条件 X is null，执行计划走 X 列单列索引或者以 X 列为前导列的组合索引无法查询出空字符串数据，且重建索引也没有作用，禁用或者删除对应索引可以正常查询出包含空字符串的数据。

原因是由于在修改兼容性参数 COMPATIBLE_MODE=2 之前，数据表中已有空字符串数据，修改 COMPATIBLE_MODE=2 之后新插入的空字符串数据自动转换为 NULL 存入表中。

COMPATIBLE_MODE 参数相当于初始化参数，需要在创建数据库实例之后或者创建业务用户表之前设置好。

【解决方法】：

新建实例在创建业务用户表之前修改兼容性参数 COMPATIBLE_MODE=2，然后使用迁移工具重新迁移数据。或者手动 update 空字符串数据为 null。

DTS 工具从 oracle 迁移到 dm 库，提示迁移完成，但实际数据量与源库不匹配，缺失数据

明确迁移源库 oracle 数据库的具体版本信息，一定要在 DTS 迁移工具中指定源 oracle 数据库的驱动为自动义的且是与 oracle 版本适配的 jdk 版本，因为默认 dts 工具在选择源端驱动时时默认的一个驱动版本，不一定和真实的库完全匹配。

oracle 数据库版本与 JDK 的版本选择，参考如下：

Oracle Database versionJDBC Jar files specific to the release
21.1—— ojdbc11.jar with JDK11, JDK12, JDK13, JDK14 and JDK15； ojdbc8.jar with JDK8, JDK11, JDK12, JDK1
19.x——ojdbc10.jar with JDK10, JDK11； ojdbc8.jar with JDK8, JDK9, JDK11
18.3——ojdbc8.jar with JDK8, JDK9, JDK10, JDK11
12.2 or 12cR2——ojdbc8.jar with JDK 8
12.1 or 12cR1——ojdbc7.jar with JDK 7 and JDK 8； ojdbc6.jar with JDK 6
11.2 or 11gR2——ojdbc6.jar with JDK 6, JDK 7, and JDK 8(Note: JDK7 and JDK8 are supported in 11.2.0.3 and

oracle 迁移到 DM，表结构一致，迁移数据报错：数据溢出

【问题说明】：
有两种可能的情况：

1. Oracle 的 int 类型自动转换成 number(38)；
2. Oracle 的 number 类型存放精度超过 38 位；

情况一：
Oracle 创建 int 类型，默认自动转换成 number(38)

```
SQL> create table test01(a int);

Table created.

SQL> desc test01;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 A                                                  NUMBER(38)
```

实际上在做插入的时候，精度超过 38 位，插入也是可以成功的
以下显示插入 40 位数字是成功的：

```
SQL> insert into test01 values (12345678901234567890123456789012345678901234567890);

1 row created.

SQL> commit;

Commit complete.
```

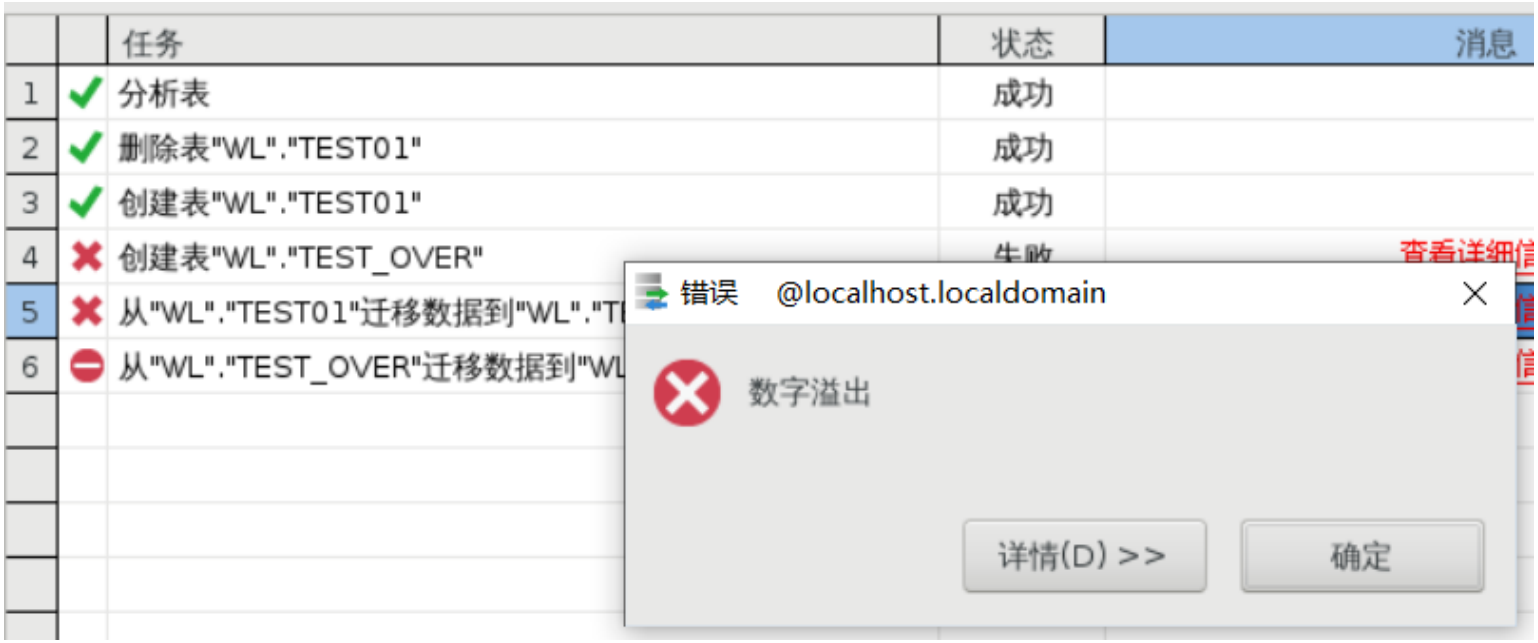
迁移到 DM 时，也会创建相同的表结构：

```
SQL> select dbms_metadata.get_ddl('TABLE','TEST01','WL');

行号     dbms_metadata.get_ddl('TABLE','TEST01','WL')
---------- ----------------------------------------------------
1          CREATE TABLE "WL"."TEST01"
(
"A" NUMBER(38,0)) STORAGE(ON "MAIN", CLUSTERBTR) ;

已用时间: 2.763(毫秒). 执行号:19984502.
```

