CHINA POSTGRESQL ASSOCIATION

Oracle到PostgreSQL最佳实践

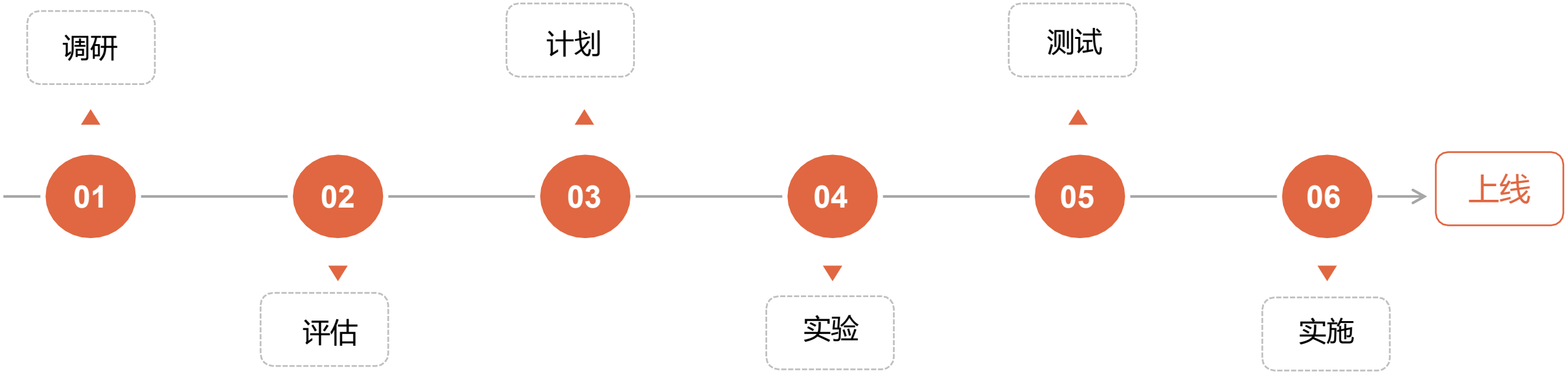
刘彦红

-- 中国PG分会金牌认证讲师

CONTENT

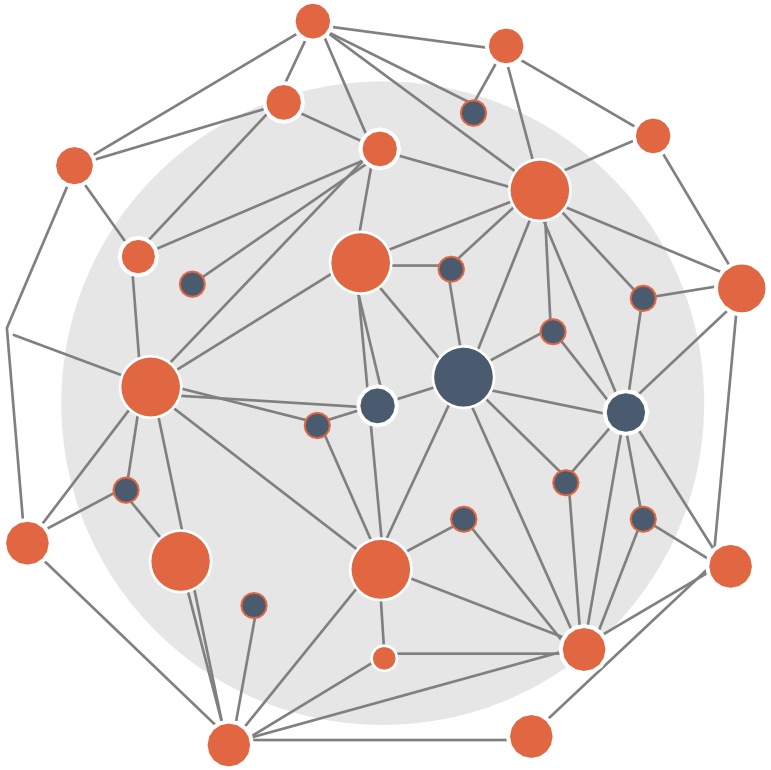
- 迁移流程
- 迁移工具
- 数据类型迁移
- 未兼容语法迁移

01. 迁移实施步骤



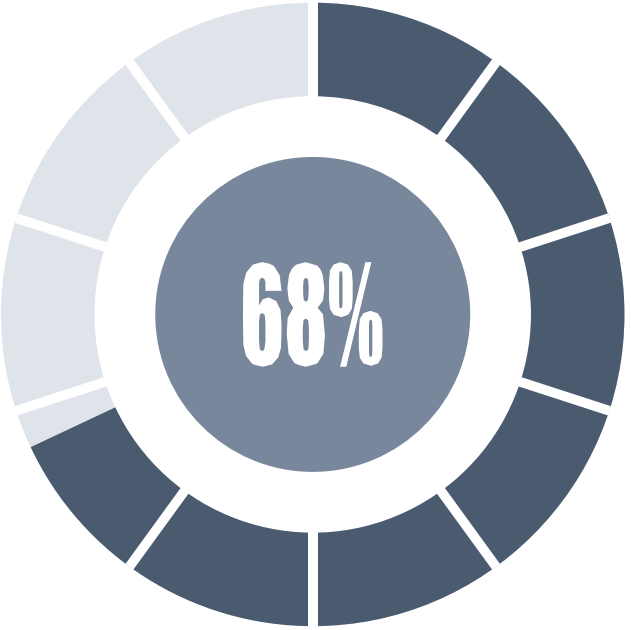
调研

- 数据库大小
- 表数量，是否有分区表
