# 迁移步骤过程

# 第一部分查看数据基本信息

## 一、查看原数据库信息

### 1.查原库oracle对象

select object\_type,count(\*) from dba\_objects where owner='INFODBA' group by object\_type;

1 SEQUENCE 318

2 PROCEDURE 3

3 PACKAGE 1

4 PACKAGE BODY 1

5 LOB 209

6 TRIGGER 100

7 TABLE 4001

8 INDEX 7150

9 VIEW 284

10 FUNCTION 9

11 JOB 2

### 2.查看原库每张表的大小和记录

select table\_name,round((num\_rows \* avg\_row\_len) / 1024 /1024,2) AS table\_size\_mb,num\_rows from all\_tables where owner = 'INFODBA' and num\_rows is not null order by 2 desc;



### 3.查看表空间和使用率

select total.tablespace\_name,

round(total.MB, 2) as Total\_MB,

round(total.MB - free.MB, 2) as Used\_MB,

round((1 - free.MB / total.MB) \* 100, 2) || '%' as Used\_Pct

from (select tablespace\_name, sum(bytes) / 1024 / 1024 as MB

from dba\_free\_space

group by tablespace\_name) free,

(select tablespace\_name, sum(bytes) / 1024 / 1024 as MB

from dba\_data\_files

group by tablespace\_name) total

where free.tablespace\_name = total.tablespace\_name

order by used\_pct desc;



### 4.查看oracle表、表大小、表总大小、行数

SELECT table\_name,

ROUND((blocks \* 8 / 1024), 2) AS table\_size\_mb,

ROUND((blocks \* 8 / 1024) + (num\_rows \* avg\_row\_len / 1024 / 1024), 2) AS total\_size\_mb,

num\_rows

FROM user\_tables;

# 第二部分数据准备

### 1.新建用户

su - postgres -c "psql -c \"

CREATE USER infodba SUPERUSER PASSWORD 'Siemens@Tcm123';

CREATE USER replica PASSWORD '123456' REPLICATION;

CREATE USER pgrewind SUPERUSER PASSWORD '123456';

create user TcClusterDB password 'tcclusterdb';

\" "

### 2.新建表空间

-- infodba\_idate /data01/pgsql/tc/infodba\_idata

-- infodba\_ilog /data01/pgsql/tc/infodba\_ilog

-- infodba\_indx /data01/pgsql/tc/infodba\_indx

tcclusterdb\_idata /data01/pgsql/tc/tcclusterdb\_idata

--CREATE TABLESPACE infodba\_idate LOCATION '/data01/pgsql/tc/infodba\_idata';

--CREATE TABLESPACE infodba\_ilog LOCATION '/data01/pgsql/tc/infodba\_ilog';

--CREATE TABLESPACE infodba\_indx LOCATION '/data01/pgsql/tc/infodba\_indx';

CREATE TABLESPACE tcclusterdb\_idata LOCATION '/data01/pgsql/tc/tcclusterdb\_idata';

### 3.新建库

su - postgres -c "psql -c \"CREATE DATABASE tc WITH owner infodba encoding 'UTF8' template template0 LC\_COLLATE='C' ;\""

create tablespace TcClusterDB\_idata owner TcClusterDB location '/data01/pgsql/tc/tcclusterdb\_idata';

create database TcClusterDB with owner TcClusterDB encoding 'UTF8' template template0 lc\_collate 'C' tablespace TcClusterDB\_idata;

grant all privileges on database TcClusterDB to TcClusterDB;

grant CREATE ON TABLESPACE TcClusterDB\_idata to TcClusterDB;

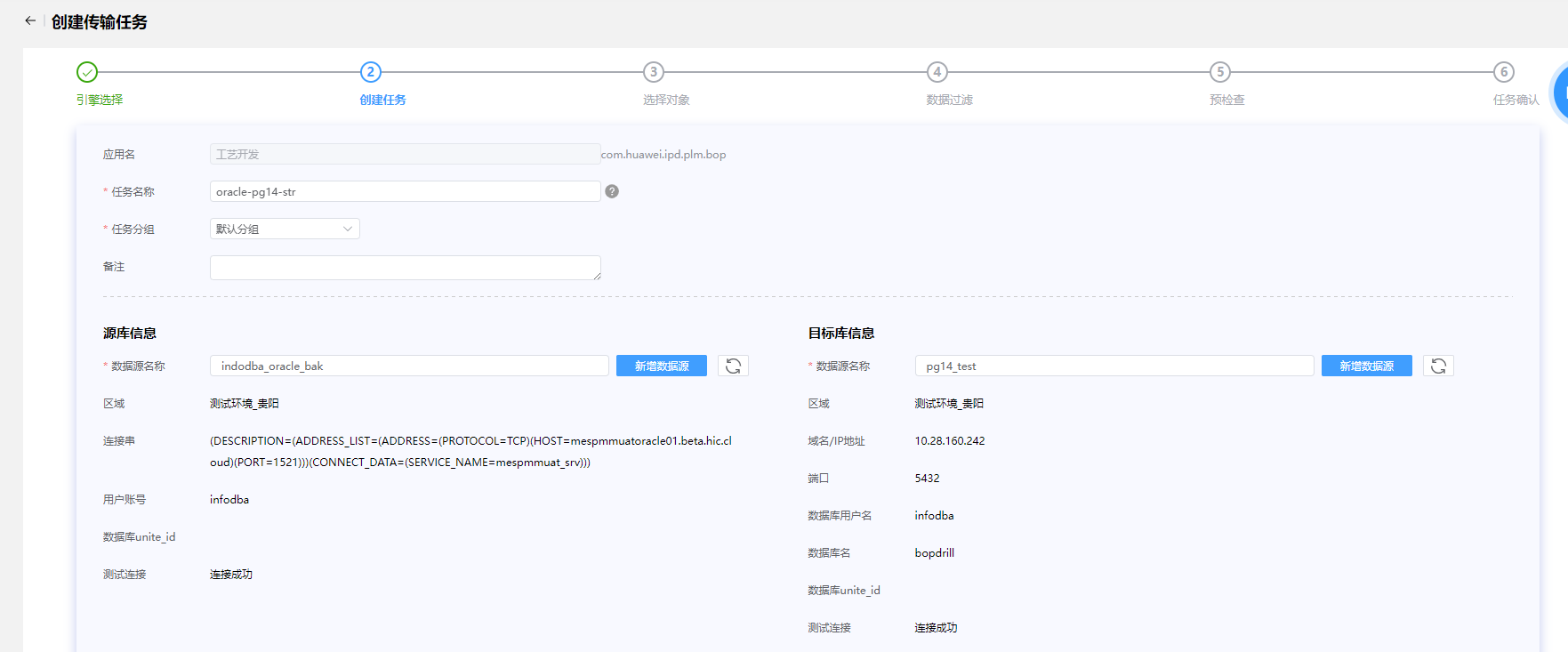
### 4.--新建schema

--su - postgres -c "psql -d bopdrill -c \"CREATE SCHEMA infodba;\" "