但 DM 的 number(38)类型，严格限制最大仅能存 38 位，超过这个精度在迁移的时候就会报：数字溢出



情况二：

Oracle 创建 number 类型，实际插入超过 38 位：

```
SQL> desc test05
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 A                                                  NUMBER

SQL> insert into test05 values (12345678901234567890123456789012345678);

1 row created.

SQL> commit;

Commit complete.
```

迁移到 DM，也是 number 类型，但超过 38 位，DM 会报数字溢出

【解决方法】：

1. 确认数据本身有没有错误，在 Oracle 中存储这种超长的数值，理论上也是不支持；

2. DM 超过 38 位的 number 类型，建议改成 varchar 类型；

数据库迁移，在执行任务迁移表的时候长时间卡住不动

【原因及解答】：

1. 表的数量较多，查询源端元数据时间较长；
2. 快速装载的问题，建议在表--》转换中，把大字段的每次读取的行数改成 1，关掉快速装载；
3. 目的端表被锁住，此时再用 DTS 向目的端的表做插入会被阻塞造成长时间卡顿，将目的端的表 drop 掉或者解锁重新迁移；

迁移后创建唯一约束报错，但数据无重复

【问题说明】：
这种情况可能是由于字段某些值的末尾存在空格。
若数据库需要兼容 oracle，则在初始化数据库时需要将空格填充模式设为与 oracle 一致。

【解决方法】：
方法一：利用 dminit 初始化实例时，添加参数 BLANK_PAD_MODE=1。
方法二：使用“DM 数据库配置助手”图形化界面初始化库时，勾选“空格填充模式”。

迁移 forall 脚本报错：语法分析出错

【问题描述】：

迁移 forall 脚本报错。Oracle 中的 forall 脚本如下图所示：

```
BEGIN
  v_sql := '
    INSERT INTO PARTS VALUES (:1,:2)';
  FOR I IN 1 .. 50 LOOP
    PNUMS(I) := I;
    PNAMES(I) := 'Part No.' || TO_CHAR(I);
  END LOOP;

  T1 := DBMS_UTILITY.GET_TIME;

  FOR I IN 1 .. 50 LOOP
    INSERT INTO PARTS VALUES (PNUMS(I), PNAMES(I));
  END LOOP;
  T2 := DBMS_UTILITY.GET_TIME;

  FORALL I IN 1 .. 50 execute immediate v_sql USING PNUMS(I),PNAMES(I);

  T3 := DBMS_UTILITY.GET_TIME;
```

报错信息如下图所示：

【问题解决】：

由于 DM 目前不支持 execute immediate 动态语句，因此直接改为非动态 sql 即可，如下图所示：

```
v_sql VARCHAR2(2000);
BEGIN
  v_sql := '
    INSERT INTO PARTS VALUES (:1,:2)';
  FOR I IN 1 .. 50 LOOP
    PNUMS(I) := I;
    PNAMES(I) := 'Part No.' || TO_CHAR(I);
  END LOOP;

  T1 := DBMS_UTILITY.GET_TIME;

  FOR I IN 1 .. 50 LOOP
    INSERT INTO PARTS VALUES (PNUMS(I), PNAMES(I));
  END LOOP;
  T2 := DBMS_UTILITY.GET_TIME;
  --forall优化
  FORALL I IN 1 .. 50
    --execute immediate v_sql USING PNUMS(I),PNAMES(I);
    INSERT INTO PARTS VALUES (PNUMS(I),PNAMES(I));

```

在迁移 Oracle 的存储过程、包时报错：试图更新只读视图

【解决办法】：

对于复杂的视图（包含 UNION、多条连接等），Oracle、达梦默认都是不允许更新的。但是若存储过程脚本中包含更新视图语句，在 Oracle 中编译能通过，而达梦由于对语法校验更加严谨，编译会不通过。此问题解决办法如下：
步骤 1：首要确认 oracle 是否真的可以更新复杂视图，一般情况是不允许的。如果 oracle 上不允许，则需要在 oracle 上修改，再迁移到达梦。
步骤 2：如果 oracle 允许修改复杂视图（打开了隐含参数_complex_view_merging），达梦则需要把修改视图的语句拆分为多个语句，单独修改视图的基表。
例如：

```
---创建表 T1 和 T2
create table T1(ID INTEGER,NAME VARCHAR2(200),FU NUMBER);
create table T2(ID INTEGER,NAME VARCHAR2(200),FU NUMBER);

---创建含 union 的视图
```

```
CREATE OR REPLACE VIEW V_T1_T2 AS
SELECT "ID","NAME","FU" FROM t1
UNION ALL
SELECT "ID","NAME","FU" FROM t2;

--原sql
CREATE PROCEDURE p5 IS
BEGIN
  UPDATE v_t1_t2 SET NAME = 'aaaa' WHERE ID = 99;
END;

--达梦把视图拆分，单独修改
CREATE PROCEDURE p5 IS
BEGIN
  UPDATE t1 SET NAME = 'aaaa' WHERE ID = 99;
  UPDATE t2 SET NAME = 'aaaa' WHERE ID = 99;
END;
```

