DM DBA 手记之 ORACLE 移植到 DM

1 概述

ORACLE 到 DM 的移植主要有以下几个方面的工作：

1．分析待移植系统，确定移植对象。

2．通过数据迁移工具 DTS 完成常规数据库对象及数据的迁移。

3．通过人工完成 PL/SQL 的移植，只需要做少量的修改即可。

4．移植完成后对移植的结果进行校验，确保移植的完整性和正确性。

5．对应用系统进行移植、测试和优化。

2 移植过程

2.1 待移植系统分析

应用后台操作系统	Red Hat Linux
数据库后台操作系统	Red Hat Linux
后台数据库	ORACLE
应用开发平台	JAVA
应用开发接口	JDBC
需要移植的数据库对象	序列 表（数据量）、分区表 视图、物化视图 自定义类型 触发器 同义词 存储过程、函数、包

对待移植系统进行分析，确定需要移植的数据库对象，给出移植列表，给用

户确认，作为移植的依据，给出 oracle 的统计脚本。

2.1.1 统计 oracle 数据库基础信息

```
--统计页大小
select name,value from   v$parameter where name ='db_block_size';
--查询编码格式
select * from v$nls_parameters a where a.PARAMETER='NLS_CHARACTERSET';
```

2.1.2 统计 oracle 数据中的对象以及表数据量

```
--根据指定用户统计用户下的各对象类型和数目
select object_type,count(*) from all_objects where
owner='OA8000_DM2015' group by object_type;
```

```
--创建移植辅助表，统计指定用户下所有的对象并插入到辅助表中
create table oracle_objects(obj_owner varchar(100),
obj_name varchar(100),obj_type varchar(50));
insert into oracle_objects select owner,object_name,object_type from
all_objects where owner='OA8000_DM2015';
select * from oracle_objects;
```

```
--创建移植辅助表，统计每个表的数据量并插入到移植辅助表中
create table oracle_tables(tab_owner varchar(100),
tab_name varchar(100),tab_count int);


begin
      for rec in (select owner,object_name from all_objects where
         owner='OA8000_DM2015' and object_type='TABLE') loop
         begin
               execute immediate 'insert into oracle_tables select '''||
rec.owner ||''','''|| rec.object_name ||''',count(*) from '|| rec.owner
```

```
|| '.' || rec.object_name;
            exception when others then
                dbms_output.putline( rec.owner || '.' || rec.object_name || 'get count error');
            end;
            end loop;
end;

select * from oracle_tables;
```

2.2 准备移植环境

本节讨论的内容是关于对移植环境的准备工作，鉴于移植工作最终的目的可能不同，我们需要对目的做一下分类，分类之后，可以更好的明确我们的环境准备工作的需求，从而使使移植的工作更加的高效；

（1）仅做移植兼容性测试。这里指的是用户或者开发商对与移植可能性和技术工作量的一个评估和确认工作，也就是尝试性的移植，移植后可能辨不会立刻进行产品级的应用功能、性能、稳定性测试，在这种情况下，我们一般搭建最基础的移植环境即可，用虚拟机和物理机服务器都可以进行，且对配置无特别要求，满足基本运行条件即可；

（2）为替换 ORACLE 上线运行进行正式移植。在这种情况下，移植完成后，会对应用进行产品级全方位的功能点测试、性能测试、压力测试以及稳定性测试等集成测试，在这种情况下搭建移植环境，一定要优先采用物理服务器搭建，并且对于物理服务器的相关硬件配置要提出要求，提出要求的配置根据系统数据量规模、性能要求、并发规模、可用性要求等基本情况向测试方提出建议，因为要想移植后测试效果好，硬件的支撑是必不可少的，如果是在很低配置情况下，又想得到很好的移植和测试效果，这本身就是不合理的。

对于数据库服务器配置要求，包括 CPU、内存、OS、磁盘（本地盘、阵列），架构方面（单机、多机）包括采用的集群架构方式。

2.2.1 DM 移植环境

- **选择合适的版本**

达梦数据库内部会有定期的版本更新说明和版本发版通知，在进行项目移植的之前，一定要先根据内部通报情况和自己所在技术团队的讨论，确定一个版本，尽量以最新版本且无额外另行通知的版本，保证已经出现的问题，在即将移植的系统中不再出现；

版本优先选择完整安装版本（无完整安装版本的平台例外），避免数据库客户端和服务器端存在版本不匹配带来的额外工作量，达梦在不同平台的不同版本上，安装包都会有差异，一定要采用严格匹配的原则，除非得到达梦原厂技术人员的允许，尽量减少干扰性的问题出现。

- **选择合适的初始化参数**

初始化库，关键的点在于对初始化参数的设置，本章节明确是从 Oracle 移植到 DM 数据库，所以具体的初始化参数建议如下：

（1）关于页大小 PAGE_SIZE。Oracle 也叫块大小（block），在 DM 数据库中，页大小可以为 4KB、8KB、16KB 或者 32KB，从 Oracle 移植到 DM，建议设置页大小为 8KB， 一旦创建好了数据库，在该库的整个生命周期内，页大小都不能够改变。除了每个字段的最大长度限制外，每条记录总长度不能大于页面大小的一半。如果系统中存在或者以后可能存在含有较长的字符串类型的表，建议该参数设置为 16 或者 32。页大小设置越大，最后数据文件的物理大小就会越大，系统运行时，每次从磁盘调入内存的数据单位也就越大，所以此处要慎重。

（2）关于簇大小 EXTENT_SIZE。数据文件使用的簇大小，即每次分配新的段空间时连续的页数，只能是 16 页或 32 页，缺省使用 16 页，从 ORACLE 移植到 DM 使用默认值就可。

（3）关于大小写敏感 CASE_SENSITIVE。DM 为了兼容不同的数据库，在初始化数据库的时候有一个参数字符串比较大小写敏感，用于确定数据库对象及数据是否区分大小写， 默认为区分，不可更改。建议 MYSQL 和 SQLSERVER 迁移过来的系统，使用大小写不敏感， ORACLE 迁移过来的系统，使用大小写敏感，以便和原来系统匹配。

（4）关于字符集 CHARSET。建议采用默认值 GB18030，如果需要国际字

符可以采用 Unicode，GB18030 数字字母占 1 个字节，普通汉字占 2 个字节，部分繁体及少数民族文字占 4 字节，Unicode 在达梦中采用 UTF-8 编码格式，欧洲的字母字符占 1 到 2 个字节， 亚洲的大部分字符占 3 个字节，附加字符为 4 个字节。如果只存储中文和字母数字，一般来说 GB18030 更节省空间一些。

（5）BLANK_PAD_MODE 空格填充模式，默认是 0，从 Oracle 移植要设置为 1。

- **合理配置 INI 参数**

DM 的 INI 参数文件中针对从 ORACLE 移植到 DM，有几个专门的参数，这里将详细介绍。

Compatibility	使用效果及建议
COMPATIBLE_MODE	是否兼容其他数据库模式。0：不兼容，1： 兼容 SQL92 标准，2：兼容 ORACLE，3：兼容 MS SQL SERVER，4：兼容 MYSQL，5：兼容 DM6，6：兼容 Teradata
CALC_AS_DECIMAL	整数相除是否保留小数位，修改为 1

在 INI 参数的 compatibility 部分，还有其它的一些参数，在涉及到之前，尽量保持默认值，在移植准备的环节，先只调整这个参数就可以了，其它参数，在移植过程中，遇到了，再具体分析。

- **创建用户和表空间**

从 Oracle 移植到 DM，要求必须创建新的用户和表空间，不要把数据迁移到系统管理员 SYSDBA 用户下和 MAIN 表空间下。

首先需要分析本次移植 Oracle 源库需要移植的是哪一个或者哪几个用户的数据，然后分别创建这些需要移植的用户和对应的表空间；大多数情况下，我们需要移植的 ORACLE 实例中可能存在有大量的用户，并不是所有的用户对象都是需要我们移植的，所以在移植准备阶段，一定要和相关技术负责人员沟通明确清楚。

2.2.2 ORACLE 移植环境

在从 Oracle 向 DM 进行移植准备阶段，也需要注意 Oracle 的移植环境：

严禁在生产环境中直接迁移。因为移植首先是一个测试的工作，所以移植应

该避免从 Oracle 生产环境数据库中直接进行移植，需要提前向应用开发商提出需要搭建一个测试环境，准备 Oracle 需要移植的环境和数据。直接从生产库上进行数据移植，有很多风险存在，例如会影响生产库的效率，引发崩溃的可能等等。

工欲善其事必先利其器，推荐使用 pl/sql developer 具进行移植测试工作，当然也可以使用 Oracle 自带的客户端工具。

2.3 常规对象及数据迁移

常规对象指的是序列、表和视图，都可以通过达梦提供的数据迁移工具从 ORACLE 完整的迁移到达梦数据库。

2.3.1 制定迁移计划

根据 2.1 节分析的情况，制定迁移计划：

（1）选择合理的迁移顺序：先迁移序列、再迁移表、最后迁移视图。

（2）对于数据量大的表单独迁移。

（3）对于分区表如果数据量没有超过 1 亿建议迁移成普通表，在分区列上创建索引。

（4）对于大字段较多的表，需要修改批量的行数，以免造成迁移工具内存溢出。

2.3.2 序列对象迁移

序列对象一般不依赖其他模式对象，而被其他如表、过程所依赖，所以一般迁移过程中，最先迁移序列对象。

- 迁移序列方法

（1）使用 DTS 工具迁移序列，此方法适用于批量迁移序列对象。DTS 工具使用方法可参考帮助->帮助主题。

（2）从源 Oracle 中获取序列定义，在目的库手动执行序列创建，此方法适用于所需迁移对象较少，或者对方法一中迁移出错的序列单独处理。

- 序列对象迁移可能遇到的问题及注意点

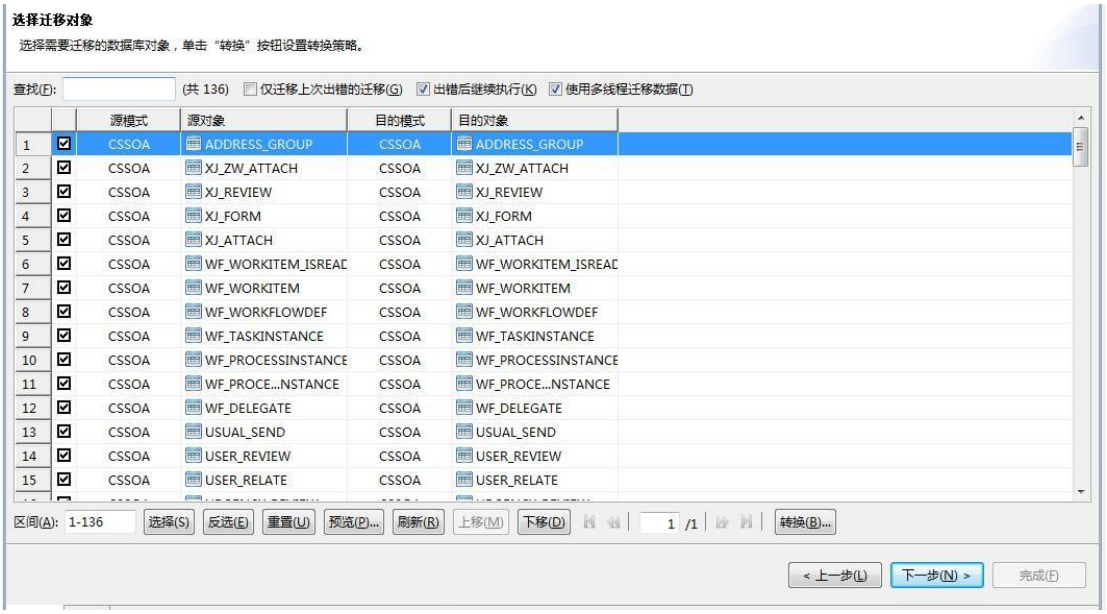
（1）序列的当前值是否需要迁移。一般来说，使用 DTS 批量迁移时，会默认设置迁移 后的目的库序列当前值为源库的当前值。手动迁移序列时，需要注意到这点，并更具实际情况确定是否需要设置迁移后目的库序列当前值与源库一致。

（2）源 oracle 序列范围超出目的 DM 序列可定义范围。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。