# 第三部分 oracle数据库数据、索引迁移到PG

# 使用flashsync导表结构

## 创建传输任务

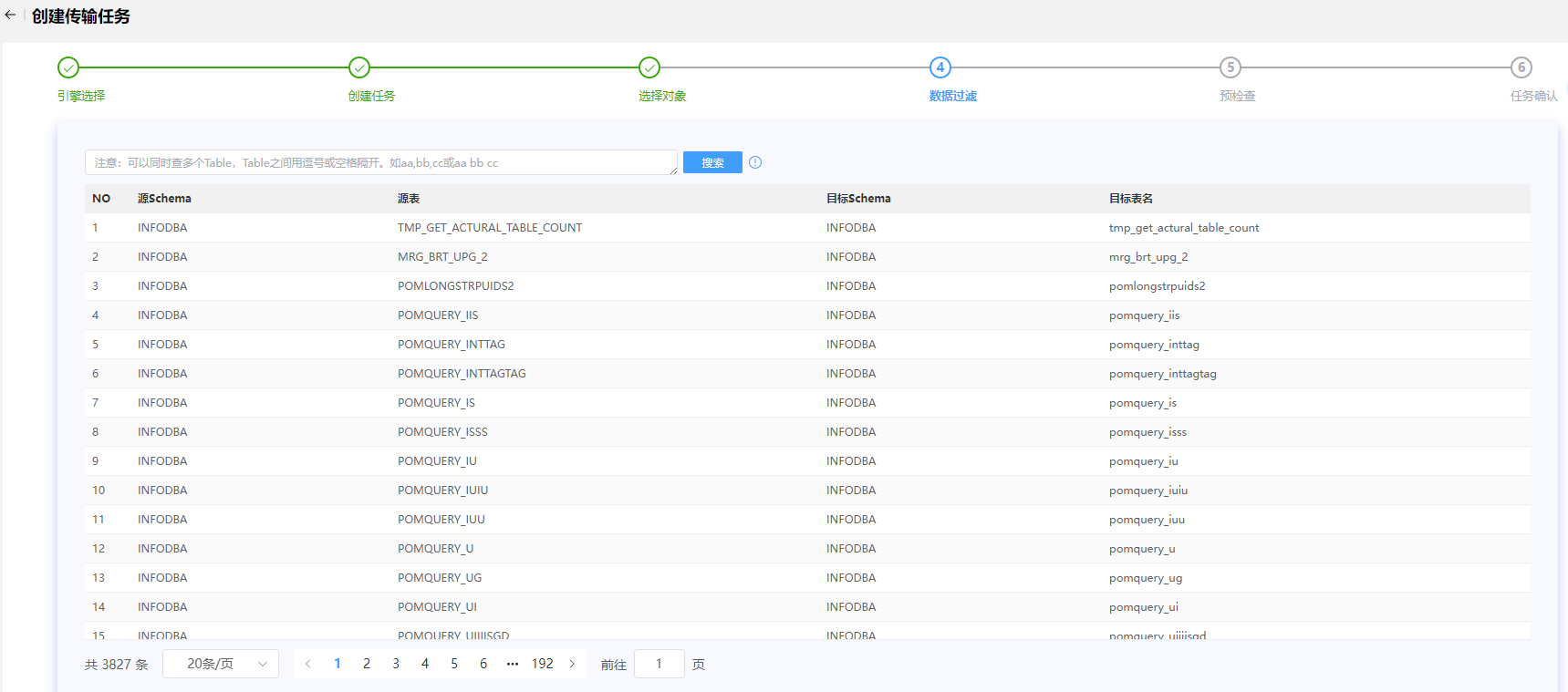


## 添加schema列表或者Table列表（数据搬迁）

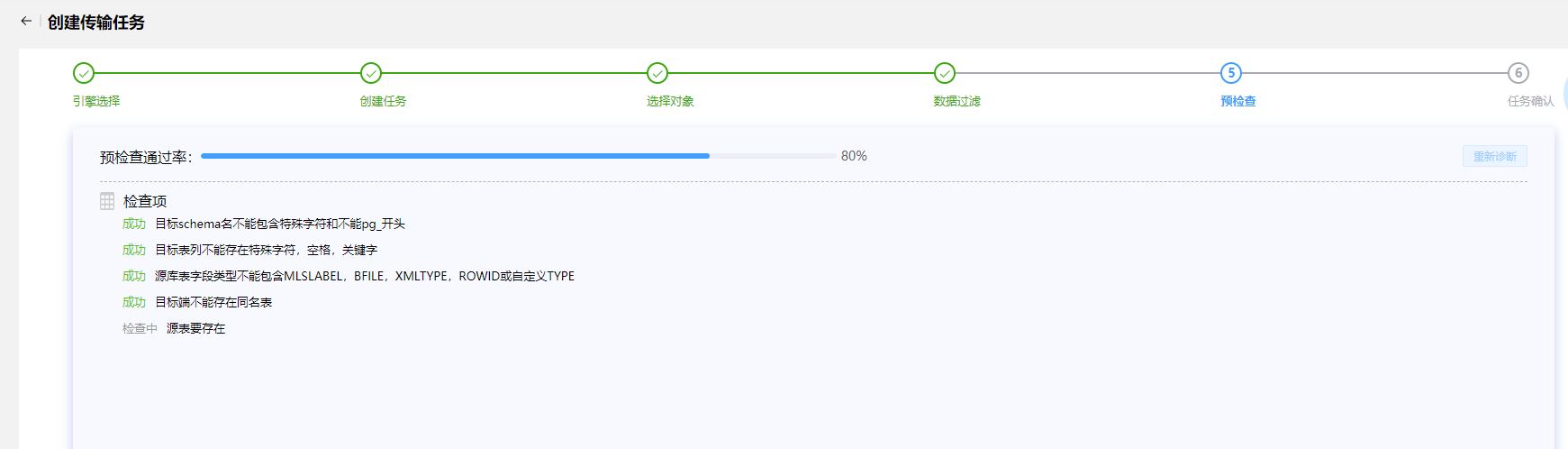


排除表（MMV\_SPATIAL\_CELL\_INDEX、TEMP\_SESSION、TEMP\_SESSION\_0618、TEMP\_SESSION\_0619\_1）

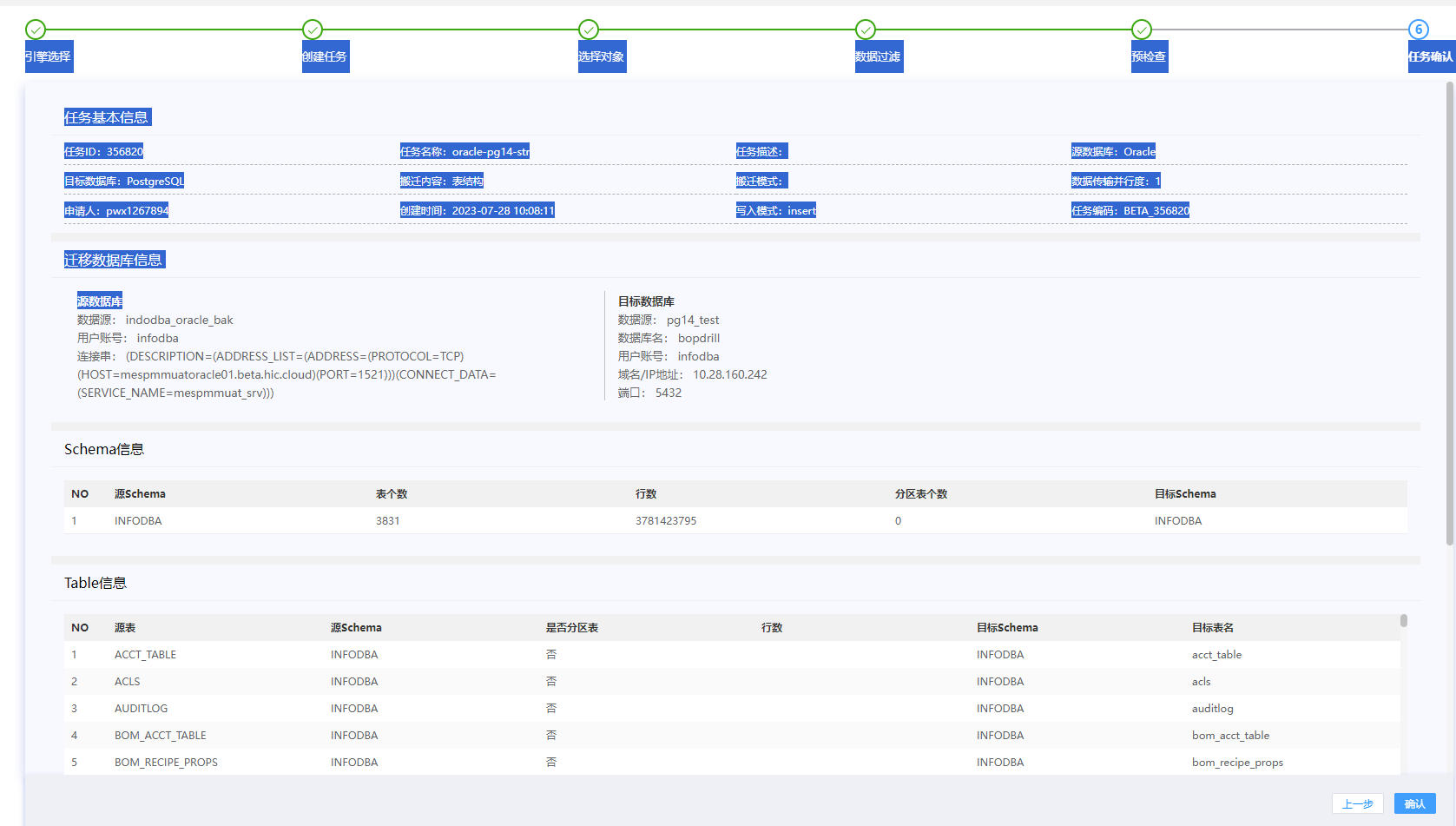
## 数据过滤



## 预检查



## 任务确认



时长：26分

## 新建排除4张的表结构

create table MMV\_SPATIAL\_CELL\_INDEX

(

puid VARCHAR(15) not null,

spatial\_rep VARCHAR(15) not null,

cell\_id INTEGER not null,

occ\_path\_prefix VARCHAR(900) not null,

occ\_path\_suffix VARCHAR(4000),

pxmin numeric,

ymin numeric,

zmin numeric,

pxmax numeric,

ymax numeric,

zmax numeric

);

create unique index MMVSPATIALCELLINDEX\_PUID\_PK on MMV\_SPATIAL\_CELL\_INDEX (PUID);

create index MMV\_SPATIAL\_CELLID1 on MMV\_SPATIAL\_CELL\_INDEX (CELL\_ID);

create index MMV\_SPATIAL\_OCCPATHPREFIX1 on MMV\_SPATIAL\_CELL\_INDEX (OCC\_PATH\_PREFIX);

create index MMV\_SPATIAL\_SPATIALREP1 on MMV\_SPATIAL\_CELL\_INDEX (SPATIAL\_REP);

create table TEMP\_SESSION

(

id numeric,

exetime DATE,

sid numeric,

"serial#" numeric,

process VARCHAR(24),

machine VARCHAR(64),

program VARCHAR(48),

osuser VARCHAR(30),

status VARCHAR(8),

prev\_sql\_id VARCHAR(24),

sql\_id VARCHAR(24)

);

create table TEMP\_SESSION\_0618

(

id numeric,

exetime DATE,

sid numeric,

"serial#" numeric,

process VARCHAR(24),

machine VARCHAR(64),

program VARCHAR(48),

osuser VARCHAR(30),

status VARCHAR(8)

);