兼容 oracle sys_extract_utc 函数实现时区类型数据转换为 utc 的 timestamp 值

【问题解决】：

```
CREATE or REPLACE
      FUNCTION sys_extract_utc FOR CALCULATE
            (
            v_time in datetime(6)
            with time zone)
            RETURN timestamp
      is
      begin
            return
            case substr(v_time, 28, 1)
```

```
        when '+' then
            to_date(substr(v_time, 0, 27))-to_number(substr(v_time, 29, 2))/24-to_number(substr(v_time, 32
        when '-' then
            to_date(substr(v_time, 0, 27))+to_number(substr(v_time, 29, 2))/24+to_number(substr(v_time, 3
        end;
    end;
```

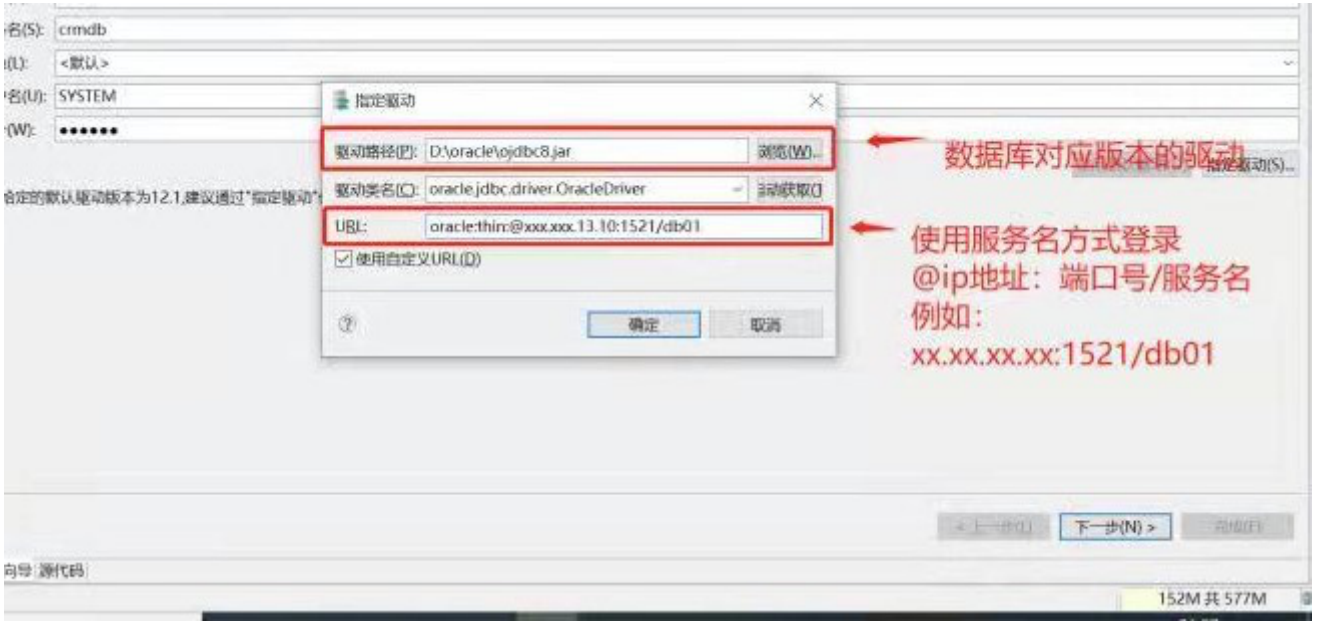
ORACLE 12C 使用默认方式连接不到需要迁移的库

【问题描述】：

ORACLE 12C 后多出一个新特性多容器，此时迁移数据时使用默认的方式迁移会导致连接不到需要迁移的库。

【问题解决】：

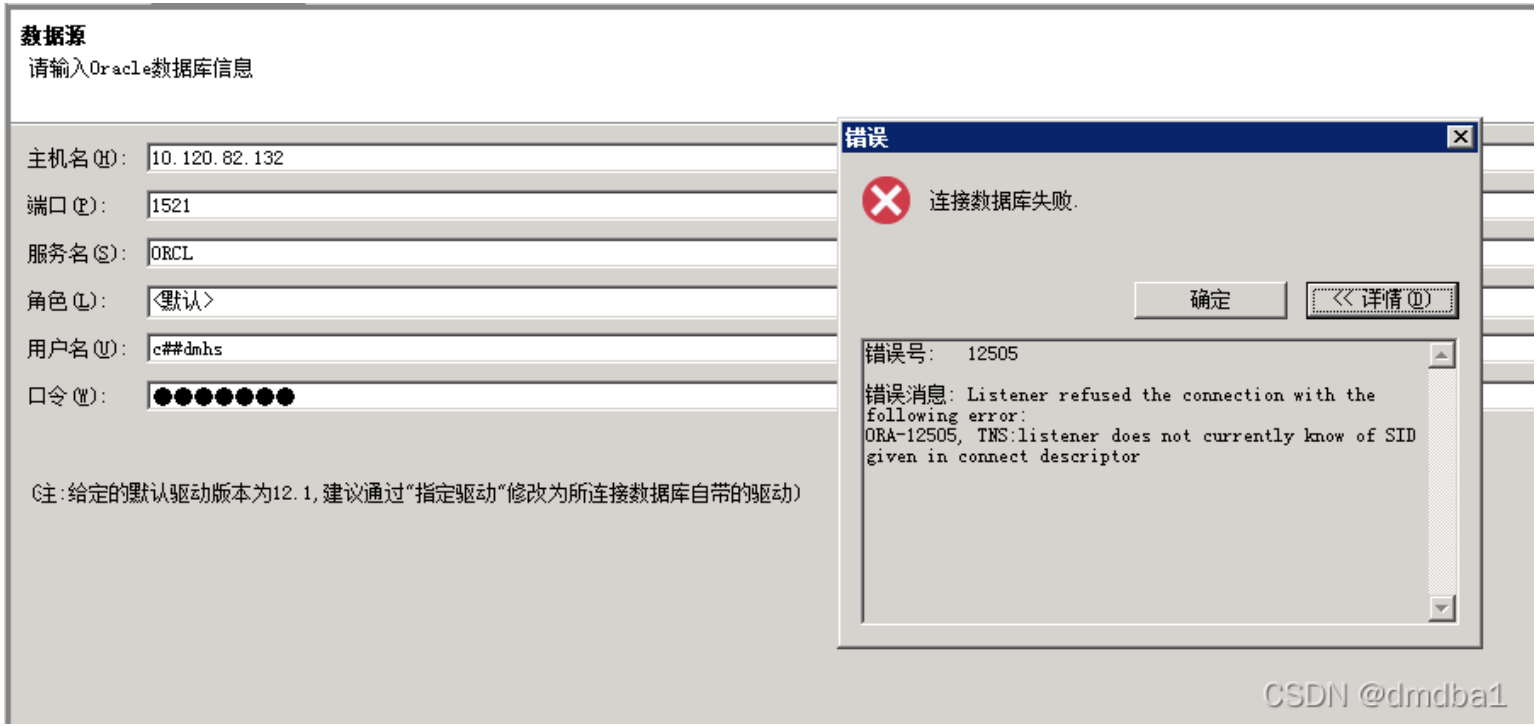
在迁移时不再使用默认方式，需要指定驱动和 URL
驱动路径：选择与数据库版本相对应的驱动
URL:使用服务名的方式登录 @ip 地址：端口号/服务名



Oracle 12c 及以上版本使用达梦迁移工具不识别 SID

【问题描述】：

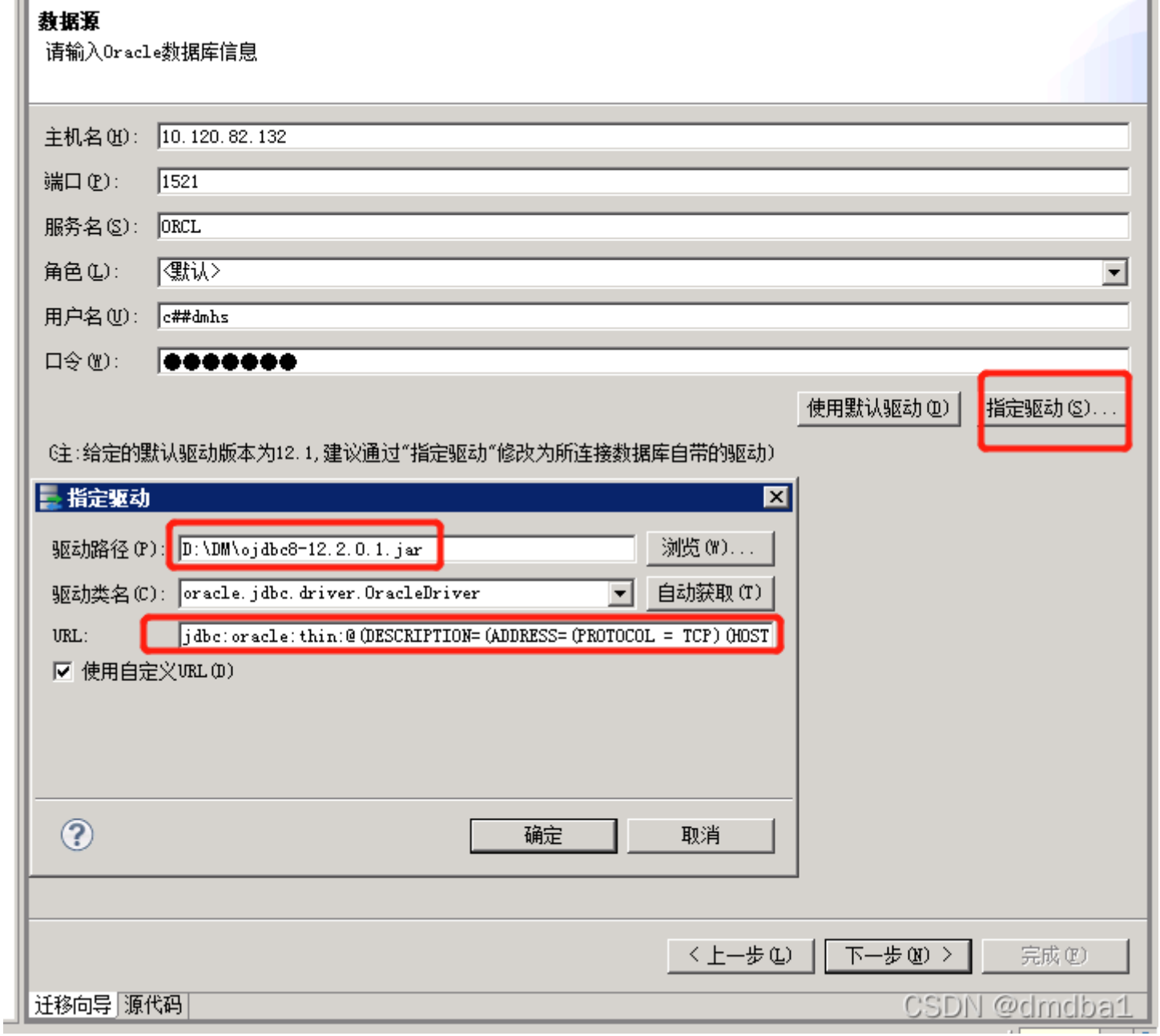
Oracle 12c 及以上版本使用达梦迁移工具不识别 SID,报错如下：
错误号: 12505
错误消息: Listener refused the connection with the following error:ORA-12505, TNS:listener does not currently know of SID given in connect descriptor