2.3.3 表对象迁移

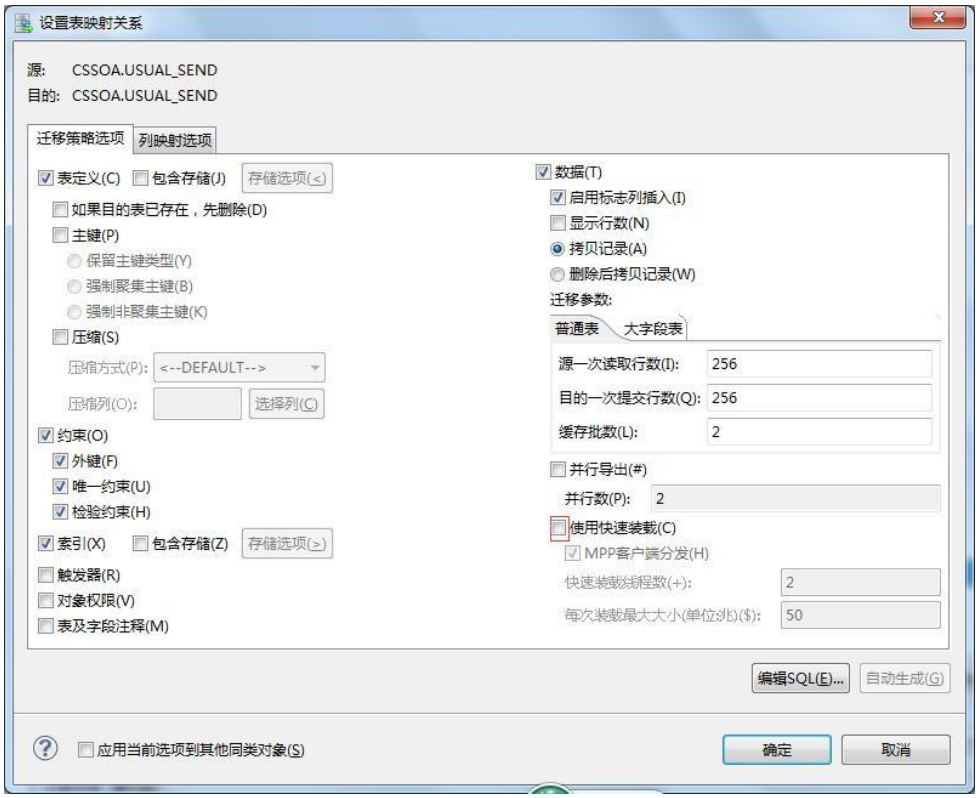
- 一次性迁移

对于表比较少，数据量不大的系统，可以通过 DTS 采取一次性迁移，全部选中即可。



- 分批次迁移

对于表比较多，数据量大的系统，建议先迁移小表再进行大表的迁移，迁移时最好不用快速装载功能。



2.3.4 视图对象迁移

- **普通视图对象迁移**

使用 DTS 工具迁移视图，此方法适用于批量迁移视图对象。DTS 工具使用方法可参考帮助-帮助主题。

从源 Oracle 中获取视图定义，在目的库手动创建视图，此方法适用于所需迁移对象较少，或者对方法一中迁移出错的视图单独处理。

- **物化视图对象迁移**

（1）使用 DTS 工具迁移物化视图，此方法适用于批量迁移物化视图对象。

（2）从源 Oracle 中获取物化视图定义，在目的库手动创建物化视图，此方法适用于所 需迁移对象较少，或者对 a 方法中迁移出错的物化视图单独处理。

- **视图对象迁移可能遇到的问题及注意点**

（1）视图查询对象依赖迁移及权限授予。由于视图查询依赖于相关的表或者其他数据库对象，在迁移视图之前需要首先迁移视图所依赖的对象，并授予视图用户相关对象的权限。

（2）目的端物化视图刷新方式支持及设置。由于目的端 DM 视具体架构（单

库、MPP）的不同存在对物化视图日志表的支持程度的差异，迁移物化视图时，需要视目的端架构，考虑是否变更物化视图刷新方式（增量刷新改为全量刷新）。

2.3.5 处理迁移过程中错误

- **标度大于精度**

Oracle 中 number（m,n）允许 n>m，但是在达梦中是不允许的，达梦中 m>=n，达梦中 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，对达梦中的数据类型进行修改。

- **连接尚未建立或已经关闭**

这个问题有可能是因为 Oracle 库中存在非法的数据如：-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（2019-7 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。

- **序列最大值超出达梦范围**

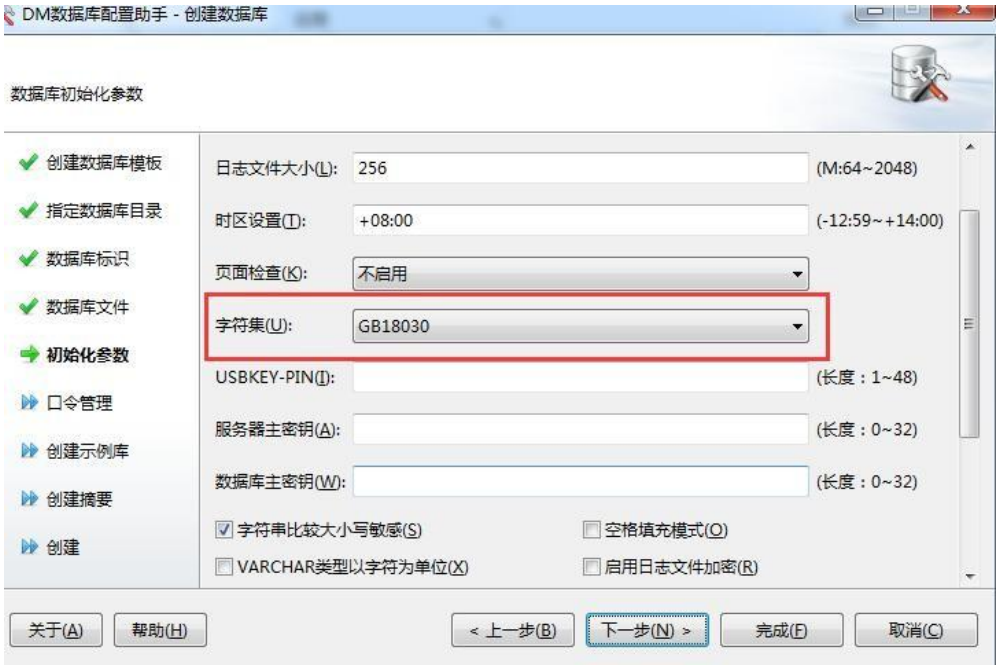
〈最大值〉指定序列能生成的最大值，如果忽略 MAXVALUE 子句，则降序序列的最大值缺省为-1，升序序列的最大值为 9223372036854775807（0x7FFFFFFFFFFFFFF F ）。非循环序列在到达最大值之后，将不能继续生成序列数；但是 oracle 中最大值 28 个 9，迁移到 DM 时报序列最大值超出达梦范围的错误（最新的 dts 版本已经直接将超过 DM 最大值的序列的最大值转换成 9223372036854775807）。

- **字符串截断**

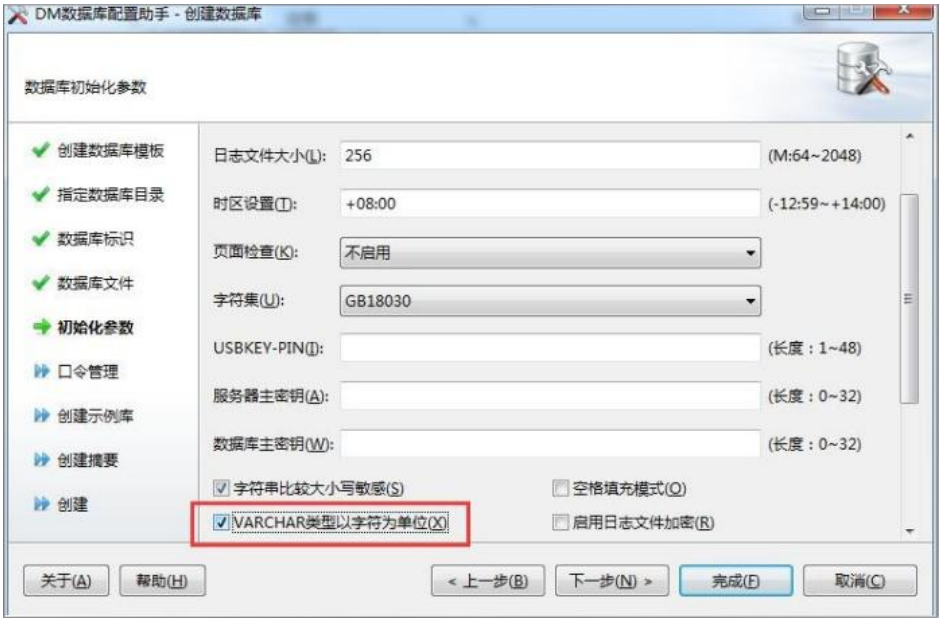
一般从 oracle 迁移到 DM 的时候，出现字符串截断的一般都是字段中含有中文，出现这种问题是因为 DM 初始化的时候选择的字符集是 Unicode（即 utf-8），该字符集的国际标准是一个汉字占 3 个字节，而 oracle 中默认情况下一个汉字占 2 个字节，此时迁移的时候就会报下面的错误：

遇到该错误有 3 种解决办法。

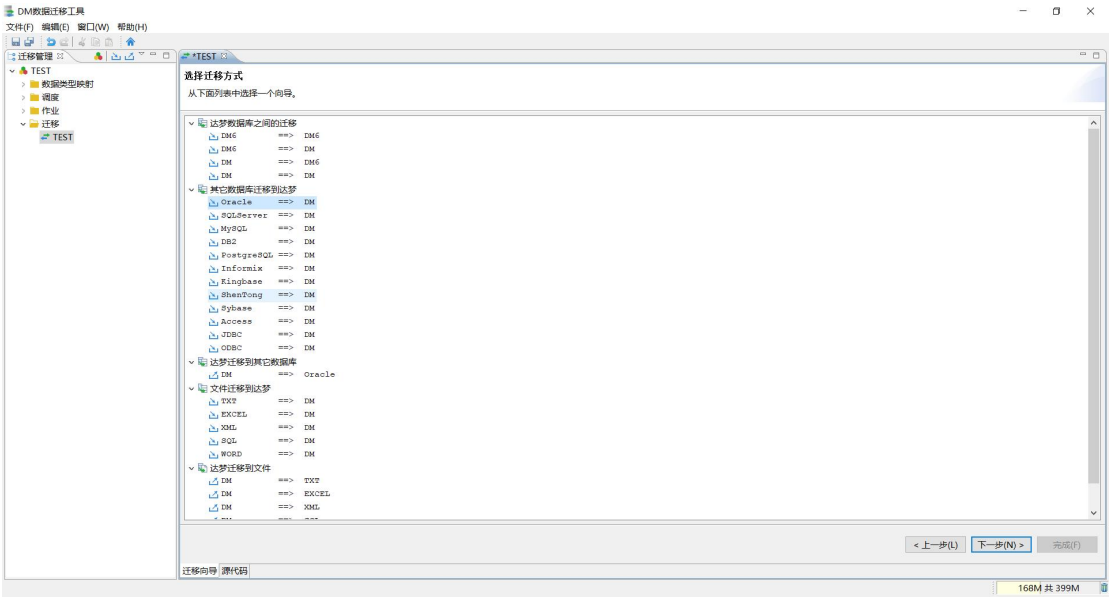
（1）是在初始化的时候，字符集选择 gb18030，如下图：



（2）是选择 VARCHAR 类型以字符为单位，如图：



（3）因为前面 2 种都需要重新初始化数据库，第三种不需要重新初始化数据库即可解决，即在选择迁移方式的时候，选择字符长度隐射关系为 2，如下图：



- **记录超长**

DM 在初始化的时候，选择的页大小影响后面表每行数据的长度，表每行的长度之和（普通数据类型）不能超过一页大小，如果超过 1 页大小即报记录超长的错误，如下图：

解决办法：

（1）找到表中 varchar 类型比较长的（如 varchar2（8000）这种），修改成 text 类型；

（2）初始化的时候页大小选择 32k。（对于表中 varchar2 类型较长，并且字段较多的情况不太适合，这种情况采用方法 1 解决。）

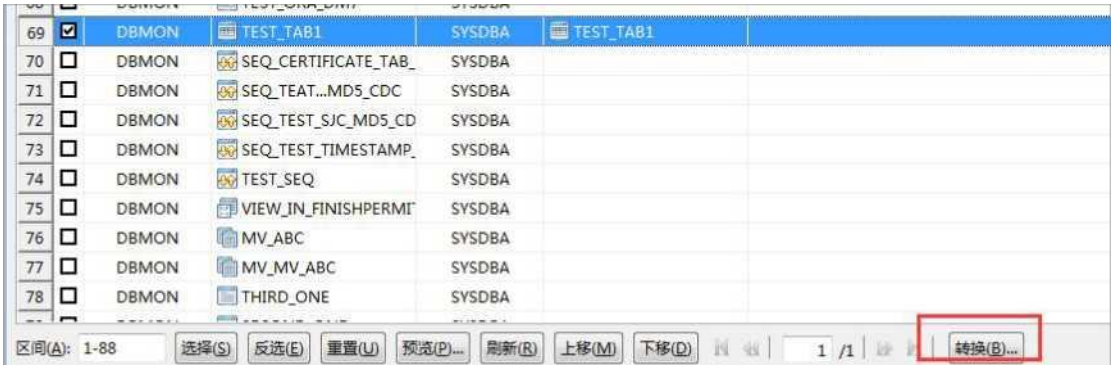
- **违反唯一性约束**

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造成的。这种情况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。