create table TEMP\_SESSION\_0619\_1

(

id numeric,

exetime DATE,

sid numeric,

"serial#" numeric,

process VARCHAR(24),

machine VARCHAR(64),

program VARCHAR(48),

osuser VARCHAR(30),

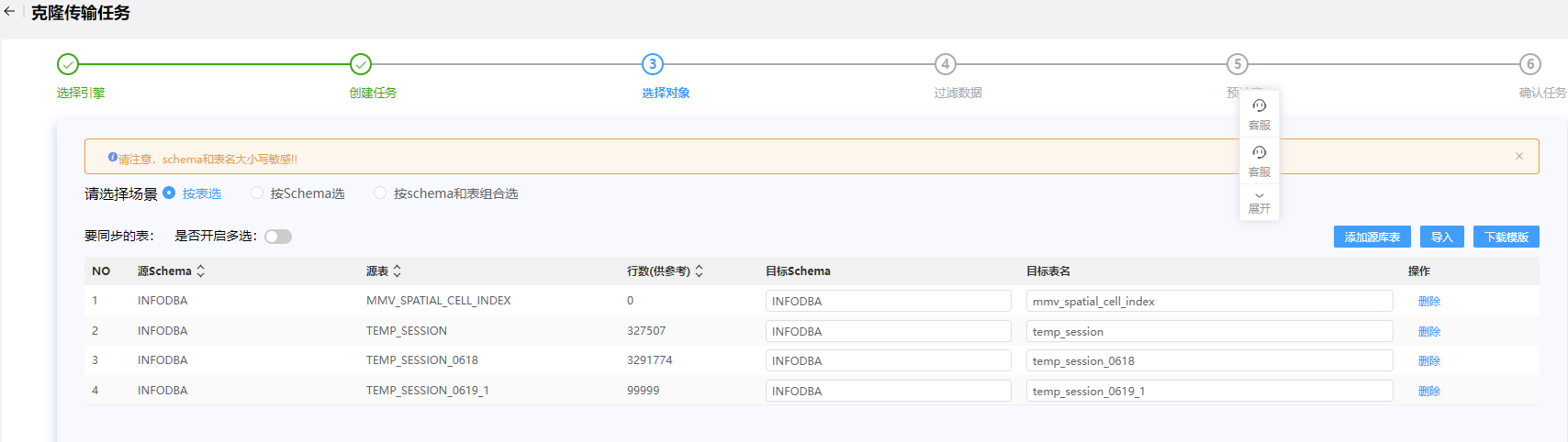
status VARCHAR(8),

prev\_sql\_id VARCHAR(24),

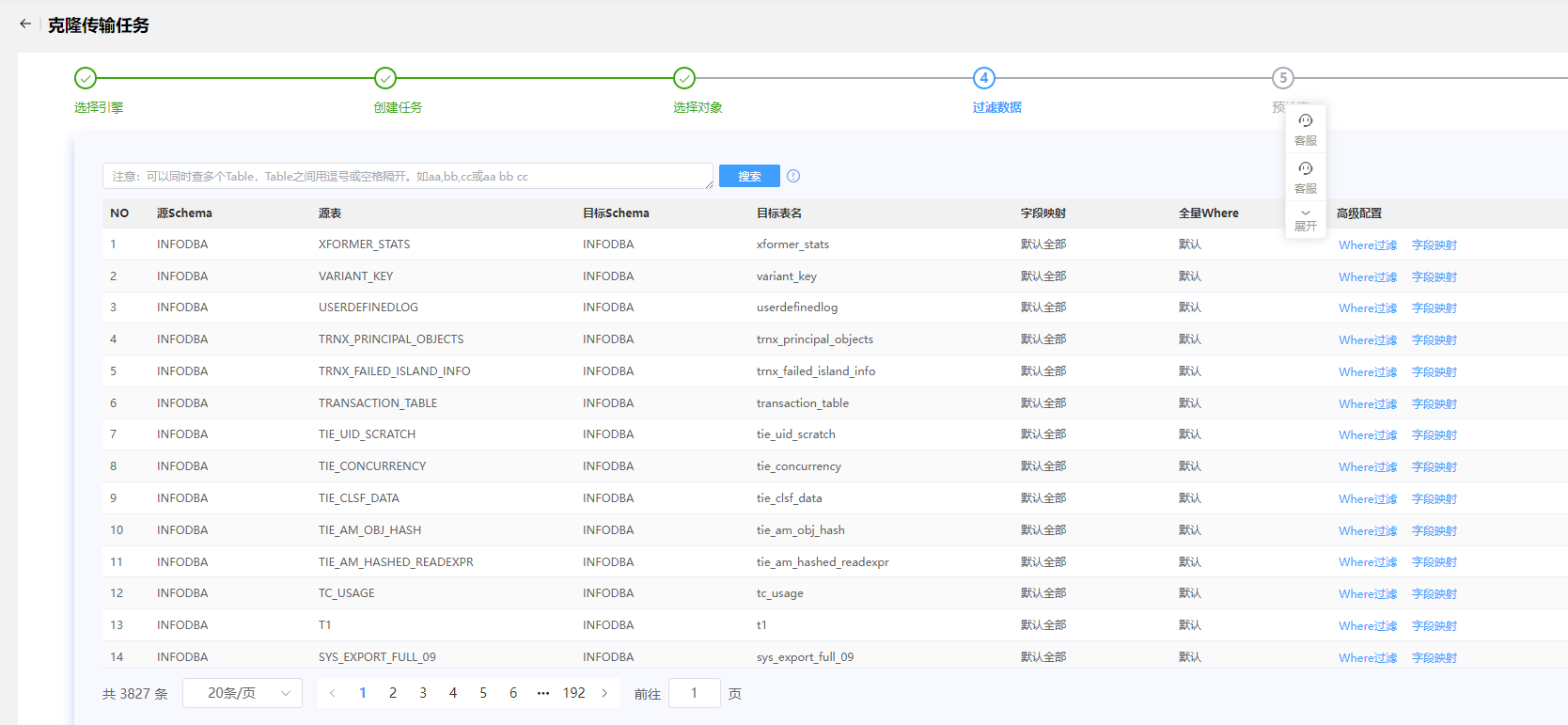
sql\_id VARCHAR(24)

);

## 导入四张表的数据，MMV\_SPATIAL\_CELL\_INDEX、TEMP\_SESSION、TEMP\_SESSION\_0618、TEMP\_SESSION\_0619\_1 （flashsync)



## 导入所有表的数据除MMV\_SPATIAL\_CELL\_INDEX、TEMP\_SESSION、TEMP\_SESSION\_0618、TEMP\_SESSION\_0619\_1 （flashsync)。

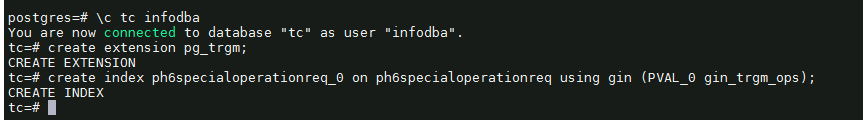


索引重建报错：rebuild index piph6specialoperationreq\_0 error,error:index row size 2808 exceeds btree version 4 maximum 2704 for index "piph6specialoperationreq\_0" DETAIL: Index row references tuple (8386,36) in relation "ph6specialoperationreq". HINT: Values larger than 1/3 of a buffer page cannot be indexed. Consider a function index of an MD5 hash of the value, or use full text indexing.

解决方案：

create extension pg\_trgm;

create index ph6specialoperationreq\_0 on ph6specialoperationreq using gin (PVAL\_0 gin\_trgm\_ops);



安装插件create extension pg\_trgm;报错信息是extenction文件目录中缺少control文件。应该是编译信息没有遍全

( tar -zxvf postgresql-14.8.tar.gz -C /app/install

cd /app/install/postgresql-14.8

./configure –prefix=/app/postgresql

make world && make install-world)

再然后在flashsync,目标表索引重建详情 选择状态 “已失败”，点击“重试”



# 对比表数据

Oracle:

select count(\*) from user\_tables; 4001

Pg14:

SELECT COUNT(\*) AS table\_count

FROM information\_schema.tables

WHERE table\_schema = 'public' and table\_type = 'BASE TABLE '; 4001

# 视图数据

Oracle:

Select count(\*) from user\_views; 284

Pg14:

SELECT table\_name

FROM information\_schema.views

WHERE table\_schema = current\_schema; 0

# 索引数据

oracle:

select index\_name as indexname from user\_indexes i where i.index\_name not like 'SYS\_%'; 6914

pg:

select upper(indexname) from pg\_indexes where schemaname = 'public' and indexname not like 'sys\_%'; 6924

oracle导出xls表，然后导入PG tc库里表名为t1比较

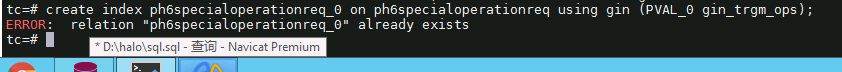


Select \* from t1

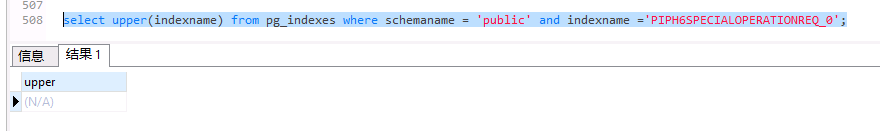
except

select upper(indexname) from pg\_indexes where schemaname = 'public' and indexname not like ' sys\_%'; 结果为

PIPH6SPCIALOPERATIONREQ\_0 显示没有建此索引。



这个索引已建



# 序列数据

oracle:

SELECT count(\*) FROM user\_sequences; 318

pg：

SELECT count(\*)

FROM information\_schema.sequences

WHERE sequence\_schema = current\_schema; 0

# 数据量的检查

1）在oracle数据库中执行：

-------------------------------------------------------------------------------

create table TMP\_GET\_ACTURAL\_TABLE\_COUNT(table\_name varchar(50),table\_cnt int);

select \* from TMP\_GET\_ACTURAL\_TABLE\_COUNT;

CREATE OR REPLACE PROCEDURE GET\_ACTURAL\_TABLE\_COUNT(isrun integer) AUTHID CURRENT\_USER IS

cursor\_sql SYS\_REFCURSOR;

record\_count number;

BEGIN

FOR cursor\_sql IN (SELECT table\_name FROM user\_tables ) LOOP

EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM ' || cursor\_sql.table\_name INTO record\_count;

INSERT INTO TMP\_GET\_ACTURAL\_TABLE\_COUNT (table\_name, table\_cnt) VALUES (cursor\_sql.table\_name, record\_count);

COMMIT;

END LOOP;

COMMIT;

END;

/

truncate table TMP\_GET\_ACTURAL\_TABLE\_COUNT;

call GET\_ACTURAL\_TABLE\_COUNT(1);

commit;

-------------------------------------------------------------------------------

2）在postgresql数据库中执行。

-------------------------------------------------------------------------------

create table TMP\_GET\_ACTURAL\_TABLE\_COUNT(table\_name varchar(50),table\_cnt bigint);

CREATE OR REPLACE PROCEDURE GET\_ACTURAL\_TABLE\_COUNT(isrun integer)

language plpgsql

AS $$

declare

table\_count bigint;

cursor\_sql record;

begin

for cursor\_sql in(

select table\_name from information\_schema.tables where table\_schema='public' and table\_type ='BASE TABLE'

) loop

execute 'SELECT COUNT(\*) FROM ' || cursor\_sql.table\_name INTO table\_count;

insert into TMP\_GET\_ACTURAL\_TABLE\_COUNT(table\_name,table\_cnt) values(cursor\_sql.table\_name,table\_count);

commit;

end loop;

end;

$$;

truncate table TMP\_GET\_ACTURAL\_TABLE\_COUNT;

/

call GET\_ACTURAL\_TABLE\_COUNT(1);

/

select \* from TMP\_GET\_ACTURAL\_TABLE\_COUNT;

-------------------------------------------------------------------------------

在postgresql创建外部表指向oracle数据库的TMP\_GET\_ACTURAL\_TABLE\_COUNT，并在postgresql库中比较相同表名的数据量的差异。

# 第四部分 oracle数据库迁移约束、序列、视图、过程、函数、触发器、包到PG

# oracle数据库迁移约束

Vi ora2pg.conf

--修改内容如下

TYPE TABLE

ora2pg -c /data01/soft\_tar/export\_dir/ora2pg.conf -o /data01/soft\_tar/export\_dir/test1/output.sql > /data01/soft\_tar/export\_dir/test1/output.sql.log

产生的 output.sql 文件用 table\_split.pl 处理下，将table的创建语句和constraint，index的创建语句分开。

table\_split.pl 内容如下

----------------------------------------------------------------------------------------------------------------------

open(INPUT, "output.sql");

open(OUTPUT, ">", "table.sql");

open(OUTPUT2, ">", "constraints.sql");

open(OUTPUT3, ">", "index.sql");

while(<INPUT>){

$str.=$\_

}

@index=$str=~/CREATE\s\*(?:UNIQUE)?\s\*INDEX\s+\w+\s+ON[^\n]\*;\n/igm;

@constraint=$str=~/ALTER\s\*TABLE[^\n]\*;\n/igm;

$str=~s/CREATE\s\*(?:UNIQUE)?\s\*INDEX\s+\w+\s+ON[^\n]\*;\n//igm;

$str=~s/ALTER\s\*TABLE[^\n]\*;\n//igm;

print OUTPUT3 @index;

print OUTPUT2 @constraint;

运行 perl table\_split.pl

导入约束

psql -U infodba -d tc -f constraints.sql > constraints\_output.log 2>&1

grep ‘psql’ constraints\_output.log

psql:constraints.sql:53:ERROR:cannot alter system column “xmin”

psql:constraints.sql:56:ERROR:cannot alter system column “xmin”

psql:constraints.sql:7127:ERROR: multiple primary keys for table “pom\_lock\_keys”

(alter table pom\_lock\_keys add primary key(puid); ) –主键创造多次，可以忽略。

alter table mmv\_spatial\_cell\_index alter column pxmin set not null;

alter table mmv\_spatial\_cell\_index alter column pxmax set not null;

# oracle数据库迁移序列

Vi ora2pg.conf

--修改内容如下

TYPE SEQUENCE

ora2pg -c /data01/soft\_tar/export\_dir/ora2pg.conf -o /data01/soft\_tar/export\_dir/test1/sequence.sql > /data01/soft\_tar/export\_dir/test1/sequence.sql.log

导入

Vi sequence.sql

把创建和设置schema删除。因为序列创建到public

psql -U infodba -d tc -f sequence.sql > sequence\_output.log 2>&1

检查：grep 'psql' sequence\_output.log

查看oracle序列个数：select count(\*) from user\_sequences; 318

查看PG序列个数：select count(\*) from information\_schema.sequences where sequence\_schema =CURRENT\_SCHEMA; 318

# oracle数据库迁移视图

Vi ora2pg.conf

--修改内容如下

TYPE VIEW

ora2pg -c /data01/soft\_tar/export\_dir/ora2pg.conf -o /data01/soft\_tar/export\_dir/test1/view.sql > /data01/soft\_tar/export\_dir/test1/view.sql.log

导入

psql -U infodba -d tc -f view.sql > view\_output.log 2>&1

grep “psql” view\_output.log

psql:view.sql:744: ERROR: function lpad(bigint,integer,unknown) does not exist

psql:view.sql:892: ERROR: function lpad(bigint,integer,unknown) does not exist

psql:view.sql:1116: ERROR: function lpad(bigint,integer,unknown) does not exist

解决方案：

CREATE OR REPLACE FUNCTION pg\_catalog.lpad(bigint, integer, varchar)

RETURNS text

LANGUAGE sql

IMMUTABLE STRICT

AS $function$

SELECT pg\_catalog.lpad($1::text, $2, $3::text)

$function$

查看oracle视图数量：select count(\*) from user\_views; 284

查看pg视图数量：select count(\*) from information\_schema.views where table\_schema = CURRENT\_SCHEMA; 281

----有相差3个对比一下

Oracle：select view\_name as table\_name from user\_views;

把它导入pg，表名为t1

Pg: select upper(table\_name) from information\_schema.views where table\_schema ='public';

Select \* from t1

Except

select upper(table\_name) from information\_schema.views where table\_schema ='public';