【问题解决】：

通过指定驱动进行连接：指定驱动--> 使用 Oracle12c 驱动包--> 填写 URL。
URL 内容如下：

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL = TCP)(HOST = 10.120.82.132)(PORT = 1521))(CO
```



DTS 工具迁移表结构时列长度修改不生效

【问题描述】：

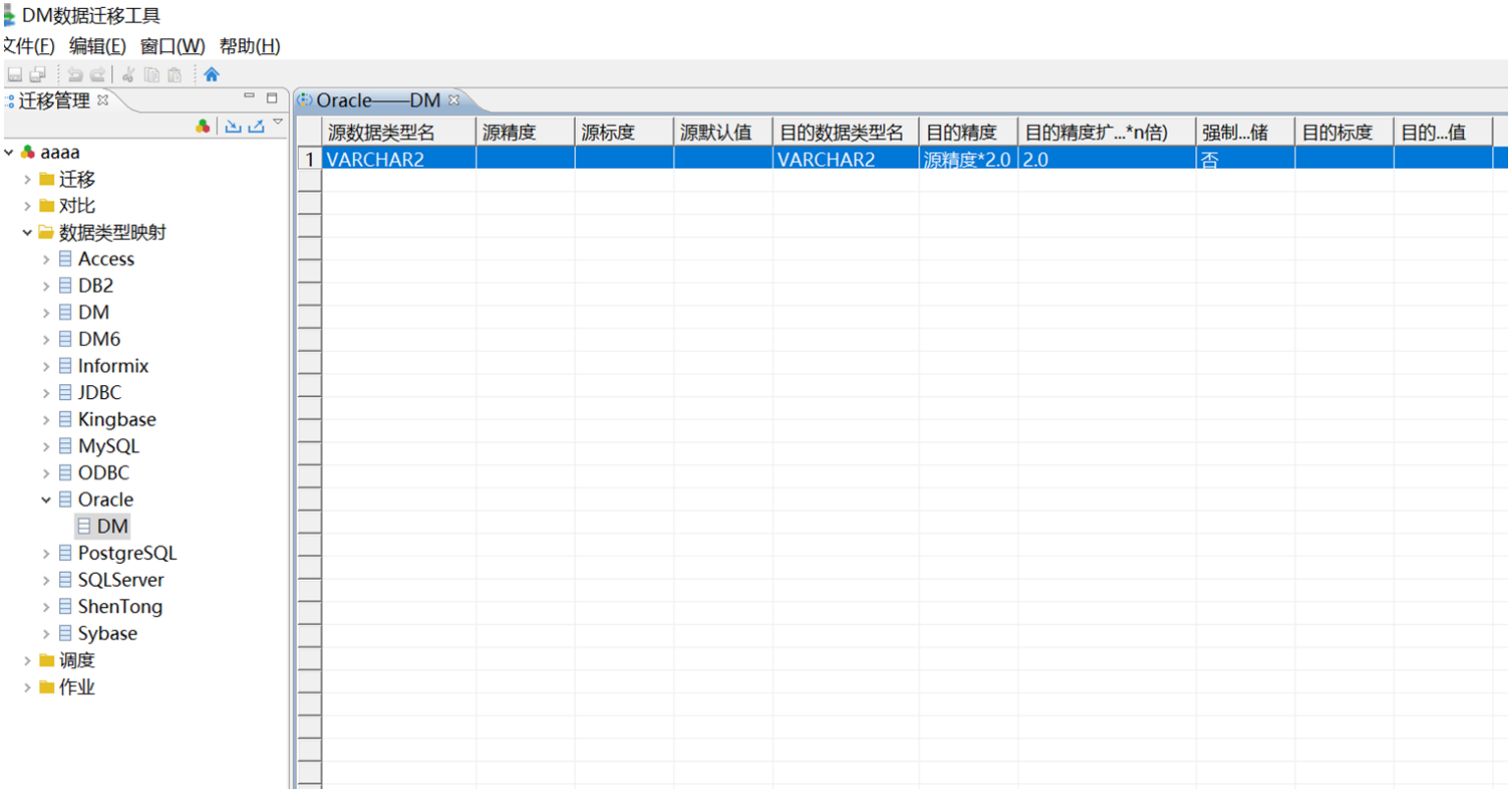
DTS 工具迁移表结构时直接改变列长度，设置列长度修改不生效。

【问题解决】：

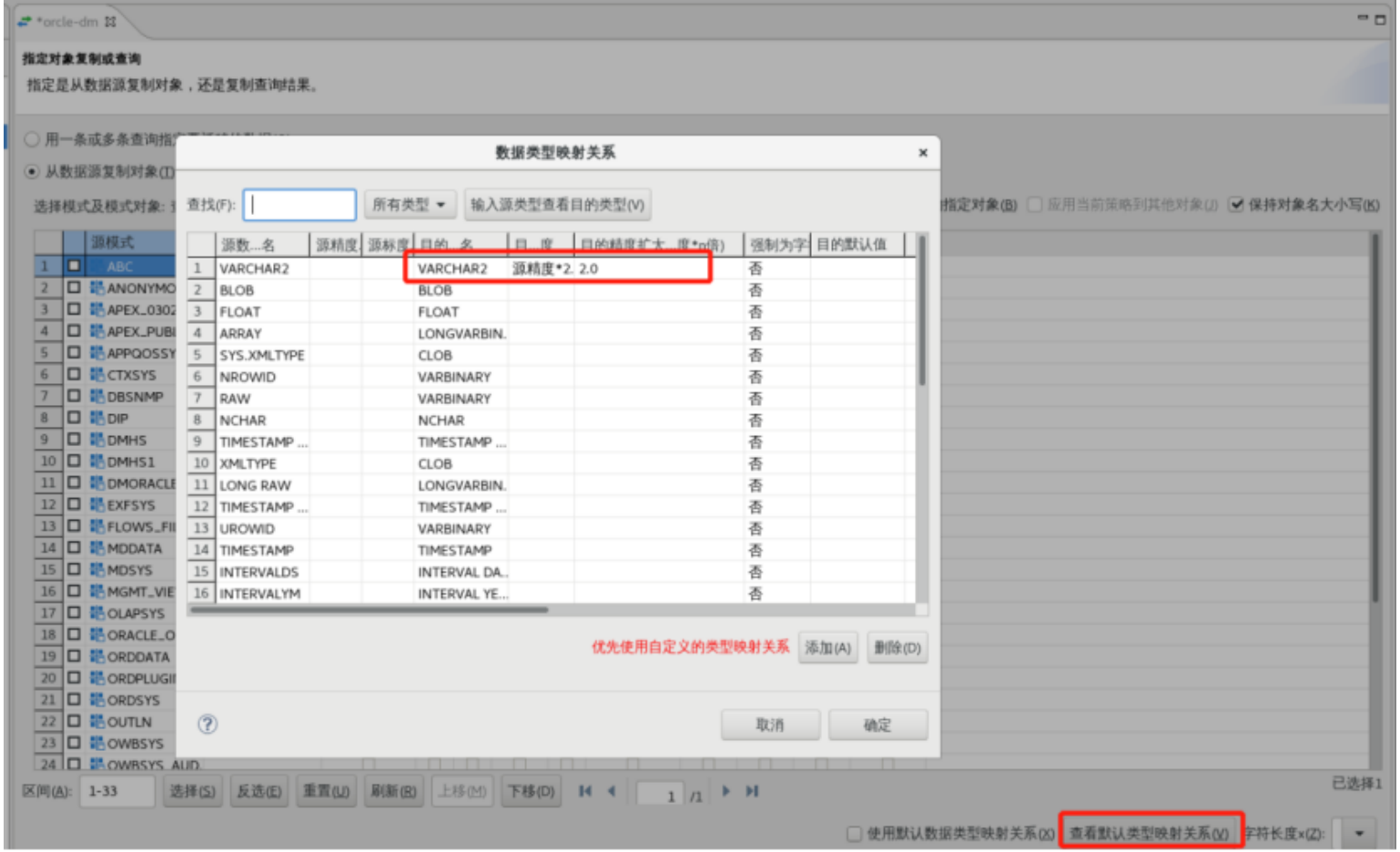
Oracle 迁移到 dm 时，在不使用自定义类型时设置扩展列长度不生效，需要在数据类型映射目录下面的 oracle 选项中进行自定义映射配置，使用自定义列映射的方法进行映射。

例如：从 Oracle 迁移到 DM 时，源端建表语句为 Create table test2(id int,name varchar2(20))；现在需要将 varchar2 类型的列长度扩展为原来的 2 倍，扩展为 varchar2(40)。具体可参考以下步骤：

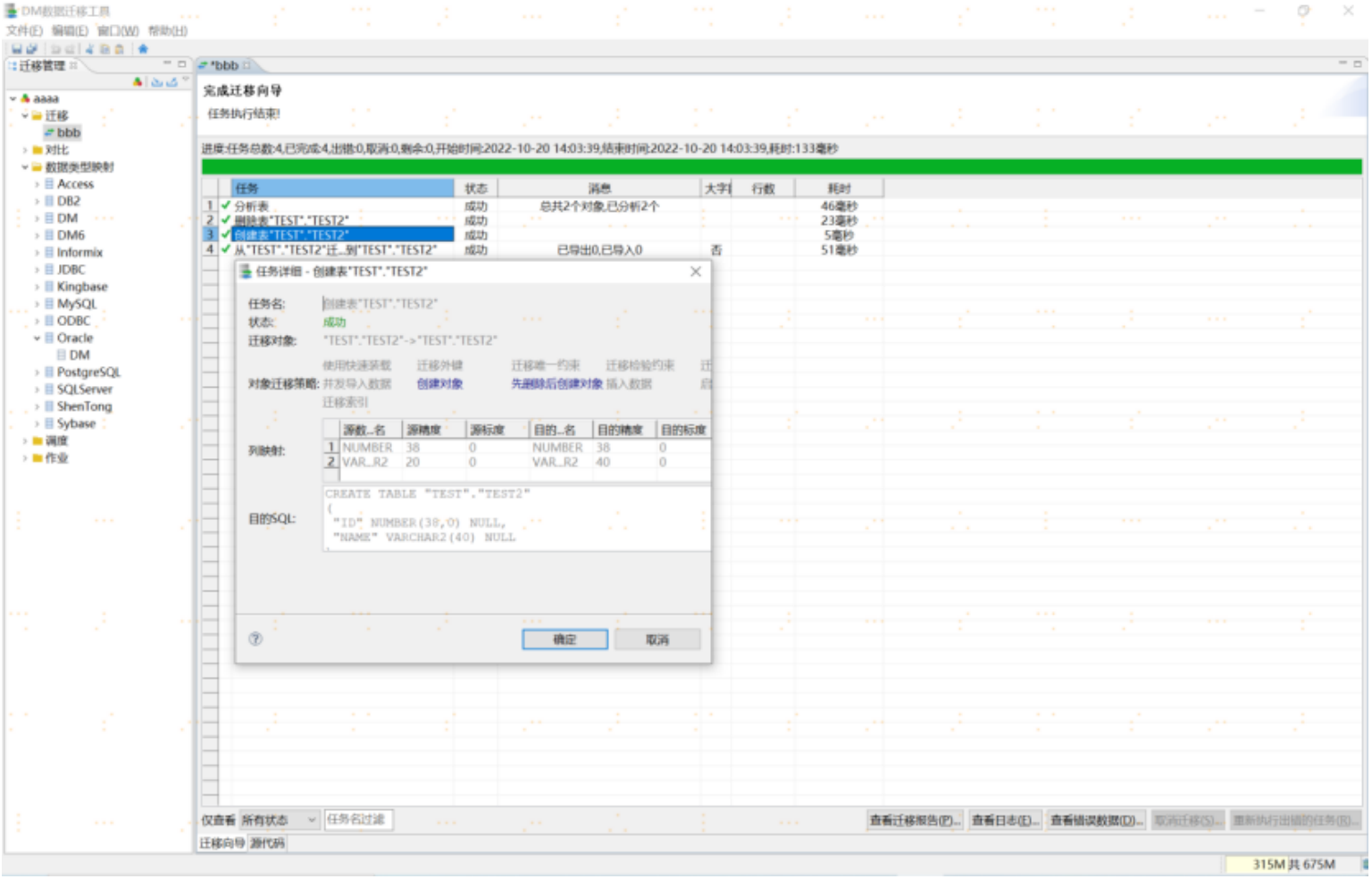