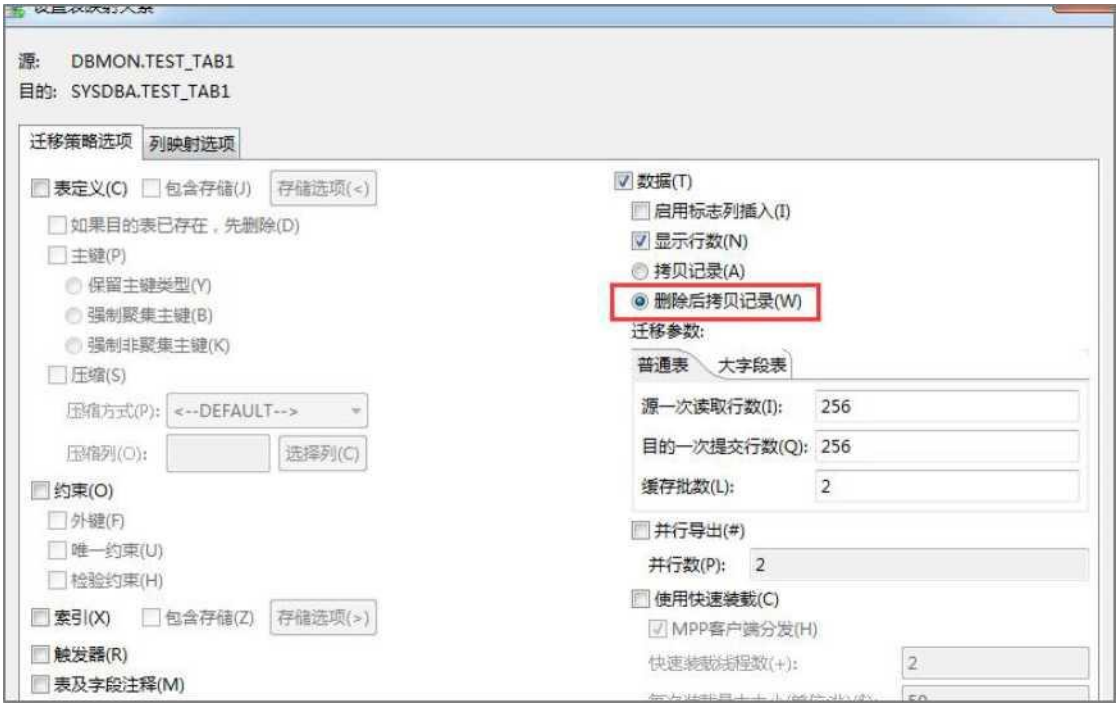
解决办法：

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：

在迁移界面中，先中要迁移的表，然后点击转换.