- 视图数量，是否有物化视图
- 存储过程数量，平均复杂度或行数
- 自定义函数数量，及复杂度或行数
- 触发器数量
- 索引数量
- 字段特殊类型



评估

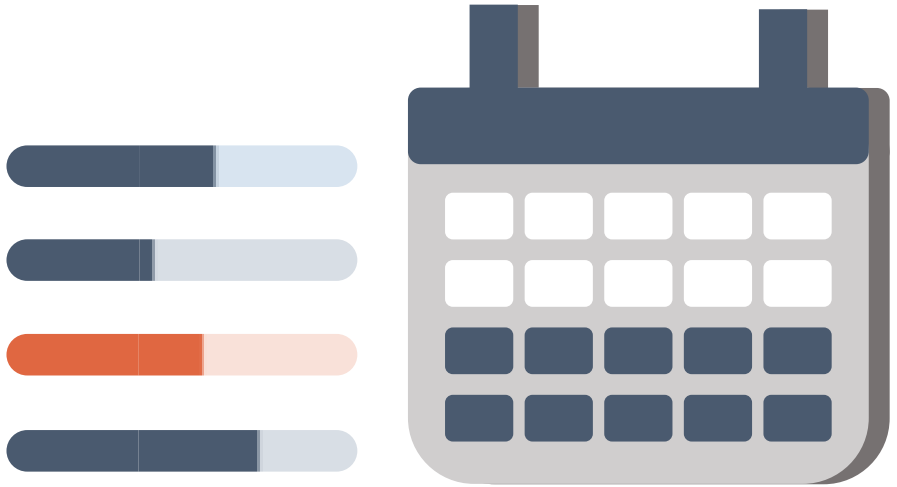
1. **根据调研的结果评估工作量**
   - ✓ 10T以下可以使用迁移工具，超过10T建议HVR
   - ✓ 一个工程师一天可以修改10个存储过程，或者30个view
   - ✓ 等等…
2. **常用或复杂函数，修改较繁琐，需通过其他方式解决**
   - ✓ 写类似函数替代
   - ✓ 等等…