1. 使用自定义映射类型，在 DTS 中新建工程以后，首先设置数据类型映射，在数据类型映射中找到对应的源端库，这里以 oracle 为例，双击 DM 对象，然后点击数据类型名可选则要扩展的数据类型类型，输入要扩展的倍数。



2. 在设置了自定义列映射以后，新建迁移，然后正常执行 dts 迁移步骤，到对象复制页面，取消勾选“使用默认数据类型映射关系”，点开“查看默认类型映射关系”，可以直接看到映射关系出现在最上方，已生效，继续后面的迁移步骤；



3. 迁移结束后，在结束面版点击查看建表语句，可以看到目的端 varchar2 类型的列长度已经扩展为原来的 2 倍；



在迁移过程中分区表迁移速度较慢，如何提高迁移到目的端的效率?

【问题解决】

可以通过分区置换的方法提高迁移到目的端的效率，前提需要创建的普通表和分区表的表结构以及索引且索引顺序全部保持一致。以下有两种迁移方式可供参考。

1. 通过分区置换的方法，置换为普通表后，DTS 开启多个窗口并行迁移到目的端，迁移完成后，目的端通过分区置换方法，再置换为分区表；
2. 通过分区置换的方法，置换为普通表后，通过表备份还原的方法，将备份还原到目的端后，通过分区置换方法，再置换为分区表。

下面对分区置换方法进行举例说明：

1. 统计分区表的分区数量：