在弹出的窗口中选择删除后拷贝，如下图：



这样即可迁移成功。

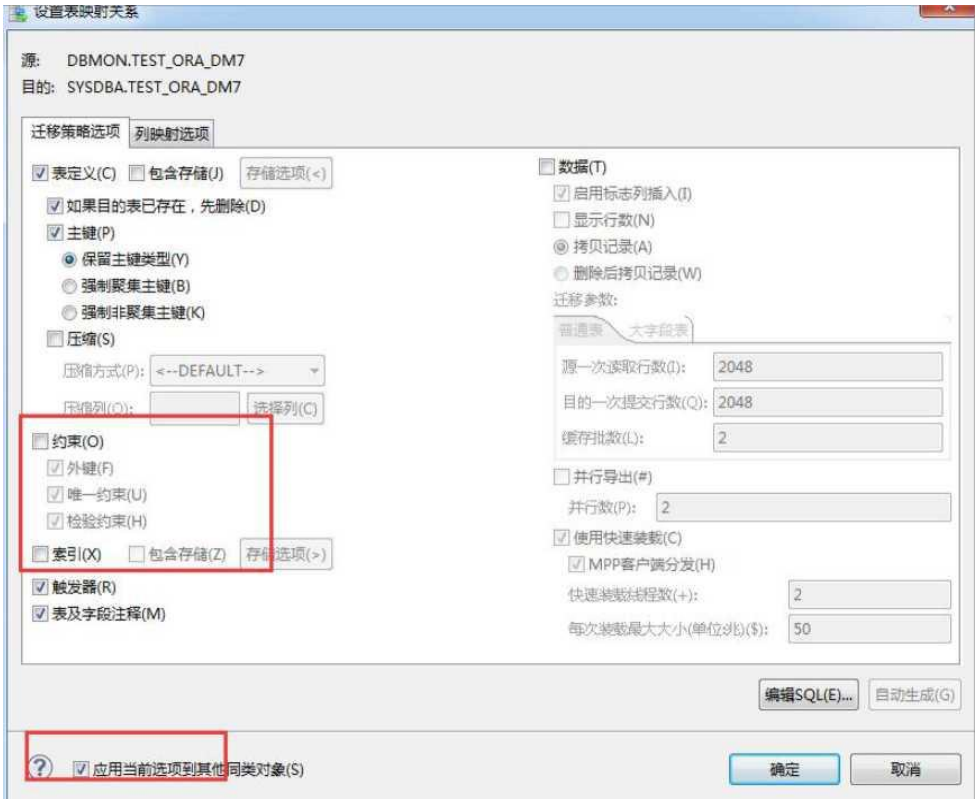
- **违反引用约束**

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，错误如图所示：

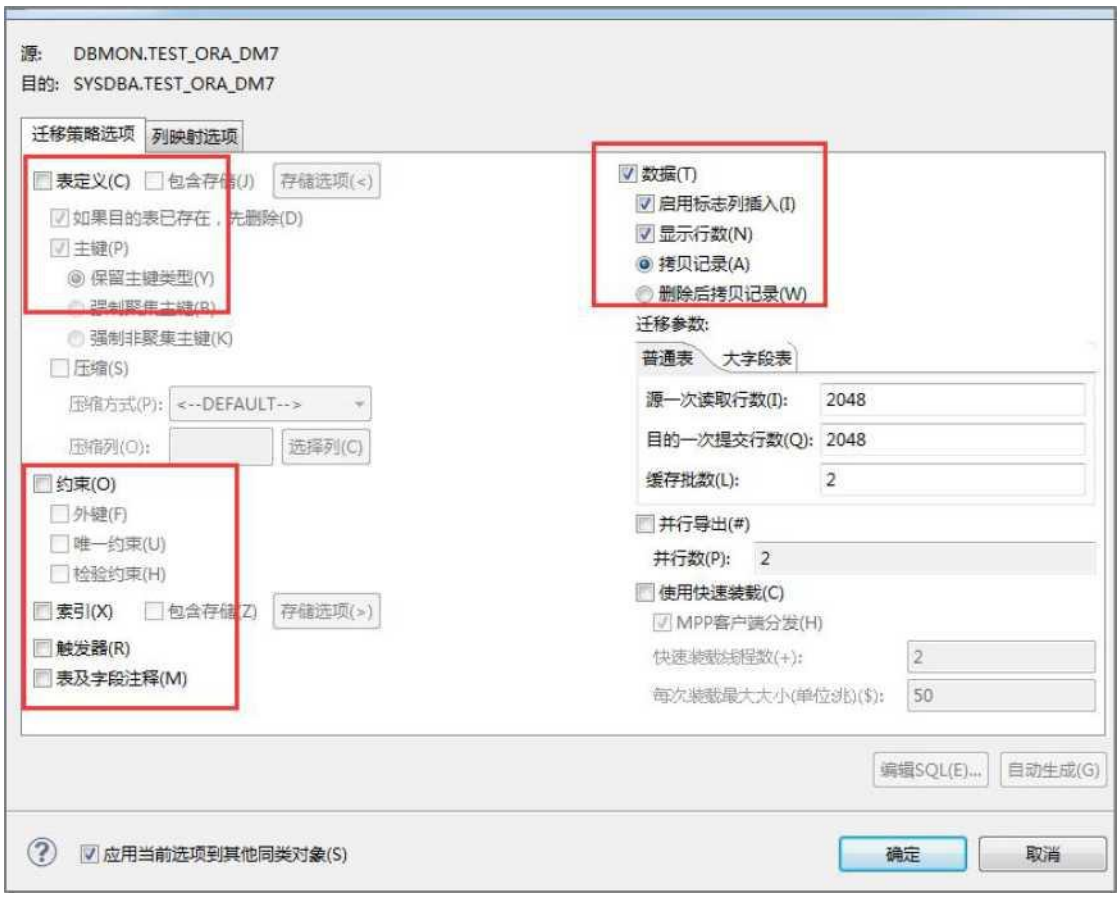
解决办法：

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，按照下面步骤迁移。

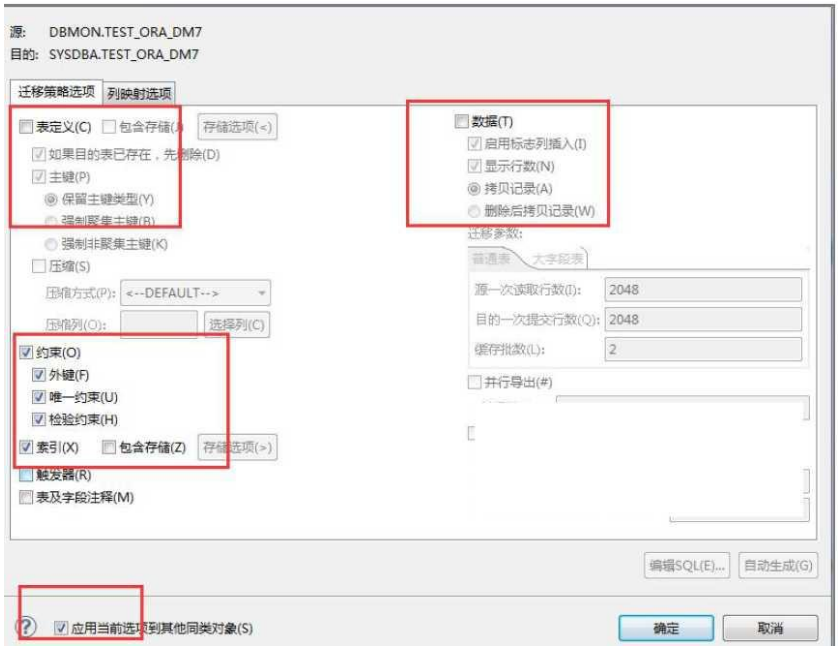
（1）第一次只选择表定义，不选择约束等，如图：



（2）第一次迁移完成后(确保没有错误)，第二次只选择数据，如图：



（3）第三步选择约束、索引等，如图：

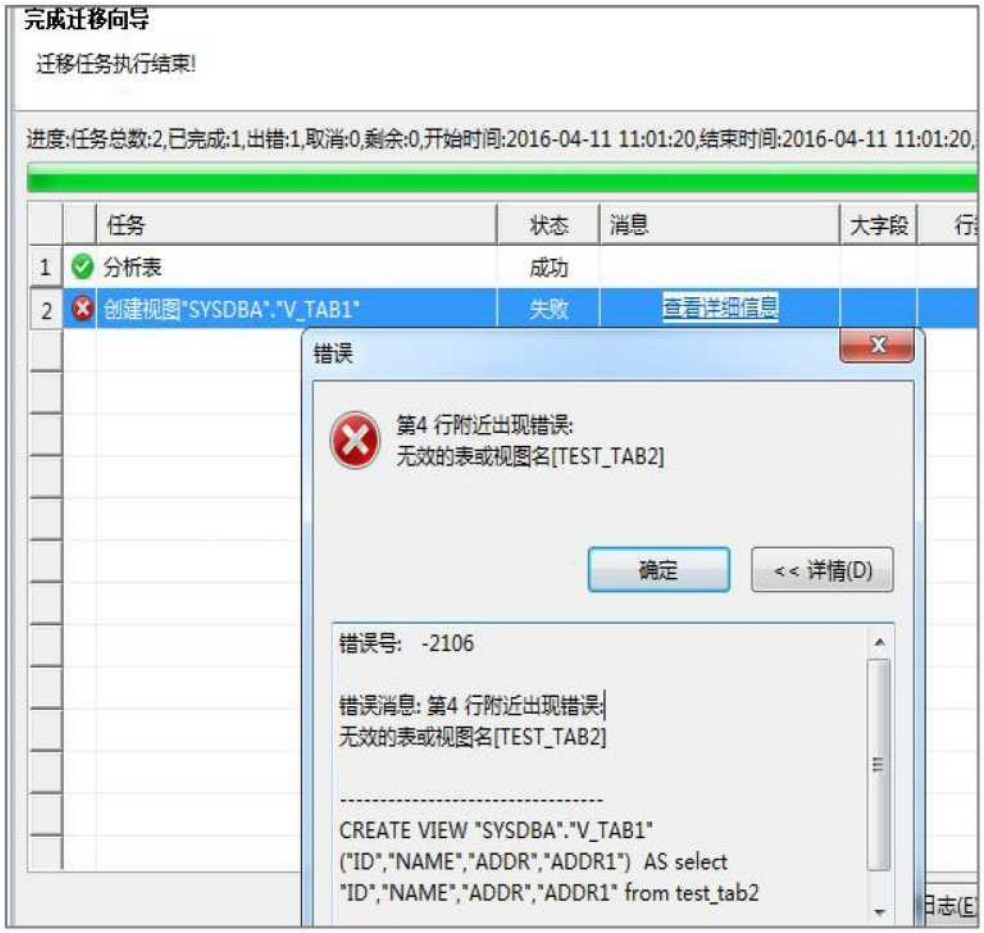


按照上面三步迁移，基本上都可以顺利迁移成功。

- **视图迁移过程中顺序问题：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，如图

所示：



解决办法：

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

- **错误码及错误描述信息的对应**

错误码	信息描述
0	字符串截取
-2665	记录超长
-6602	违反唯一性约束
-6607	违反引用约束
-2107	无效的对象

2.4 PL/SQL 移植

接下来对自定义类型、存储过程、函数、触发器进行移植，可以使用达梦

DTS 工具进行迁移，如果遇到大量错误难以排查，也可以使用如下的方法进行分步迁移。

2.4.1 导出待移植对象的脚本

- **方法一：使用 PL/SQL DEVELOPER 等工具导出自定义类型、存储过程、函数、触发器等 PL/SQL 脚本**

在 PL/SQL 工具->导出用户对象，可以看到对象名称和类型，可以分类导出到多个文件并整理，以便排查问题。

- **方法二：在 sqlplus 中使用 ORACLE 自带的包导出**

导出某个 SCHEMA 的全部触发器：

```
set pagesize 0
set long 90000
set feedback off
set echo off
spool triggers.sql
select DBMS_METADATA.GET_DDL ( ' TRIGGER' ,u.object_name, 'DMHS ') from all_objects u where owner= ' DMHS ' and object_type = ' TRIGGER ';
spool off;
```

导出存储过程：

```
set pagesize 0
set long 90000
set feedback off
set echo off
spool procedures.sql
select DBMS_METADATA.GET_DDL ( ' PROCEDURE ' ,u.object_name, 'DMHS ')
from all_objects u where owner= ' DMHS ' and object_type = ' PROCEDURE';
spool off;
```

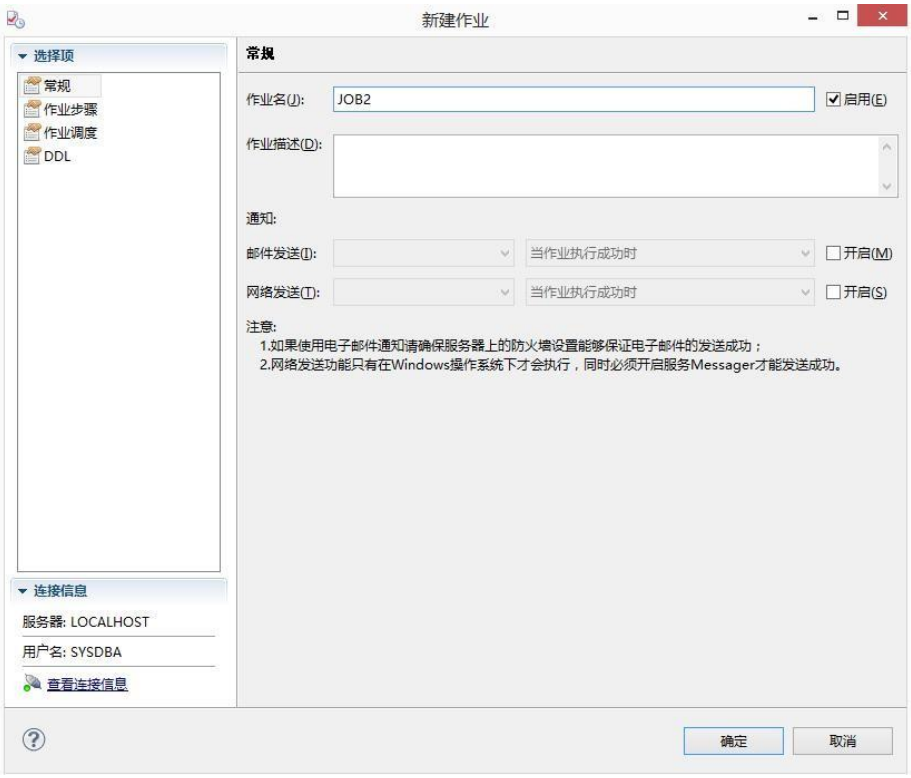
导出自定义函数：

```
set pagesize 0
set long 90000
set feedback off
set echo off
spool functions.sql
select DBMS_METADATA.GET_DDL ( 'FUNCTION' ,u.object_name, 'DMHS') from
all_objects u where owner= ' DMHS ' and object_type = ' FUNCTION ';
spool off;
```

同理，其他类型如 TYPE，PACKAGE，PACKAGEBODY 等也可以用这种方法导出。

ORACLE 的作业：

由于和达梦机制不同，需要了解具体意义后在达梦的代理中重新配置，可以在最后进行该部分的移植：使用管理工具登录，代理->创建代理环境->作业->新建作业->根据需求配置作业步骤和作业调度即可。



2.4.2 运行脚本并处理错误

根据依赖顺序在达梦中执行并分析错误。

- **错误的包或类名：**

```
[执行语句1]:
create or replace PROCEDURE p1()
as
begin
adbms_output.new_line;
end;
执行失败(语句1)
第4 行附近出现错误:
错误的包或类名或当前环境不支持 [ADBMS_OUTPUT]
```

（1）检查是否依赖的包或类未创建

（2）如果使用 ORACLE 系统包可以在达梦中使用 SP_CREATE_SYSTEM_PACKAGES(1)语句创建。

（3）如果创建后也没有，需要根据具体情况改写，如 dbms_output.put_line 在达梦中可以使用 PRINT 代替。

- **无效的方法**

```
[执行语句1]:
create or replace PROCEDURE p1()
as
begin
dbms_output.new_line;
end;
执行失败(语句1)
第4 行附近出现错误:
无效的方法名[NEW_LINE]
```

（1）检查是否依赖的方法未创建

（2）如果使用ORACLE系统方法可以在达梦中使用SP_CREATE_SYSTEM_PACKAGES(1)语句创建

（3）如果创建后也没有，需要根据具体情况改写。

- **非法的基类名**

```
create or replace PROCEDURE p1()
as
a NVARCHARr;
begin
print('a');
end;
执行失败(语句1)
第3 行附近出现错误:
非法的基类名[NVARCHARR]
```

需要注意达梦和ORACLE部分数据类型的精度和意义是不一样的，需要进行改写。

ORACLE	DM
LONG	TEXT
DATE	TIMESTAMP
NVARCHAR2	VARCHAR2（最新版本可以兼容NVARCHAR2）
LONG RAW	VARBINARY

- **语法分析错误**

```
[执行语句1]:
create or replace PROCEDURE p1()
as
begin
select sysdate link from dual;
end;
执行失败(语句1)
第 4 行，第 16 列[link]附近出现错误:
语法分析出错
```

（1）由于使用达梦的保留字冲突导致，建议如果可以更换尽量更换，如果不行可以采用屏蔽关键字的方法进行屏蔽。

（2）字符兼容问题，是否使用了中文的标点符号

（3）ORACLE 和 DM 有语法不一致的地方，需要根据具体问题具体分析，例如：

ORACLE	DM
Select "DUMMY" From dual	Select "ID" From dual
to_nchar	to_char
NLS_UPPER	NLS_UPPER

2.5 核对数据库移植结果

- **统计达梦数据基础信息**

```
--统计页大小
select page;
--通过编码格式
select unicode;
--统计大小写敏感参数
select case_sensitive;
```

- **统计达梦数据中的对象以及表数据量**

A. 根据指定用户统计用户下的各对象类型和数目

```
select object_type,count(*) from all_objects where
owner='OA8000_DM2015' group by object_type;
```

B. 统计指定用户下所有的对象，并记录到新的记录表中

```
create table dm_objects(obj_owner varchar(100),obj_name
varchar(100),obj_type varchar(50));
insert into dm_objects select owner,object_name,object_type from
all_objects where owner='OA8000_DM2015';
```

C. 统计每个表的数据量到表数据记录表

```
1.  create table dm_tables(tab_owner varchar(100),tab_name
      varchar(100),tab_count int);


2.begin
for rec in (select owner,object_name from all_objects where
owner='OA8000_DM2015' and object_type='TABLE') loop begin
execute immediate 'insert into dm_tables select '''|| rec.owner
||''','''|| rec.object_name ||''',count(*) from '|| rec.owner || '.'
|| rec.object_name;
exception when others then
print rec.owner || '.' || rec.object_name || 'get count error';
end;
end   loop；
end；
3. select * from dm_tables;
```

- **对比达梦数据库中对象和 oracle 库中对象以及数据量差异**

A. 比对对象，找出没有迁移的对象

```
select * from oracle_objects where (obj_owner,obj_name) not in (
select obj_owner,obj_name from dm_objects
) --and obj_type='TABLE'
```

B. 比对表数据量，找出数据量不相等的表

```
select a.tab_owner,a.tab_name,a.tab_count-b.tab_count from
oracle_tables a, dm_tables b
where a.tab_owner=b.tab_owner and a.tab_name=b.tab_name
```

```
and a.tab_count-b.tab_count<>0
```

2.6 数据库移植完毕后的收尾工作

- **更新统计信息**

数据核对完成无问题后，应进行一次全库的统计信息更新工作。统计信息更新脚本示例如下：

```
DBMS_STATS.GATHER_SCHEMA_STATS( 'HNSIMIS',
--HNSIMIS  为模式名 100,
FALSE,
' FOR ALL COLUMNS SIZE AUTO ' );
```

更新统计信息的目的在于大批量迁移数据后，可能会导致数据库优化器根据错误的统计信息得到错误的查询计划，严重影响查询性能。

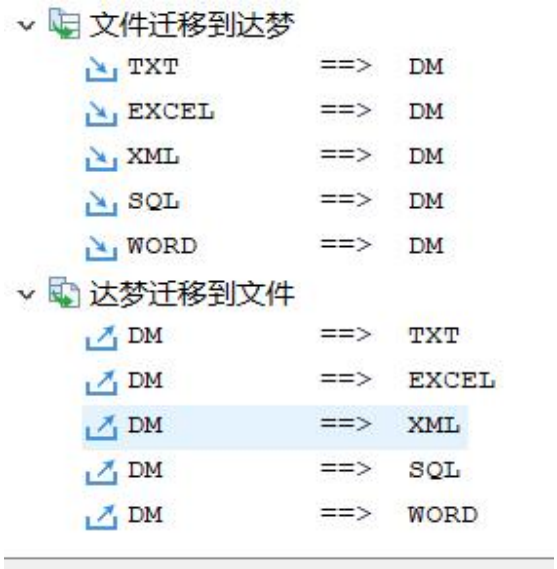
- **数据备份**

在对数据更新完统计信息后，在数据量不大，磁盘空间足够的情况下应进行一次数据备份工作。数据备份有两种方式：正常停止数据库后，拷贝备份 data 文件夹；或者开启归档日志后，进行物理备份。