结果为

VL10N\_EINFO

VL10N\_VALUE

VL10N\_AWP0CONTENTNAMS

CREATE

OR REPLACE VIEW VL10N\_EINFO AS SELECT

puid,

SUBSTR( pval\_0, 1, 5 ) AS locale,

SUBSTR( pval\_0, 7, 1 ) AS preference,

SUBSTR( pval\_0, 9, 1 ) AS status,

SUBSTR( pval\_0, 11, 4 ) AS sequence\_no,

SUBSTR( pval\_0, 49, 240 ) AS pval\_0

FROM

PL10N\_EINFO UNION ALL

SELECT

PEINFO.puid,

'NONE' AS locale,

'M' AS preference,

'M' AS status,

lpad( pseq, 4, '0' ) AS sequence\_no,

pval\_0

FROM

PEINFO,

PICML

WHERE

PICML.puid = PEINFO.puid

AND PICML.VLA\_487\_5 = 0;

CREATE

OR REPLACE VIEW VL10N\_VALUE AS SELECT

puid,

SUBSTR( pval\_0, 1, 5 ) AS locale,

SUBSTR( pval\_0, 7, 1 ) AS preference,

SUBSTR( pval\_0, 9, 1 ) AS status,

SUBSTR( pval\_0, 11, 4 ) AS sequence\_no,

SUBSTR( pval\_0, 49, 256 ) AS pval\_0

FROM

PL10N\_VALUE UNION ALL

SELECT

PVALUE.puid,

'NONE' AS locale,

'M' AS preference,

'M' AS status,

lpad( pseq, 4, '0' ) AS sequence\_no,

pval\_0

FROM

PVALUE,

PSTXT

WHERE

PSTXT.puid = PVALUE.puid

AND PSTXT.VLA\_491\_19 = 0;

CREATE

OR REPLACE VIEW VL10N\_AWP0CONTENTNAMES AS SELECT

puid,

SUBSTR( pval\_0, 1, 5 ) AS locale,

SUBSTR( pval\_0, 7, 1 ) AS preference,

SUBSTR( pval\_0, 9, 1 ) AS status,

SUBSTR( pval\_0, 11, 4 ) AS sequence\_no,

SUBSTR( pval\_0, 49, 128 ) AS pval\_0

FROM

PL10N\_AWP0CONTENTNAMES UNION ALL

SELECT

PAWP0CONTENTNAMES.puid,

'NONE' AS locale,

'M' AS preference,

'M' AS status,

lpad( pseq, 4, '0' ) AS sequence\_no,

pval\_0

FROM

PAWP0CONTENTNAMES,

PAWP0TILETEMPLATE

WHERE

PAWP0TILETEMPLATE.puid = PAWP0CONTENTNAMES.puid

AND PAWP0TILETEMPLATE.VLA\_1310\_9 = 0;

# oracle数据库迁移过程

Vi ora2pg.conf

--修改内容如下

TYPE PROCEDURE

ora2pg -c /data01/soft\_tar/export\_dir/ora2pg.conf -o /data01/soft\_tar/export\_dir/test1/procedure.sql > /data01/soft\_tar/export\_dir/test1/procedure.sql.log

导入

psql -U infodba -d tc -f procedure.sql > procedure\_output.log 2>&1

grep ‘psql’ procedure\_output.log

psql:procedure.sql:59:ERROR: syntax error at or near “dbms\_stats”

9.5函数

Vi ora2pg.conf

--修改内容如下

TYPE FUNCTION

ora2pg -c /data01/soft\_tar/export\_dir/ora2pg.conf -o /data01/soft\_tar/export\_dir/test1/function.sql > /data01/soft\_tar/export\_dir/test1/function.sql.log

导入 (注意把schema修改去掉)

psql -U infodba -d infodba\_test -f function.sql > function\_output.log 2>&1

psql:procedure.sql:59 ERROR：syntax error at or near”dbms\_stats”

CREATE OR REPLACE PROCEDURE tc\_indexer\_proc () AS $body$

BEGIN

dbms\_stats.gather\_schema\_stats(ownname=>'infodba', estimate\_percent=>100, method\_opt=>'FOR ALL COLUMNS SIZE AUTO', degree=>32, cascade=>true, no\_invalidate=>FALSE);

END;

$body$

LANGUAGE PLPGSQL

SECURITY DEFINER

;

存储过程tc\_indexer\_proc ()导入报错。 --记录，这个oracle特性。

# oracle数据库迁移函数

Vi ora2pg.conf

--修改内容如下

TYPE FUNCTION

ora2pg -c /data01/soft\_tar/export\_dir/ora2pg.conf -o /data01/soft\_tar/export\_dir/test1/function.sql > /data01/soft\_tar/export\_dir/test1/function.sql.log

导入（注意把schema删除）

psql -U infodba -d tc -f function.sql > function\_output.log 2>&1

grep ‘psql’ function\_output.log

psql:function.sql:81:ERROR:syntax error at or near “is” ---此函数初始化里有，直接复制初始化库的。

# oracle数据库迁移触发器

Vi ora2pg.conf

--修改内容如下

TYPE TRIGGER

ora2pg -c /data01/soft\_tar/export\_dir/ora2pg.conf -o /data01/soft\_tar/export\_dir/test1/trigger.sql > /data01/soft\_tar/export\_dir/test1/trigger.sql.log

导入

psql -U infodba -d tc -f trigger.sql > trigger\_output.log 2>&1

报错



触发器有100个，原生PG不支持，是不是使用，因为初始化库触发器。

# oracle数据库迁移包

Vi ora2pg.conf

--修改内容如下

TYPE PACKAGE

ora2pg -c /data01/soft\_tar/export\_dir/ora2pg.conf -o /data01/soft\_tar/export\_dir/test1/package.sql > /data01/soft\_tar/export\_dir/test1/package.sql.log

导入

psql -U infodba -d tc -f package.sql > package\_output.log 2>&1

# oracle迁移数据库与初始化库int字段类型对比

# 1.导出所有int字段

PG：select table\_name,column\_name,data\_type,udt\_name from information\_schema.columns where table\_schema = 'public' and udt\_name like 'int%';

**2.oracle迁移到PG的字段int，导入表名为oracle**

select table\_name,column\_name,data\_type,udt\_name from information\_schema.columns where table\_schema = 'public' and udt\_name like 'int%';

**3.PG初始化的数据库的字段int，导入表名为pgi**

select table\_name,column\_name,data\_type,udt\_name from information\_schema.columns where table\_schema = 'public' and udt\_name like 'int%';

**4.Oracle迁移到PG与PG初始化相同的int字段**

Select table\_name,column\_name,data\_type,udt\_name from pgi

Intersect

Select table\_name,column\_name,data\_type,udt\_name from oracle;

**5.Oracle迁移到PG与PG初始化不同的int字段**

select oracle.table\_name,oracle.column\_name,oracle.data\_type,oracle.udt\_name,pgi.data\_type as pg\_type, pgi.udt\_name as pg\_udt from oracle,pgi

where oracle.table\_name=pgi.table\_name and oracle.column\_name = pgi.column\_name and (oracle.data\_type !=pgi.data\_type or oracle.udt\_name != pgi.udt\_name);

总结：1）int类型，只有一条相同的int4

2）int类型有条相同其它都不同，从oracle迁移过来都是int8，初始化int4多，int2少。 3）基本上int不同。

# 第六部分 oracle迁移的数据库与初始化数据库比较

# 表的迁移

Pg:init表名

Select table\_name from information\_schema.tables where table\_schema='public' and table\_type ='BASE TABLE'; 3948