```
select count(*) from "TEST"."TT_上海";
select count(*) from "TEST"."TT_云南";
select count(*) from "TEST"."TT_内蒙";
```

2. 创建和分区表相同表结构的普通表以及索引：

```
CREATE TABLE "TEST"."TT_上海_1" ("area_code" VARCHAR(255),"TTsn" VARCHAR(80),"org_code" VARCHAR(64
CREATE TABLE "TEST"."TT_云南_2" ("area_code" VARCHAR(255),"TTsn" VARCHAR(80),"org_code" VARCHAR(64
CREATE TABLE "TEST"."TT_内蒙_3" ("area_code" VARCHAR(255),"TTsn" VARCHAR(80),"org_code" VARCHAR(64

CREATE  INDEX "index-TT_1" ON "TEST"."TT_上海_1"("org_code" ASC) STORAGE(ON "TEST_DATA", CLUSTERBT
CREATE  INDEX "index-TT_2" ON "TEST"."TT_云南_2"("org_code" ASC) STORAGE(ON "TEST_DATA", CLUSTERBT
CREATE  INDEX "index-TT_3" ON "TEST"."TT_内蒙_3"("org_code" ASC) STORAGE(ON "TEST_DATA", CLUSTERBT
```

3. 将分区表的各个分区转换为普通表：