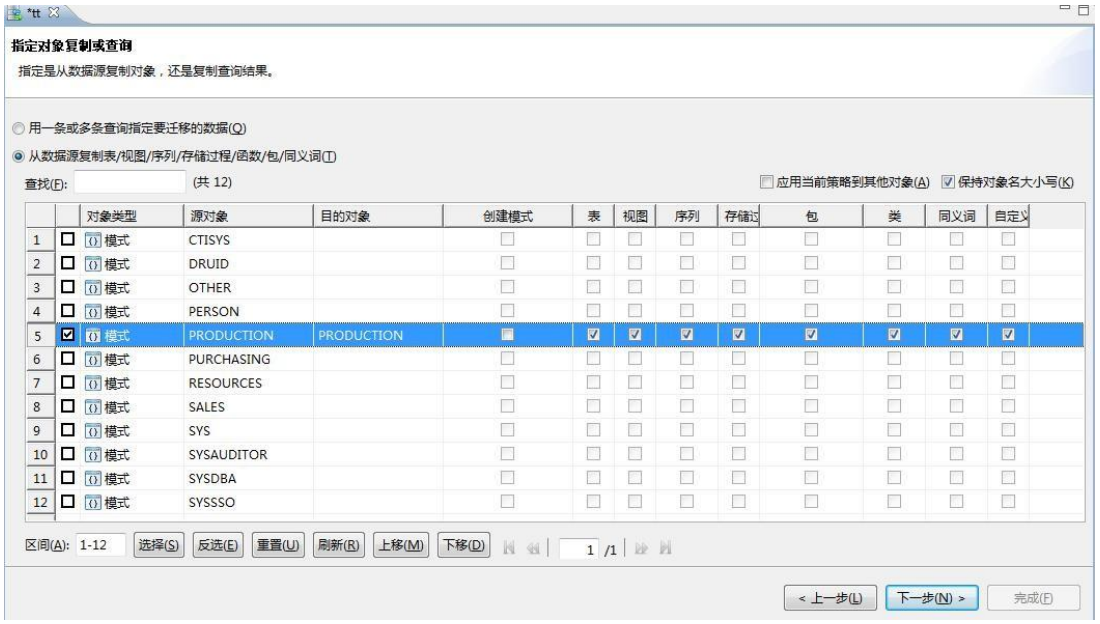
- **整理对象脚本**

整理所有数据库对象脚本，这是为了对项目移植情况进行记录和备份，方便再次进行数据迁移。备份的数据库对象脚本包括：序列定义及当前值，表定义，索引定义，视图定义，函数定义，存储过程定义，包及包体定义、自定义类型和同义词定义。

脚本对象导出可通过达梦数据迁移 DTS 工具来进行。执行步骤如下：


文件迁移到达梦
TXT ==> DM
EXCEL ==> DM
XML ==> DM
SQL ==> DM
WORD ==> DM
达梦迁移到文件
DM ==> TXT
DM ==> EXCEL
DM ==> XML
DM ==> SQL
DM ==> WORD


*tt
文件
请输入SQL脚本文件信息。
定义脚本文件(B): d:\test.sql
浏览(W)...
数据脚本文件:
使用对象定义脚本文件(S)
对每个表使用单独的数据脚本文件(I)
浏览(F)...
所有数据脚本使用一个单独的文件(O)
浏览(F)...
文件编码(E): GBK
迁移选项:
仅迁移对象定义(A)
仅迁移数据(D)
迁移对象定义和数据(C)
定义脚本:
可执行(G)
不可执行(H)


*tt
指定对象复制或查询
指定是从数据源复制对象，还是复制查询结果。
用一条或多条查询指定要迁移的数据(Q)
从数据源复制表/视图/序列/存储过程/函数/包/同义词(T)
查找(F):
(共 12)
应用当前策略到其他对象(A)
保持对象名大小写(K)
对象类型 源对象 目的对象 创建模式 表 视图 序列 存储过 包 类 同义词 自定义
1 模式 CTISYS
2 模式 DRUID
3 模式 OTHER
4 模式 PERSON
5 模式 PRODUCTION PRODUCTION
6 模式 PURCHASING
7 模式 RESOURCES
8 模式 SALES
9 模式 SYS
10 模式 SYSAUDITOR
11 模式 SYSDBA
12 模式 SYSSSO
区间(A): 1-12
选择(S)
反选(E)
重置(U)
刷新(R)
上移(M)
下移(D)
1 /1
< 上一步(L)
下一步(N) >
完成(F)

后面直接点“下一步”即可。

2.7 系统测试与优化

数据库和应用系统移植完毕后开启 sql 日志，对系统进行全面测试，排除移植过程中错误的地方，对慢的 sql 语句进行优化，开启日志的方法如下：

```
--设置 SQL 过滤规则，记录全部的 SQL
SELECT SF_SET_SYSTEM_PARA_VALUE('SQL_TRACE_MASK','1',0,1);
--排查错误的时候也可以设置过滤规则为 23，只记录报错的 SQL SELECT
SF_SET_SYSTEM_PARA_VALUE('SQL_TRACE_MASK','23',0,1);
--开启 SQL 日志：
SP_SET_PARA_VALUE(1, 'SVR_LOG', 1);
```

在功能测试和性能测试的时候可以开启 SQL 日志，然后通过日志分析工具从执行时间和执行次数两个维度对 SQL 日志进行分析，生产分析结果，然后根据分析结果对系统性能进行优化。使用日志分析工具时最好采用 32k 页大小的 DM 作为分析库。

附录、DM 与 ORACLE 11G 对比

1.1 数据类型

1.1.1 有区别的数据类型

重点对比 dm 和 oracle 有区别的数据类型比如:date 等。

Oracle		DM		备注
类型名	说明	类型名	说明	
CHAR(n)	定长字符串，最大长度 2000，长度不足 n 时，自动填充空格	CHAR(n) 、CHARACTER（n）	定长字符串，最大长度 8000（页大小为 16K 或以上时），长度不足 n 时，自动填充空格	DM 允许的串最大长度可通过设置页大小参数分别调整为 1900，3900，8000，16200 之一。
VARCHAR(n) VARCHAR2(n)	可变长字符串，最大长度 4000	VARCHAR(n) 、VARCHAR2(n)	可变长字符串，最大长度 8000（页大小为 16K 或以上时）	
LONG	可变长文本，最大长度 134217727，表中只能有一个 LONG 列	TEXT 、LONGVARCHAR	长文本，最大长度 2G－1,表中可有多个 TEXT/LONGVARCHAR 列	可用于存储较长的非格式化文本,文本较小时可考虑用 Oracle 的 VARCHAR2 或 DM 的 VARCHAR
NUMBER	可变长数值列，	BIT、BOOLEAN	BIT 类型用于存	这些数据类型

	最大数值长度为 38 个数字，NUMBER 是 Oracle 特有的非标准数据类型		储整数数据 1、0 或 NULL 。BOOLEAN 类型的字值可以取常量 TRUE 和 FALSE，或者数值 1 和 0	都不带精度描述。DM 的 BIT 类型与 SQLSERVER 2000 的 BIT 数据类型相似。
		BYTE、TINYINT	1 字节存储的整数，精度 3，标度 0	BIT 、BOOLEAN 都可以用来支持 ODBC 和 JDBC 的布尔数据类型，MONEY 类型便于存储货币数据。
		SMALLINT	2 字节存储的整数，精度 5，标度 0	
		INT、INTEGER	4 字节存储的整数，精度 10，标度 0	
		BIGINT	8 字节存储的长整数，精度 19，标度 0	
		DEC、DECIMAL、NUMERIC 、NUMBER	定点数，精度 16，标度 0	
NUMBER(m,n)	可变长数值列，最大数值长度为 38 个数字	DECIMAL(m,n) 、NUMBER(m,n)、NUMERIC(m,n)		这些数据类型都带精度描述。
FLOAT	浮点数，精度 126	FLOAT、DOUBLE 、DOUBLEPRECISI	浮点数，最大二进制精度 53，十进制精度 15	

		ON		
REAL	实数类型，NUMBER(63)，精度更高	REAL	浮点数，二进制精度 24，十进制精度 7	
(LONG)RAW	可变长二进制数据最大长度 134217727，表中只能有一个 LONGRAW 列	BINARY（n）	定长二进制数据，缺省长度 1，最大长度同 CHAR，长度不足 n 时，自动填充 0	
		VARBINARY（n）	可变长二进制数据，缺省长度 1，最大长度同 CHAR	
		IMAGE、LONGVARBINARY	可变长的长二进制数据，最大长度 2G－1，表中可有多个 IMAGE/LONG VARBINARY 列	可用于存储多媒体图像、WORD 文档等，长度较小时可考虑用 DM 的 VARBINARY
BLOB（n）	可变长二进制大对象，最大长度 4G－1	BLOB（n）	可变长二进制大对象，最大长度 2G－1	
CLOB（n）	可变长二进制大对象，最大长度 4G－1	CLOB（n）	可变长字符串大对象，最大长度 2G－1	
DATE	固定长度(7 字节)的日期型，时	DATETIME	日期型，包括年、月、日、时分秒	时间间隔类型 Oracle、DM 均

	间也作为一部分存储其中。Oracle 的 DATE 是其特有的非标准数据类型。		信息。数据格式：datetime'2013-11-0612:23:22'	支持（略）
		TIME	时间型，包括时、分、秒信息。格式：time’09:10:21’	
		TIMESTAMP	时间戳型，包括年月日时分秒信息。例如：timestamp ‘1999-07-13 10:11:22’	
BFILE	存放在数据库外的二进制数据,最大长度 4G	BFILE	存储在操作系统中的二进制文件，文件存储在操作系统而非数据库中，仅能进行只读访问，最大长度 2G-1。	
NCHAR(n)	根据字符集而定的固定长度字符串,最大长度 2000bytes	NCHAR(n)	根据字符集确定需要扩展的长度，最大长度 8000（页大小为 16K 或以上时）	
NVARCHAR2	根据字符集而定的可变长度字符串,最大长度 4000bytes	NVARCHAR2(n)	根据字符集确定需要扩展的长度，最大长度 8000（页大小为	

			16K 或以上时）	
NCLOB	根据字符集而定的字符数据，最大长度 4G	NCLOB	可变长字符串大对象，最大长度 2G－1	
ROWID	数据表中记录的唯一行号，10bytes	VARCHAR(n) 、VARCHAR2(n)	可变长字符串，最大长度 8000（页大小为 16K 或以上时）	
NROWID	二进制数据表中记录的唯一行号，最大长度 4000bytes	VARCHAR(n) 、VARCHAR2(n)	可变长字符串，最大长度 8000（页大小为 16K 或以上时）	

1.1.2 无区别的数据类型

无区别的数据类型暂无

1.2 常用函数

1.2.1 有区别的函数

序号	ORACLE 函数名	DM 函数名	功能简要说明
数值函数			
1	/	DEGREES(n)	求弧度 n 对应的角度值
2	/	GREAT(n1,n2)	求 n1、n2 两个数中最大的一个
3	/	LOG10(n)	求数值 n 以 10 为底的对数
4	REMAINDER(n2,n1)	/	求余数
5	WIDTH_BUCKET(expr, min_value,max_value,num_buckets)	/	等量分级

6	/	RADIANS(n)	求角度 n 对应的弧度值
字符串函数			
1	/	INS(char1,begin,n,char2)	删除在字符串 char1 中以 begin 参数所指位置开始的 n 个字符,再把 char2 插入到 char1 串的 begin 所指位置
2	/	INSERT(char1,n1,n2,char2)INSSTR(char1,n1,n2,char2)	将字符串 char1 从 n1 的位置开始删除 n2 个字符，并将 char2 插入到 char1 中 n1 的位置
3	/	LEFT(char,n)/LEFTSTR(char,n)	返回字符串最左边的 n 个字符组成的字符串
4	/	OCTET_LENGTH(char)	返回字符串的字节数,等价于 LENGTHB
5	NCHAR	CHAR	n 个字符的固定长度 Unicode 字符数据
6	nvarchar	varchar	n 个字符的可变长度 Unicode 字符数据
7	INSTR	POSITION(char1,/INchar2)	求串 1 在串 2 中第一次出现的位置
8	LPAD,RPAD	REPLICATE(char,times)	把字符串 char 自己复制 times 份
9	substr	RIGHT/RIGHTSTR(char,n)	返回字符串最右边 n 个字符组成的字符串
10	/	SPACE(n)	返回一个包含 n 个空格的字符串
11	/	STRPOSDEC(char)	把字符串 char 中最后一个字符的值减一
12	/	STRPOSDEC(char,pos)	把字符串 char 中指定位置 pos 上的字符值减一
13	/	STRPOSINC(char)	把字符串 char 中最后一个字符的