Pg:oracle表名：

Select table\_name from information\_schema.tables where table\_schema='public' and table\_type ='BASE TABLE'; 4002

select \* from dblink('dbname=tc\_test','Select table\_name from information\_schema.tables where table\_schema= ''public'' and table\_type =''BASE TABLE''') as t(table\_name varchar(50))

except

select \* from dblink('dbname=tc','Select table\_name from information\_schema.tables where table\_schema=''infodba'' and table\_type =''BASE TABLE''') as t(table\_name varchar(50));

在初始化库上有的表，oracle中没有的14

pg\_dump -s -t pfndoimpactingobjects -f output\_file.sql tc\_test

psql -U infodba -d tc -f output\_file.sql > output.log 2>&1

grep 'psql' output.log

(初始化的函数先创建

lock\_keys\_deltete\_trigger\_func、lock\_keys\_insert\_trigger\_func）

# 索引迁移

Pg:oracle索引名

Select indexname from pg\_indexes where schemaname='public' and indexname not like 'sys\_%'; 6945

Pg:init索引名

Select indexname from pg\_indexes where schemaname='public' and indexname not like 'sys\_%'; 6884

select \* from dblink('dbname=tc\_test','select indexname from pg\_indexes where schemaname=''public'' and indexname not like ''sys\_%''') as t(table\_name varchar(100))

except

select \* from dblink('dbname=tc','Select indexname from pg\_indexes where schemaname=''public'' and indexname not like ''sys\_%''') as t(table\_name varchar(100));

在初始化库上有的索引，oracle中没有的 13

select \* from pg\_indexes where indexname in (用notep++填定索引名);

CREATE UNIQUE INDEX auditlog\_puid\_pk ON public.auditlog USING btree (puid);

CREATE UNIQUE INDEX pipfnd0effectivityexprblock\_2 ON public.pfnd0effectivityexprblock USING btree ((

CASE arev\_category

WHEN 48 THEN rfnd0affectedobjectu

WHEN 16 THEN rfnd0affectedobjectu

ELSE NULL::character varying

END));

CREATE UNIQUE INDEX pipfnd0variantexprblock\_2 ON public.pfnd0variantexprblock USING btree ((

CASE arev\_category

WHEN 48 THEN rfnd0affectedobjectu

WHEN 16 THEN rfnd0affectedobjectu

ELSE NULL::character varying

END));

CREATE INDEX piph6specialoperationreq\_0 ON public.ph6specialoperationreq USING btree (pval\_0);

CREATE INDEX pipimanrelation\_4 ON public.pimanrelation USING btree (rsecondary\_objectu);

CREATE INDEX pipimanrelation\_9 ON public.pimanrelation USING btree (pfnd0copystableid);

CREATE UNIQUE INDEX pipqs\_product\_path\_index\_1 ON public.pqs\_product\_path\_index USING btree (pchilditem, pproduct, pitempath);

CREATE INDEX pipqs\_product\_path\_index\_2 ON public.pqs\_product\_path\_index USING btree (pproduct);

CREATE UNIQUE INDEX pipqs\_product\_path\_index\_3 ON public.pqs\_product\_path\_index USING btree (pseq);

CREATE INDEX pipqs\_product\_path\_index\_4 ON public.pqs\_product\_path\_index USING btree (plast\_modified\_date);

CREATE INDEX pitie\_am\_obj\_hash\_0 ON public.tie\_am\_obj\_hash USING btree (object\_uid, am\_readexpr\_hash);

CREATE UNIQUE INDEX pk\_principal\_obj\_uid ON public.principal\_obj\_in\_trans\_table USING btree (transaction\_id, principal\_obj\_uid);

CREATE UNIQUE INDEX pk\_site\_id ON public.export\_to\_site\_table USING btree (transaction\_id, site\_id);

# 约束

# 序列

Pg：init序列 312

Select count(\*) from information\_schema.sequences where sequence\_schema='public';

'select sequence\_name from information\_schema.sequences where sequence\_schema= ''public'''

Pg:oracle序列318

select \* from dblink('dbname=tc\_test', 'select sequence\_name from information\_schema.sequences where sequence\_schema= ''public''') as t(table\_name varchar(100))

except

select \* from dblink('dbname=tc', 'select sequence\_name from information\_schema.sequences where sequence\_schema= ''public'''

) as t(table\_name varchar(100)); 312说明序列没有相同名字

导出序列：pg\_dump -U sequence\_name -f output\_file.sql database\_name

psql -U infodba -d tc -f output\_file.sql > output.log 2>&1

grep 'psql' output.log

# 视图的迁移

select count(\*) from information\_schema.views where table\_schema = 'public';

select \* from dblink('dbname=tc\_test','select table\_name from information\_schema.views where table\_schema = ''public''') as t(table\_name varchar(100))

except

select \* from dblink('dbname=tc',' select table\_name from information\_schema.views where table\_schema = ''public''') as t(table\_name varchar(100));

pg\_dump -U infodba -d tc -t vl10n\_fnd0instructions -s -f output\_file.sql

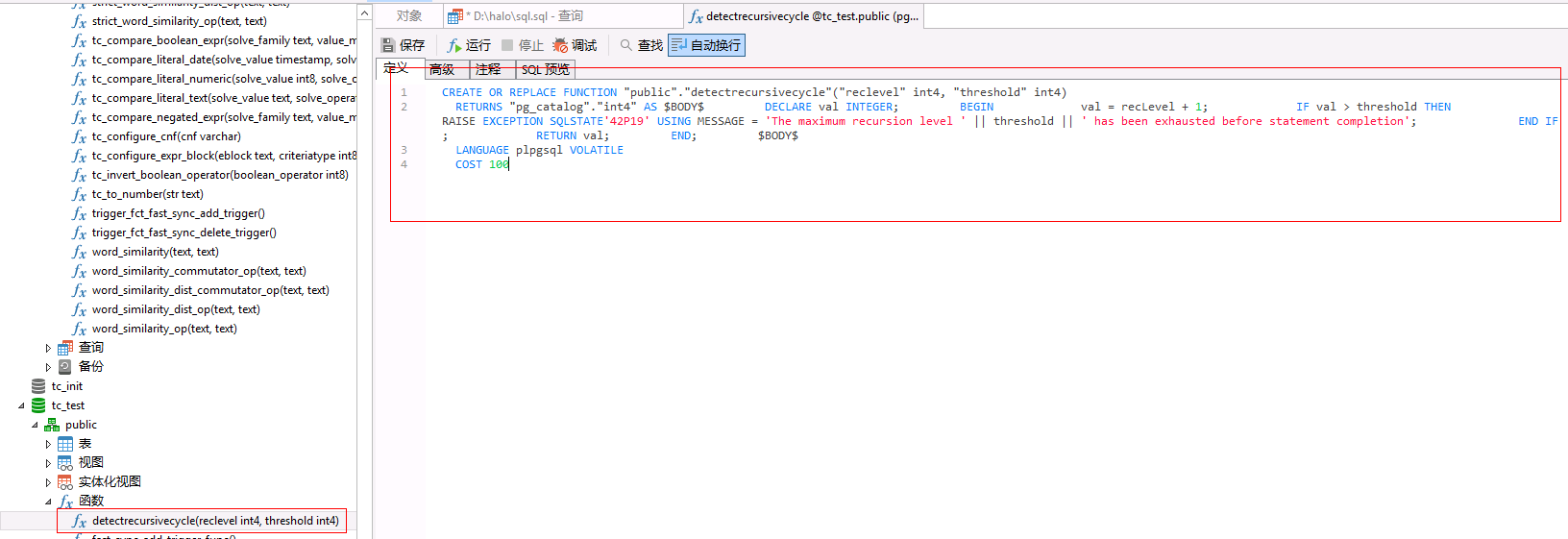
psql -U infodba -d tc -f output\_file.sql > output.log 2>&1