```
alter table "TEST".TT exchange PARTITION  "上海"  with table "TEST"."TT_上海_1";
alter table "TEST".TT exchange PARTITION  "云南"  with table "TEST"."TT_云南_2";
alter table "TEST".TT exchange PARTITION  "内蒙"  with table "TEST"."TT_内蒙_3";
```

4. 备份表普通表：

```
BACKUP TABLE "TEST"."TT_上海_1" BACKUPSET '/data2/dmdata/dbbak/TT_1';
BACKUP TABLE "TEST"."TT_云南_2" BACKUPSET '/data2/dmdata/dbbak/TT_2';
BACKUP TABLE "TEST"."TT_内蒙_3" BACKUPSET '/data2/dmdata/dbbak/TT_3';
```

5. 将所有的表备份发送到目标机器 10.xx.xx.12 上：

```
scp -r /data2/dmdata/dbbak dmdba@10.xx.xx.12:/dev/shm/dbbak
```

6. 还原表结构：

```
RESTORE TABLE struct FROM BACKUPSET '/dev/shm/dbbak/dbbak/TT_1';
RESTORE TABLE struct FROM BACKUPSET '/dev/shm/dbbak/dbbak/TT_2';
RESTORE TABLE struct FROM BACKUPSET '/dev/shm/dbbak/dbbak/TT_3';
```

7. 还原表数据：

```
RESTORE TABLE FROM BACKUPSET '/dev/shm/dbbak/dbbak/TT_1';
RESTORE TABLE FROM BACKUPSET '/dev/shm/dbbak/dbbak/TT_2';
RESTORE TABLE FROM BACKUPSET '/dev/shm/dbbak/dbbak/TT_3';
```

8. 在 10.xx.xx.12 上新建一张分区表：

```
CREATE TABLE "TEST"."TT10"
(
"area_code" VARCHAR(255),
"TTsn" VARCHAR(80),
"org_code" VARCHAR(64),
"file_url" VARCHAR(50),
"entity" TEXT,
"created" DATETIME(6))
PARTITION BY LIST("area_code")

(
PARTITION  "内蒙"  VALUES('150000') ,
PARTITION  "云南"  VALUES('530000') ,
PARTITION  "上海"  VALUES('310000') ,
)
```

```
CREATE  INDEX "index-orgCode10" ON "TEST"."TT10"("org_code" ASC) STORAGE(ON "TEST_DATA", CLUSTERE
```

9. 将 10.xx.xx.12 上的普通表转换为分区表：

```
alter table "TEST".TT exchange PARTITION  "上海"  with table "TEST"."TT_上海_1";
alter table "TEST".TT exchange PARTITION  "云南"  with table "TEST"."TT_云南_2";
alter table "TEST".TT exchange PARTITION  "内蒙"  with table "TEST"."TT_内蒙_3";
```

至此分区置换完成。

Oracle 集群迁移到 DM 连接 Oracle 端时报错：“xxxORA-12505xxx”

【问题描述】

Oracle 11G RAC 集群迁移到 DM 连接 Oracle 端时报错：Listener refused the connection whth the following error:ORA-12505, Thn:listener does not currently know of SID Giver in connect descriptor。

【问题解决】

出现此报错需要指定 Oracle 驱动，自定义 url：jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=IP)(PORT= 端口)))(CONNECT_DATA=(SERVICE_NAME= 服务名)))。

参数说明：
DESCRIPTION：描述
ADDRESS_LIST：地址列表
ADDRESS：地址
PROTOCOL：使用的网络协议
HOST：Oracle 数据库地址
PORT：Oracle 端口
CONNECT_DATA：连接数据库名称
SERVICE_NAME：Oracle 服务名

存储过程执行报错：对象定义被修改

【问题描述】

从 ORACLE 中迁移存储过程到 DM 中，存储过程编译可通过，但在存储过程执行时报错：对象定义被修改。

【问题分析】

出现该问题，一般是存储过程中，对一张表先进行了 DDL 操作（如删除索引，添加主键），而后又对该表进行了 DML 操作（ INSERT|DELETE ），就会报错：对象定义被修改。由于达梦数据库会获取 SQL 所有执行计划，再去执行 SQL， 如果在一个过程里先修改分区表定义，就会修改字典信息，所以在执行后续 DML 的时候和之前获取的计划不一致，所以就会报错。可以将 DML 改为动态方式去执行，因为动态执行时才会生成计划，就不会有影响。

【问题解决】

将存储过程中的 DML（INSERT|DELETE）语句使用动态语句执行，即可解决该问题。

例如：

```
EXECUTE IMMEDIATE 'insert into t1 select * from t10 where rownum=1 ';
```

ISNERT 未使用动态语句：

```
CREATE or REPLACE  PROCEDURE "PP_2"  ( MM  IN  NUMBER) AS MM1     VARCHAR2(1000);
BEGIN
EXECUTE IMMEDIATE 'drop index idx_t1';
insert into t1 select * from t10 where rownum=1 ;
commit;
EXECUTE IMMEDIATE 'create INDEX idx_t1 on t1(c1)';
END PP_2;
CALL PP_2(1);   --报错：对象定义被修改
```

INSERT 使用动态语句：

```
CREATE or REPLACE  PROCEDURE "PP_1"   ( MM  IN  NUMBER) AS MM1      VARCHAR2(1000);
BEGIN
EXECUTE IMMEDIATE 'drop index idx_t1';
EXECUTE IMMEDIATE 'insert into t1 select * from t10 where rownum=1 ';
commit;
EXECUTE IMMEDIATE 'create INDEX idx_t1 on t1(c1)';
END PP_1;
call PP_1(1);    --执行成功
```