			值加一
14	/	STRPOSINC(char,pos)	把字符串 char 中指定位置 pos 上的字符值加一
15	/	SUBSTRING(charFROMm[FORn])	返回 char 中从字符位置 m 开始的 n 个字符
16	/	OVERLAY(char1PLACINGchar2FROMint[FORint])	字符串覆盖函数，用 char2 覆盖 char1 中指定的子串，返回修改后的 char1
17	/	TEXT_EQUAL	返回两个 LONGVARCHAR 类型的值的比较结果，相同返回 1，否则返回 0
18	/	BLOB_EQUAL	返回两个 LONGVARBINARY 类型的值的比较结果，相同返回 1，否则返回 0
19	/	NLSSORT	返回对汉字排序的编码
20	/	GREAT(char1,char2)	求 char1、char2 中最大的字符串
21	/	UNISTR(char)	将字符串 char 中，ascii 码（‘\XXXX’4 个 16 进制字符格式）转成本地字符。对于其他字符保持不变。
22	CHARTOROWID	/	将字符数据类型转换为 ROWID 类型
23	NLS_CHARSET_DECL_LEN(byte_width,charset)	/	返回一个 NCHAR 值的声明宽度(以字符为单位).
24	NLS_CHARSET_ID(charset_name)	/	返回指定字符集 charset_name 的数字 ID。
25	NLS_CHARSET_NAME([charset_id])	/	返回指定字符集IDcharset_id的名字。

26	NLS_INITCAP(string[,nlsparams])	/	以字符串中每个单词第一个字符大写而单词中其余字母小写的形式返回 string.nlsparams 指定了一个与该会话缺省的不同的排序次序。
27	NLS_UPPER(string[,nlsparams])	如果没有指定 nlsparams,NLS_UPPER 与 UPPER 相同	以大写形式返回 string,不是字母的字符不受影响。
28	NLS_LOWER(string[,nlsparams])	如果没有指定 nlsparams，NLS_LOWER 与 LOWER 相同	以小写形式返回 string,不是字母的字符不受影响
29	NLSSORT(string[,nlsparams])	/	返回用于排序 string 的字符串字节。
日期类函数			
1	DATE	CURDATE()	返回系统当前日期
2	to_char(sysdate,'hh24:mi:ss')	CURTIME()	返回系统当前时间
3	to_char(sysdate,'hh24:mi:ss')	CURRENT_TIME(n)	返回系统当前时间
4	/	DATEADD(datepart,n,date)	向指定的日期加上一段时间
5	/	DATEDIFF(datepart,date1,date2)	返回跨两个指定日期的日期和时间边界数
6	/	DATEPART(datepart,date)	返回代表日期的指定部分的整数
7	TO_CHAR(TO_DATE(date,'YYYY/MM/DD'),'D')	DAYNAME(date)	返回日期的星期名称
8	Extract(day from date),	DAYOFMONTH(dat	返回日期为所在月份中的第几天

		e)	
9	TO_CHAR(TO_DATE(date,'YYYY/MM/DD'),'D')	DAYOFWEEK(date)	返回日期为所在星期中的第几天
10	Extract(year from date),	DAYOFYEAR(date)	返回日期为所在年中的第几天
11	floor(date2- date1)	DAYS_BETWEEN(date1,date2)	返回两个日期之间的天数
12	SYSDATE	GETDATE()	返回系统当前时间戳
13	to_char(sysdate,'hh24')	HOUR(time)	返回时间中的小时分量
14	to_char(sysdate,'mi')	MINUTE(time)	返回时间中的分钟分量
15	to_char(sysdate,'mm')	MONTH(date)	返回日期中的月份分量
16	to_char(sysdate,'mm')	MONTHNAME(date)	返回日期中月分量的名称
17	Sysdate	NOW()	返回系统当前时间戳
18	to_char(sysdate,'q')	QUARTER(date)	返回日期在所处年中的季节数
19	to_char(sysdate, 'ss')	SECOND(time)	返回时间中的秒分量
20	/	TIMESTAMPADD(interval,n,time stamp)	返回时间 timestamp 加上 n 个 interval 类型时间间隔的结果
21	/	TIMESTAMPDIFF(interval,timeStamp1,timestam p2)	返回一个表明 timestamp2 与 timestamp1 之间的 interval 类型时间间隔的整数
22	/	SYSTIMESTAMP(n)	返回系统当前带数据库时区信息的时间戳
23	/	WEEK(date)	返回日期为所在年中的第几周
24	/	WEEKDAY(date)	返回当前日期的星期值
25	/	WEEKS_BETWEEN(date1,date2)	返回两个日期之间相差周数
26	to_char(sysdate,'yyyy')	YEAR(date)	返回日期的年分量
27	/	YEARS_BETWEEN(date1,date2)	返回两个日期之间相差年数
28	DBTIMEZONE	/	数据库服务器所在的时区

29	sysdate	LOCALTIME()	返回系统当前时间
30	NEW_TIME(dt1,c1,c2)	LOCALTIME()	给出时间 dt1 在 c1 时区对应 c2 时区的日期和时间
31	ORA_DST_AFFECTED(datetime_expr)	/	修改数据库时区时使用
32	ORA_DST_CONVERT(datetime_expr [, integer [, integer]])	/	修改数据库时区时使用
33	ORA_DST_ERROR(date time_expr)	/	修改数据库时区时使用
34	SCN_TO_TIMESTAMP(number)	/	将 SCN 转成 TIMESTA MP
35	SESSIONTIMEZONE	/	返回当前 session 时区
空值判断函数			
1	/	ISNULL(n1,n2)	当 n1 为非空时，返回 n1；若 n1 为空，则返回 n2
2	/	NULL_EQU	返回两个类型相同的值的比较
3	lnnvl	/	用于 where 子句中的条件，如果条件为 true 就返回 false；如果条件为 UNKNOWN 或者 false 就返回 true。该函数不能用于复合条件如 AND,OR,orBETWEEN 中
类型转换函数			
1	/	BINTOCHAR(exp)	将 exp 转换为 CHAR
2	RAWTONHEX(raw)	/	等价于 TO_NCHAR(RAWTOHEX(raw))
集函数			
1	STATS_BINOMIAL_TE ST(expr1,expr2,p,TWO_	/	统计二项测试是一个精确概率测试用于二分变量,那里只有两个可

	SIDED_PROB|EXACT_PROB|ONE_SIDED_PROB_OR_MORE|ONE_SIDED_PROB_OR_LESS)		能值存在。它测试一个样品之间的差异比例和给定的比例
2	STATS_CROSSTAB(expr1,expr2,CHISQ_OBS|CHISQ_SIG|CHISQ_DF|PHI_COEFFICIENT|CRAMERS_V|CONT_COEFFICIENT|COHENS_K)	/	用于分析两个变量
3	STATS_F_TEST(expr1,expr2,[STATISTIC|DF_NUM|DF_DEN|ONE_SIDED_SIG|,expr3]|TWO_SIDED_SIG)	/	测试两个方差是否有明显的不同
4	STATS_KS_TEST (expr1 , expr2,STATISTIC|SIG)	/	是柯尔莫哥洛夫斯米尔诺夫函数比较两个样品测试他们是否来自相同的总体或有相同分布的总体
5	STATS_MODE (expr)	/	统计模式需要一组值作为它的参数并且返回发生以最大的频率
6	STATS_MW_TEST (expr1 , expr2,STATISTIC|U_STATISTIC|ONE_SIDED_SIG, expr3|TWO_SIDED_SIG)	/	一个曼惠特尼测试比较两个独立样本来测试该无效假设，这两个种群具有相同的分布函数与替代并且假设这两个分布函数是不同的
7	STATS_ONE_WAY_ANOVA(expr1 ,	/	单向方差分析函数(统计一维方差分析) 测试差异在意味着(为团体

	expr2,SUM_SQUARES_BETWEEN|SUM_SQUARES_WITHIN|DF_BETWEEN|DF_WITHIN|MEAN_SQUARES_BETWEEN|MEAN_SQUARES_WITHIN|F_RATIO|SIG)		或变量),通过比较两种统计学意义不同的估计方差
8	STATS_T_TEST_*	/	若干函数，不一一列举
9	STATS_WSR_TEST (expr1 ,expr2,STATISTIC|ONE_SIDED_SIG|TWO_SIDED_SIG)	/	
10	/	AREA_MAX	求区间范围内最大值集函数
分析函数			
1	CORR_K(expr1,expr2,COEFFICIENT|ONE_SIDED_SIG|ONE_SIDED_SIG_POS|ONE_SIDED_SIG_NEG|TWO_SIDED_SIG));	/	协方差函数,相关系数
2	CORR_S(expr1,expr2,COEFFICIENT|ONE_SIDED_SIG|ONE_SIDED_SIG_POS|ONE_SIDED_SIG_NEG|TWO_SIDED_SIG));	/	协方差函数,相关系数
3	MEDIAN	/	返回一组数据的中间值

	(expr)OVER(query_partition_clause)		
4	REGR_SLOPE	/	线性回归
5	REGR_INTERCEPT	/	线性回归
6	REGR_COUNT	/	线性回归
7	REGR_R2	/	线性回归
8	REGR_AVGX	/	线性回归
9	REGR_AVGY	/	线性回归
10	REGR_SXX	/	线性回归
11	NTH_VALUE(measure_expr,n)	/	直接返回任意行的值
XML 函数			
1	XMLAGG (XMLType_instance order_by_clause)	/	返回 Xml 文档
2	XMLCAST (value_expression AS datatype)	/	xml 值转换
3	XMLCDATA (value_expr)	/	
4	XMLCOLATTVAL(value_expr [AS { c_alias| EVALNAME value_expr }][, value_expr [AS { c_alias| EVALNAME value_expr }]]...)	/	
5	XMLCOMMENT (value_expr)	/	

6	XMLCONCAT(XMLType_instance [,XMLType_instance]...)	/	
7	XMLDIFF (XMLType_document,XMLType_document [, integer, string])	/	
8	XMLELEMENT([ENTITYESCAPING | NOENTITYESCAPING] [NAME]{ identifier| EVALNAME value_expr}[, XML_attributes_clause][, value_expr [[AS] c_alias]]...)	/	
9	XMLEXISTS (XQuery_string[XML_passing_clause])	/	
10	XMLFOREST(value_expr [AS { c_alias | EVALNAME value_expr }][, value_expr [AS { c_alias | EVALNAME value_expr }]]...)	/	
11	XMLISVALID (XMLType_instance [,XMLSchema_URL [,	/	

	element]])		
12	XMLPARSE({ DOCUMENT | CONTENT } value_expr [WELLFORMED])	/	
13	XMLPATCH (XMLType_document,XMLType_document)	/	
14	XMLPI({ [NAME] identifier| EVALNAME value_expr} [, value_expr])	/	
15	XMLQUERY(XQuery_string[XML_passing_clause]RETURNING CONTENT [NULL ON EMPTY])	/	
16	XMLROOT(value_expr, VERSION{ value_expr | NO VALUE }[, STANDALONE { YES | NO | NO VALUE }])	/	
17	XMLSEQUENCE(XMLType_instance| sys_refcursor_instance [, fmt])	/	
18	XMLSERIALIZE({ DOCUMENT | CONTENT } value_expr [AS	/	

	datatype][ENCODING xml_encoding_spec][VERSION string_literal][NO INDENT | { INDENT [SIZE =number] }][{ HIDE | SHOW } DEFAULTS])		
19	XMLTABLE([XMLnamespaces_clause ,] XQuery_string XMLTABLE_options)	/	
20	XMLTRANSFORM(XMLType_instance,{ XMLType_instance| string})	/	
21	APPENDCHILDXML(XMLType_instance, XPath_string, value_expr [, namespace_string])	/	增加节点函数
22	DELETEXML(XMLType_instance, XPath_string[, namespace_string])	/	删除节点函数
23	INSERTCHILDXML(XMLType_instance, XPath_string, child_expr, value_expr [, namespace_string])	/	插入子节点函数
24	INSERTCHILDXMLAF	/	插入后一个子节点函数