没有创建上的视图需要确认： user\_ind\_columns、user\_tables、user\_ind\_expressions、

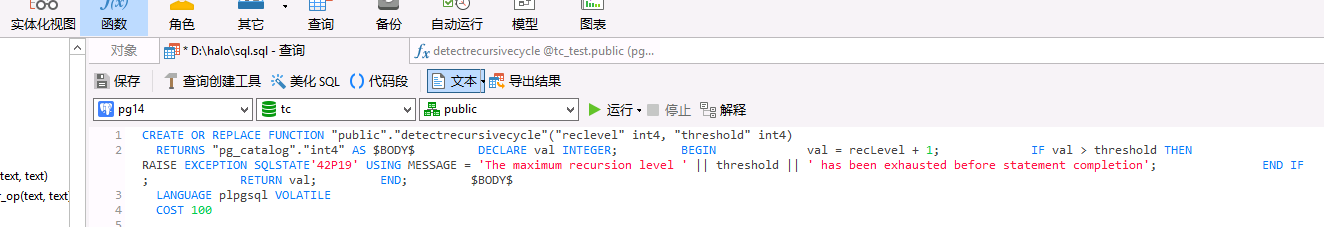
User\_indexes、user\_tab\_columns、user\_views

# 存储过程、函数、触发器

打开工具Navicat Premium，选择相应的库—函数。然后复制。



选择相应的库和schema，然后粘贴，再运行。



# 第七部分 验证数据

# 表结构的一致性-全量 ；验证

1. oracle表结构与oracle迁移到PG数据库的表结构比较;验证

Oracle：select table\_name,column\_name from user\_tab\_columns where column\_id <> 0 and table\_name not in(

select view\_name from user\_views) order by table\_name,column\_name;19144 

PG：select table\_name,column\_name from information\_schema.columns where table\_schema='public' and table\_name not in(select table\_name from information\_schema.views) order by table\_name,column\_name; 19261 

Oracle:导入表名为oracle，pg:导入表名为pg

Select \* from oracle

Except

Select upper(table\_name),upper(column\_name) from pg; 记录有2条

Table\_name column\_name

Mmv\_spatial\_cell\_index xmax

Mmv\_spatial\_cell\_index xmix

这两条导入就改了字段名。所以结论。Oracle库的表结构与迁移过来的表结构一样的。

1. oracle迁移到PG数据库与初始化库的一结构比较;验证;

PG：select table\_name,column\_name from information\_schema.columns;

把数据导出

PGtoPG:

Select select pg.table\_name,pg.column\_name,pg.ordinal\_position::int-1 from (select table\_name,column\_name,data\_type,character\_maximum\_length,character\_octet\_length,numeric\_precision\_radix,numeric\_scale,interval\_type from pg where data\_type not in ('double precision','integer','smallint')

except

select table\_name,column\_name,data\_type,character\_maximum\_length,character\_octet\_length,numeric\_precision\_radix,numeric\_scale,interval\_type from oracle) as a,pg where a.table\_name=pg.table\_name and a.column\_name=pg.column\_name;

select b.table\_name,b.column\_name,pg.data\_type,pg.character\_octet\_length,pg."column\_name" from (select pg.table\_name,pg.column\_name,pg.ordinal\_position::int - 1 as position from (select table\_name,column\_name,data\_type,character\_maximum\_length,character\_octet\_length,numeric\_precision\_radix,numeric\_scale,interval\_type from pg where data\_type not in ('double precision','integer','smallint')

except

select table\_name,column\_name,data\_type,character\_maximum\_length,character\_octet\_length,numeric\_precision\_radix,numeric\_scale,interval\_type from oracle) as a,pg where a.table\_name=pg.table\_name and a.column\_name=pg.column\_name) as b,pg where b.table\_name=pg.table\_name and b.position=pg.ordinal\_position::int order by ordinal\_position;

表名：少列名

Ptc\_project：pfnd0projectcategory

2：数据量的一致性-全量；验证

1）在oracle数据库中执行：

oracle

create table TMP\_GET\_ACTURAL\_TABLE\_COUNT(table\_name varchar(50),table\_cnt int);

select \* from TMP\_GET\_ACTURAL\_TABLE\_COUNT;

CREATE OR REPLACE PROCEDURE GET\_ACTURAL\_TABLE\_COUNT(isrun integer) AUTHID CURRENT\_USER IS

cursor\_sql SYS\_REFCURSOR;

record\_count number;

BEGIN

FOR cursor\_sql IN (SELECT table\_name FROM user\_tables ) LOOP

EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM ' || cursor\_sql.table\_name INTO record\_count;

INSERT INTO TMP\_GET\_ACTURAL\_TABLE\_COUNT (table\_name, table\_cnt) VALUES (cursor\_sql.table\_name, record\_count);

COMMIT;

END LOOP;

COMMIT;

END;

/

truncate table TMP\_GET\_ACTURAL\_TABLE\_COUNT;

call GET\_ACTURAL\_TABLE\_COUNT(1);

commit;

2）在postgresql数据库中执行。

create table TMP\_GET\_ACTURAL\_TABLE\_COUNT(table\_name varchar(50),table\_cnt bigint);

CREATE OR REPLACE PROCEDURE GET\_ACTURAL\_TABLE\_COUNT(isrun integer)

language plpgsql

AS $$

declare

table\_count bigint;

cursor\_sql record;

begin

for cursor\_sql in(

select table\_name from information\_schema.tables where table\_schema='public' and table\_type ='BASE TABLE'

) loop

execute 'SELECT COUNT(\*) FROM ' || cursor\_sql.table\_name INTO table\_count;

insert into TMP\_GET\_ACTURAL\_TABLE\_COUNT(table\_name,table\_cnt) values(cursor\_sql.table\_name,table\_count);

commit;

end loop;

end;

$$;

truncate table TMP\_GET\_ACTURAL\_TABLE\_COUNT;

/

call GET\_ACTURAL\_TABLE\_COUNT(1);

/

select \* from TMP\_GET\_ACTURAL\_TABLE\_COUNT;在postgresql创建外部表指向oracle数据库的TMP\_GET\_ACTURAL\_TABLE\_COUNT，并在postgresql库中比较相同表名的数据量的差异。

3：数据本身的一致性-抽查30% ；验证

4：数据编码的一致性-抽查30%;验证同上