	TER(XMLType_instance, XPath_string, child_expr, value_expr [, namespace_string])		
25	INSERTCHILDXMLBEFORE(XMLType_instance, XPath_string, child_expr, value_expr [, namespace_string])	/	插入前一个节点函数
26	INSERTXMLAFTER(XMLType_instance, XPath_string, value_expr [, namespace_string])	/	插入后一个节点函数
27	INSERTXMLBEFORE(XMLType_instance, XPath_string, value_expr [, namespace_string])	/	插入前一个节点函数
28	EXISTSNODE	/	检查 XML 中的某一个节点是否存在
29	EXTRACTVALUE	/	对 XML 文件的检索功能,只能返回一个节点的一个值
杂类函数			
1	/	ISDATE(exp)	判断表达式是否为有效的日期
2	/	ISNUMERIC(exp)	判断表达式是否为有效的数值
3	ORA_HASH(expr [, max_bucket [, seed_value]])	DM_HASH (exp)	根据给定表达式生成 HASH 值
4	ITERATION_NUMBER	/	model 字句专用
5	CLUSTER_ID(<schame.	/	返回预测对

	model><mining_attribute_clause>)		mining_attribute_clause 指定预测集概率最高的簇群标识符
6	CLUSTER_PROBABILITY(<schema.model>,<cluster_id> <mining_attribute_clause>)	/	返回一个与指定模型相关联的一个输入行成员的隶属度的度量。
7	CLUSTER_SET(<schema.model>, <topN>,<cutoff><mining_attribute_clause>)	/	返回一个动态数组对象包含所有可能的集群，一个给定的行属于。
8	COMPOSE	/	将参数的字符串转化成相同字符集下的 Unicode 字符串
9	CUBE_TABLE	/	将 AW 中存储的多维数据集作为关系对象进行查询。
10	CV([dimension_column])	/	CV()用于规则的右侧，以复制左侧指定的当前维度值。
11	DATAOBJ_TO_PARTITION	/	
12	BIN_TO_NUM(ch)	/	转换位向量为一个数字[ch 为逗号隔开的 0 或 1]
13	CARDINALITY (nested_table)	/	返回内嵌表中元素的个数
14	MAKE_REF(obj_name,key)	/	生成一个引用
15	DEREF(expr)	/	返回引用指向的对象
16	REF (correlation_variable)	/	返回对象引用
17	REFTOHEX (expr)	/	将引用转成 16 进制

18	VALUE(correlation_variable)	/	返回关联变量的值
19	NANVL(n2,n1)	/	只适用于数据类型 BINARY_FLOAT 和 BINARY_DOUBLE。若 n2 为 null，则返回 n1，else 返回 n2
20	PATH(correlation_integer)	/	xml 模式下使用，配合 EQUALS_PATHand UNDER_PATH 使用
21	DEPATH(correlation_integer)	/	
22	FEATURE_ID(<schame.model><mining_attribute_clause>)	/	返回一个具有最高值系数值的特征的标识号
23	FEATURE_SET(<schema.model>, <topN>,<cutoff><mining_attribute_clause>)	/	返回一个包含所有可能的功能对象的动态数组。在动态数组的每个对象是一对标量值包含特征标识和特征值。
24	FEATURE_VALUE(<schema.model>,<feature_id><mining_attribute_clause>)	/	返回给定的特征值。
25	PREDICTION ([schema .] model [cost_matrix_clause]mining_attribute_clause)	/	返回模型的最佳预测。
26	PREDICTION_COST(<schema.model>,<class><cost_matrix_clause><mining_attribute_clause>)	/	返回一个给定的预测值作为一个给定的预测数

27	PREDICTION_DETAILS(<schema.model><mining_attribute_clause>)	/	返回一个包含模型相关信息的字符串，该字符串包含输入行的评分
28	PREDICTION_PROBABILITY(<schema.model><class><mining_attribute_clause>)	/	返回给定预测的概率作为一个甲骨文号
29	PREDICTION_SET(<schema.model>, <best N>, <cutoff><cost_matrix_clause><mining_attribute_clause>)	/	返回一个动态数组在多类分类方案包含所有类的对象
30	PRESENTNNV(cell_reference, expr1, expr2)	/	模型函数
31	PRESENTV(cell_reference, expr1, expr2)	/	模型函数
32	PREVIOUS(cell_reference)	/	模型函数
33	ROWIDTONCHAR(rowid)	/	将 INT 类型的 ROWID 值转换成 NVARCHAR2 类型
34	SET (nested_table)	/	过滤嵌套表的重复记录
35	COLLECT(distinct|unique,column,orderby expr)	/	将结果转换为一个嵌套表
36	POWERMULTISET(expr)	/	包装嵌套表
37	POWERMULTISET_BY_CARDINALITY(expr,cardinality)	/	包装嵌套表
REGEXP 函数(正则)			

1	/	REGEXP_LIKE(str, pattern [, match_param])	根据 pattern 正则表达式，查找 str 字符串是否存在符合正则表达式的子串，并符合匹配参数 match_param
2	{ REGR_SLOPE| REGR_INTERCEPT| REGR_COUNT| REGR_R2| REGR_AVGX| REGR_AVGY| REGR_SXX| REGR_SYY| REGR_SXY}(expr1 , expr2)[OVER (analytic_clause)]	/	线性回归
环境函数			
1	USERENV('parameter')	USERENV	oracle：同 SYS_CONTEXT,为向后兼容；DM：作为 sys_context 的参数

重点比对 dm 和 oracle 有区别的函数如 wmcancat

1.2.2 无区别的函数

序号	ORACLE 函数名	DM 函数名	功能简要说明
数值函数			
1	ABS(n)	ABS(n)	求数值 n 的绝对值
2	ACOS(n)	ACOS(n)	求数值 n 的反余弦值
3	ASIN(n)	ASIN(n)	求数值 n 的反正弦值
4	ATAN(n)	ATAN(n)	求数值 n 的反正切值
5	ATAN2(n1,n2)	ATAN2(n1,n2)	求数值 n1/n2 的反正切值

6	CEIL(n)	CEIL(n) 或 CEILING(n)	求大于或等于数值 n 的最小整数
7	COS(n)	COS(n)	求数值 n 的余弦值
8	COSH(n)	COSH(n)	求数值 n 的双曲余弦值
9	COT(n)	COT(n)	求数值 n 的余切值
10	EXP(n)	EXP(n)	求 e 的 n 次幂值，e=2.71828183...
11	FLOOR(n)	FLOOR(n)	求小于或等于数值 n 的最大整数
12	GREATEST(n1,n2,n3)	GREATEST(n1,n2,n3)	求 n1 、n2 和 n3 中的最大浮点数
13	LEAST(n1,n2,n3)	LEAST(n1,n2,n3)	求 n1 、n2 和 n3 中的最小浮点数
14	LN(n)	LN(n)	求数值 n 的自然对数
15	LOG(n1,n2)	LOG(n1,n2)	求数值 n2 以 n1 为底数的对数
16	MOD(m,n)	MOD(m,n)	求数值 m 被数值 n 除的余数
17	PI()	PI()	得到常数π
18	POWER(n1,n2)	POWER(n1,n2)	求数值 n2 以 n1 为基数的指数
19	RAND([n])	RAND([n])	求一个 0 到 1 之间的随机浮点数
20	ROUND(n[,m])	ROUND(n[,m])	求四舍五入值函数
21	SIGN(n)	SIGN(n)	判断数值的数学符号
22	SIN(n)	SIN(n)	求数值 n 的正弦值
23	SINH(n)	SINH(n)	求数值 n 的双曲正弦值
24	SQRT(n)	SQRT(n)	求数值 n 的平方根
25	TAN(n)	TAN(n)	求数值 n 的正切值
26	TANH(n)	TANH(n)	求数值 n 的双曲正切值
27	TO_NUMBER (char [,fmt])	TO_NUMBER (char [,fmt])	将 CHAR 、VARCHAR、VARCHAR2 等类型的字符串转

			换为 DECIMAL 类型的数值
28	TRUNC(n[,m])	TRUNC(n[,m])	截取数值函数
29	TRUNCATE(n[,m])	TRUNCATE(n[,m])	截取数值函数，等价于 TRUNC
30	TO_CHAR(n [, fmt [,'nlsparam']])	TO_CHAR(n [, fmt [,'nlsparam']])	将数字类型的数据 转 换 为 VARCHAR 类型输出
31	BITAND(n1, n2)	BITAND(n1, n2)	求两个数值型数值按位进行 AND 运算的结果
字符串函数			
1	ASCII(char)	ASCII(char)	返回字符对应的整数
2	ASCIISTR(char)	ASCIISTR(char)	将字符串 char 中， 非 ASCII 的字符转成\XXXX(UTF-16)格式，ASCII 字符保持不变
3	BIT_LENGTH(char)	BIT_LENGTH(char)	求字符串的位长度
4	CHAR(n)或 CHR(n)	CHAR(n)或 CHR(n)	返回整数 n 对应的字符
5	CHAR_LENGTH(char)	CHAR_LENGTH(char)	求字符串的串长度
6	CHARACTER_LENGTH(char)	CHARACTER_LENGTH(char)	求字符串的串长度
7	CONCAT(char1,char2)	CONCAT(char1,char2)	顺序联结 2 个字符串成为一个字符串
8	DIFFERENCE(char1,char2)	DIFFERENCE(char1,char2)	以整数返回两个字 符 串 的 SOUNDEX 值 之差
9	INITCAP(char)	INITCAP(char)	将字符串中单词的首字符转换成大写的字符
10	INSTR(char1,char2,n,m)	INSTR(char1,char2,n,m)	从输入字符串 char1 的第 n 个字符开始查找字符串 char2 的第 m 次的出现，以字符计算
11	INSTRB(char1,char2,n,m)	INSTRB(char1,char2,n,m)	从输入 字符串 char1 的第 n 个字符开始查找字符串 char2

			的 第 m 次的出现，以字节计算
12	LCASE(char)	LCASE(char)	将大写的字符串转换为小写的字符串
13	LENGTH(char)	LEN(char) 或 LENGTH(char)	返回给定字符串表达式的字符(而不是字节)个数（汉字为一个字符）， 其中不包含尾随空格
14	LENGTHB(char)	LENGTHB(char)	返回输入字符串的字节数
15	LOCATE(char1,char2)	LOCATE(char1,char2)	返 回 char1 在 char2 中首次出现的位置
16	LOWER(char)	LOWER(char)	将大写的字符串转换为小写的字符串
17	LOWER(char)	LOWER(char)	在输入字符串的左边填充上 char2 指定的字符，将其拉伸至 n 个字符长
18	LTRIM(char1,char2)	LTRIM(char1,char2)	从输入字符串中删除所有的前导字符，这些前导字符由 char2 来定义
19	REPEAT(char, n)	REPEAT(char, n) / REPEATSTR(char, n)	返回将字符串重复 n 次形成的字符串
20	REPLACE (arg1,arg2,arg3)	REPLACE (arg1,arg2,arg3)	将在 arg1 中的所有 arg2 替 换成 arg3
21	REVERSE(char)	REVERSE(char)	将字符串反序
22	RPAD(char1,n,char2)	RPAD(char1,n,char2)	类似 LPAD 函数， 只是向右拉伸该字符串使之达到 n 个字符串长
23	RTRIM(char1,char2)	RTRIM(char1,char2)	从输入字符串的右 端 开 始删 除 char2 参数中的字符
24	SOUNDEX(char)	SOUNDEX(char)	返回一个表示字符串发音的字符串

25	STUFF(char1,begin,n,char 2)	STUFF(char1,begin,n,char2)	删除在字符串 char1 中以 begin 参数所指位置开始的 n 个字符，再把 char2 插入到 char1 串的 begin 所指位置
26	SUBSTR(char,m,n)	SUBSTR(char,m,n)	返回 char 中从字符位置 m 开始的 n 个字符
27	SUBSTRB(char,n,m)	SUBSTRB(char,n,m)	SUBSTR 函数等价的单字节形式
28	TO_CHAR(DATE[,fmt])	TO_CHAR(DATE[,fmt])	日期转换为字符串函数
29	TRANSLATE(S,S1,S2)	TRANSLATE(S,S1,S2)	将所有出现在搜索字符集中的字符转换成字符集中的相应字符
30	TRIM([LEADING|TRAILIN G|BOTH] [exp] [] FROMchar2])	TRIM([LEADING|TRAILING| BOTH] [exp] [] FROMchar2])	删去字符串 char2 中由串 char1 指定的字符
31	UPPER(char)	UCASE(char)	将小写的字符串转换为大写的字符串
32	UPPER(char)	UPPER(char)	将小写的字符串转换为大写的字符串
33	GREATEST(char 1, char 2,char 3)	GREATEST(char 1, char 2,char 3)	求 char 1、char 2 和 char 3 中最大的字符串
34	to_single_byte (char)	to_single_byte (char)	将多字节形式的 字符（串）转换为对应的单字节形式
35	TO_MULTI_BYTE(char)	to_multi_byte (char)	将单字节形式的字符（串）转换为对应的多字节形式
36	EMPTY_CLOB ()	EMPTY_CLOB ()	初始化 clob 字段
37	EMPTY_BLOB ()	EMPTY_BLOB ()	初始化 blob 字段
日期类函数			
1	ADD_DAYS(date,n)	ADD_DAYS(date,n)	返回日期加上 n 天后的新日期
2	ADD_MONTHS(date,n)	ADD_MONTHS(date	在输入日期上加上指定的几个月

		,n)	返回一个新日期
3	ADD_WEEKS(date,n)	ADD_WEEKS(date,n)	返回日期加上 n 个星期后的新日期
4	CURRENT_DATE()	CURRENT_DATE()	返回系统当前日期
5	CURRENT_TIMESTAMP(n)	CURRENT_TIMESTAMP(n)	返回系统当前带时区信息的时间戳
6	EXTRACT(时间字段 FROMdate)	EXTRACT(时间字段 FROMdate)	抽取日期时间或时间间隔类型中某一个字段的值
7	GREATEST(n1,n2,n3)	GREATEST(n1,n2,n3)	求 n1 、n2 和 n3 中的最大日期
8	LAST_DAY(date)	LAST_DAY(date)	返回输入日期所在月份最后一天的日期
9	LEAST(n1,n2,n3)	LEAST(n1,n2,n3)	求 n1 、n2 和 n3 中的最小日期
10	MONTHS_BETWEEN(date1,date2)	MONTHS_BETWEEN(date1,date2)	返回两个日期之间的月份数
11	NEXT_DAY(date1,char2)	NEXT_DAY(date1,char2)	返回输入日期指定若干天后的日期
12	ROUND (date1,char2)	ROUND (date1,char2)	把日期四舍五入到最接近格式元素指定的形式
13	SYSDATE()	SYSDATE()	返回系统的当前日期
14	TO_DATE(CHAR[,fmt])	TO_DATE(CHAR[,fmt])	字符串转换为日期数据类型
15	TRUNC(date[,format])	TRUNC(date[,format])	把日期截断到最接近格式元素指定的形式
16	LOCALTIMESTAMP()	LOCALTIMESTAMP()	返回系统当前时间戳
17	OVERLAPS	OVERLAPS	返回两两时间段是否存在重叠
18	NUMTODSINTERVAL(dec,interval_unit)	NUMTODSINTERVAL(dec,interval_unit)	转换一个指定的 DEC 类型到 INTERVAL DAY TO SECOND

19	NUMTOYMINTERVAL (dec,interval_unit)	NUMTOYMINTERVAL(dec,interval_unit)	转换一个指定的 DEC 类型值到 INTERVAL YEAR TO MONTH
20	FROM_TZ(timestamp,tim ezone|tz_name])	FROM_TZ(timestamp ,time zone|tz_name])	将时间戳类型 timestamp 和时区类型 timezone（或时区名称 tz_name）转化为 timestamp withtimezone 类型
21	TO_CHAR(DATE[,fmt])	TO_CHAR(DATE[,fmt])	将日期数据类型 DATE 转换为一个在日期语法 fmt 中指定语法的 VARCHAR 类型字符串
空值判断函数			
1	COALESCE(n1,n2,…nx)	COALESCE(n1,n2,…nx)	返回第一个非空的值
2	IFNULL(n1,n2)	IFNULL(n1,n2)	当 n1 为非空时，返回 n1；若 n1 为空，则返回 n2
3	NULLIF(n1,n2)	NULLIF(n1,n2)	如果 n1=n2 返回 NULL，否则返回 n1
4	NVL(n1,n2)	NVL(n1,n2)	返回第一个非空的值
类型转换函数			
1	CAST(value AS 类型说明)	CAST(value AS 类型说明)	将 value 转换为指定的类型
2	CONVERT(类型说明,value)	CONVERT(类型说明,value)	将 value 转换为指定的类型
3	HEXTORAW(string)	HEXTORAW(string)	将由 string 表示的二进制数值转换为一个 RAW 数值
4	RAWTOHEX(rawvalue)	RAWTOHEX(rawvalue)	将 RAW 类数值 rawvalue 转换为一个相应的十六进制表示的字符串
集函数			

1	COUNT	COUNT	对组内发生的事情进行累计
2	Avg	Avg	计算一个组和数据窗口内表达式的平均值；
3	MAX	MAX	在一个组中的数据窗口中查找表达式的最大值
4	MIN	MIN	在一个组中的数据窗口中查找表达式的最小值
5	SUM	SUM	该函数计算组中表达式的累积和
6	DISTINCT	DISTINCT	去重函数
7	covar_pop	covar_pop	协方差函数
8	covar_samp	covar_samp	协方差函数
9	CORR	CORR	协方差函数
10	var_pop	var_pop	方差函数
11	var_samp	var_samp	方差函数
12	VARIANCE	VARIANCE	该函数返回表达式的方差
13	stddev_pop	stddev_pop	方差函数
14	STDDEV	STDDEV	计算当前行关于组的标准偏离
15	STDDEV_SAMP	STDDEV_SAMP	该函数计算累积样本标准偏离，并返回总体变量的平方根，其返回值与 VAR_POP 函数的平方根相同
16	FIRST/LAST	FIRST/LAST	AVG|MAX|MIN| COUNT|SUM([ALL] < 值表达式 >) KEEP (DENSE_RANKFIRST|LAST ORDER BY 子句)；
17	LISTAGG	LISTAGG	字符串集函数
分析函数			
1	COUNT(<distinct> <*>)	COUNT(<distinct>	计数

		<*>)	
2	Avg	Avg	完全分析函数
3	MAX	MAX	完全分析函数
4	MIN	MIN	完全分析函数
5	COUNT|SUM([ALL]< 值表达式>)OVER	COUNT|SUM([ALL]<值表达式>) OVER	完全分析函数
6	covar_pop	covar_pop	协方差函数
7	covar_samp	covar_samp	协方差函数
8	CORR	CORR	协方差函数,相关系数
9	PERCENT_RANK	PERCENT_RANK	百分比函数
10	PERCENTILE_CONT	PERCENTILE_CONT	连续百分比
11	PERCENTILE_DISC	PERCENTILE_DISC	分布百分比
12	CUME_DIST	CUME_DIST	百分比函数
13	RATIO_TO_REPORT	RATIO_TO_REPORT	百分比函数
14	DENSE_RANK	DENSE_RANK	排序函数
15	RANK	RANK	排序函数
16	ROW_NUMBER	ROW_NUMBER	排序函数
17	FIRST_VALUE	FIRST_VALUE	首尾函数
18	LAST_VALUE	LAST_VALUE	首尾函数
19	LAG	LAG	相邻函数
20	LEAD	LEAD	相邻函数
21	NTILE	NTILE	分组函数
22	BFILENAME('directory','filename')	BFILENAME('directory','filename')	生成一个 BFILE 类型数据
23	DUMP(expr[, return_fmt[, start_position [,	DUMP(expr[, return_fmt [, start_position [,	获得表达式的内部存储字节

	length]]])	length]]])	
杂类函数			
1	DECODE(exp, search1, result1, … searchn, resultn [,default])	DECODE(exp, search1, result1, … searchn, resultn [,default])	decode 函数比较表达式和搜索字，如果匹配，返回结果；如果不匹配，返回 default 值；如果未定义 default 值，则返回空值。
2	ROWIDTOCHAR(rowid)	ROWIDTOCHAR(rowid)	将 INT 类型的 ROWID 值转换成 CHAR 类型
3	VSIZE(expr)	VSIZE(expr)	返回 n 的核心内部表示的字节数。
层次查询			
1	SYS_CONNECT_BY_PATH(col_name,char)	SYS_CONNECT_BY_PATH(col_name,char)	该函数得到从根节点到当前节点路径上所有节点名为 col_name 的某列的值，之间用 char 指明的字符分隔开。
group by 字句相关函数			
1	GROUP_ID()	GROUP_ID()	表示结果集来自于哪一个分组，用于区别相同分组的结果集
2	GROUPING(expr)	GROUPING(expr)	用来标识某列是否为分组列
3	GROUPING_ID(expr)	GROUPING_ID(expr)	表示参数列是否为分组列
REGEXP 函数(正则)			
1	REGEXP_COUNT(str,pattern[, position [, match_param]])	REGEXP_COUNT(str,pattern[, position [, match_param]])	根据 pattern 正则表达式，从 str 字符串的第 position 个字符开始查找符合正则表达式的子串的个数，并符合匹配参数 match_param
2	REGEXP_INSTR(str,pattern[, position[,occurrence	REGEXP_INSTR(str, pattern[,	根据 pattern 正则表达式，从 str 字符串的第 position 个字符开始

	[, return_opt [,match_param [,subexpr]]]]])	position[,occurrence [, return_opt [,match_param [,subexpr]]]]])	查找符合 subexpr 正则表达式的子串，如果 return_opt 为 0，返回第 occurrence 次出现的位置，如果 return_opt 为大于 0，则返回该出现位置的下一个字符位置，并符合匹配参数。Subexpr 为匹配的子 pattern。
3	REGEXP_SUBSTR(str,pattern [,position [,occurrence[,match_param[,subexpr]]]])	REGEXP_SUBSTR(str,pattern [,position [,occurrence[,match_param[,subexpr]]]])	根据 pattern 正则表达式，从 str 字符串的第 position 个字符开始查找符合 subexpr 正则表达式的子串，返回第 occurrence 次出现的子串，并符合匹配参数 match_param。
4	REGEXP_REPLACE(str, pattern [, replace_str [,position [, occurrence[,match_param]]]])	REGEXP_REPLACE (str,pattern [, replace_str [,position [, occurrence[,match_param]]]])	根据 pattern 正则表达式，从 str 字符串的第 position 个字符开始查找符合正则表达式的子串，并用 replace_str 进行替换第 occurrence 次出现的子串，并符合匹配参数 match_param。
环境函数			
1	SYS_CONTEXT('namespace', 'parameter' [,length])	SYS_CONTEXT('namespace', 'parameter' [, length])	返回参数相关的上下文

其他区别较小或去区别的数据类型

1.3 PL/SQL 与 DM-SQL

1.3.1 有区别的语法

重点对比 dm 和 oracle 有区别的语法

将 oracle 下的存储过程、函数、包、移植到 DM。详细情况如下：

```
select * from dba_source where owner='JQ_FSSC'
```

数据类型转换。

```
--ORACLE 的 RAW 类型替换为 DM 的 VARBINARY
--ORACLE 的 NVARCHAR2 替换为 DM 的 VARCHAR
```

改写不兼容的函数

```
-- ORACLE 函数 NUMTODSINTERVAL 修改如下：
NUMTODSINTERVAL(N,'day')修改为 n
NUMTODSINTERVAL(N,'hour') 修改为 n/24*1.0
NUMTODSINTERVAL(N,'minute')修改为 n/1440*1.0
NUMTODSINTERVAL(N,'second')修改为 n/1440/60*1.0
```

1.3.2 无区别的语法

无区别的语法

项目	ORACLE	DM
语句块	相同	相同
赋值语句	相同	相同
条件语句	相同	相同
存储过程、包、函数	相同	相同
循环语句	while/for loop	相同

1.4 其他

其他上面三个没有包含的部分，比如大小写敏感问题，并发控制等

大小写敏感移植

并发控制主要是锁机制上区别：

Oracle 可以在数据库、表、行这 3 个级别上使用锁，锁定数据库有 2 种方法:将数据库设置成受限方式、将数据库更改成只读方式。

DM 可以在事务 TID 和对象这 2.个级别上使用锁，其中对象锁分为数据字典

锁和表锁。只有多个事务同时修改同一行记录时，才会产生新的 TID 锁。而对象锁将对数据字典的封锁和表锁合并为对象锁，以达到减少封锁冲突、提升系统并发性能的目的。