计划

根据工作量及迁移复杂度，制定计划

- ✓ 计算数据库迁移完成时间
- ✓ 应用迁移时间
- ✓ 测试时间
- ✓ 正式迁移时间
- ✓ 上线时间



实验

备份一个正在使用的数据库，做一次迁移

- ✓ 记录迁移中问题及解决方式
- ✓ 整理实验步骤
- ✓ 汇总修改后的SQL
- ✓ 比对数据完整性
- ✓ 记录操作时间

测试

将应用连上PG，进行功能测试

- ✓ 数据准确性
- ✓ 功能健壮性
- ✓ 函数兼容性
- ✓ 尽量全面详细

实施

根据操作时间停机，迁移数据库，启动数据库

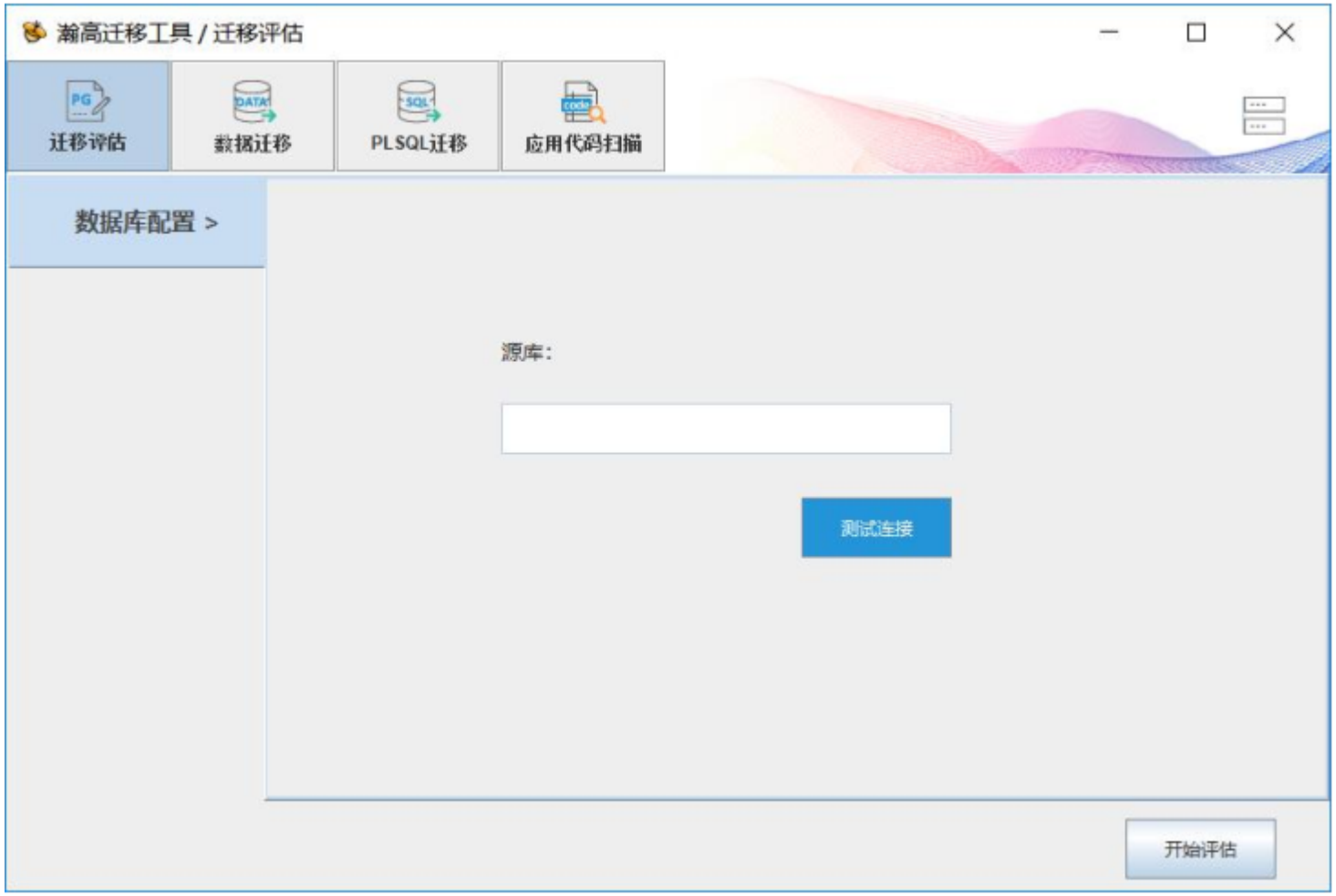
- ✓ 重复实验中内容
- ✓ 多方案准备

上线

迁移完成正式数据库，切换应用连接，完成上线

- ✓ 监控运行状态
- ✓ 日志错误
- ✓ 定期巡检

迁移工具的使用



- 迁移评估
- 数据迁移
- 数据对比
- Plsql迁移
- 应用代码扫描

迁移评估

- Data 对象数量统计

  TABLE、INDEX、SEQUENCE、LOB 和SYNONYM

- Data 对象大小

  TABLE、INDEX、LOBINDEX 和 LOBSEGMENT

- Top10 表行数
- Top10 数据对象大小
- 分区表统计
- 触发器统计
- PLSQL 对象数量统计

  PACKAGE、VIEW、PACKAGE BODY、FUNCTION、PROCEDURE、MATERIALIZED VIEW、TYPE 和 TRIGGER

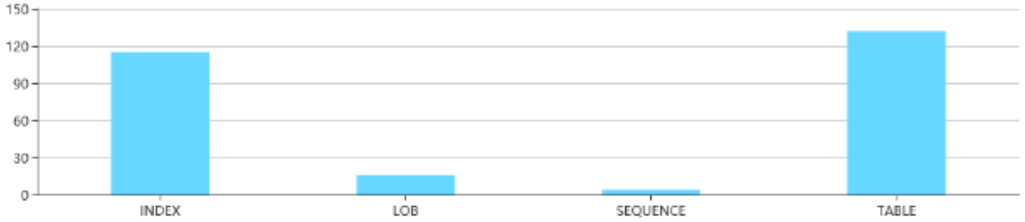
- TOP 10 PLSQL 对象行数

迁移评估

评估结果

评估开始时间：2019-08-05 10:01:19　评估结束时间：2019-08-05 10:01:21

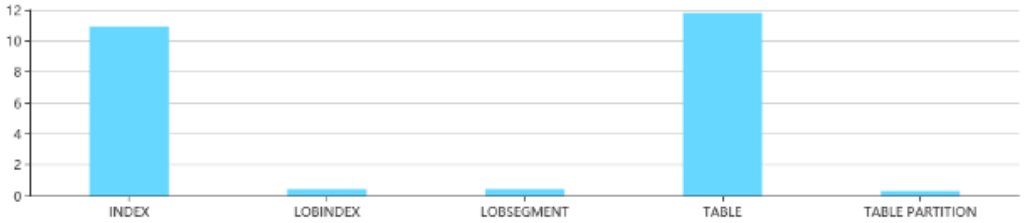
Data对象数量统计



Data对象数量统计

序号	对象名称	对象数量
1	INDEX	115
2	LOB	16
3	SEQUENCE	4
4	TABLE	132

Data对象大小



Data对象大小

序号	对象名称	对象大小(MB)
1	INDEX	10.94
2	LOBINDEX	0.44
3	LOBSEGMENT	0.44

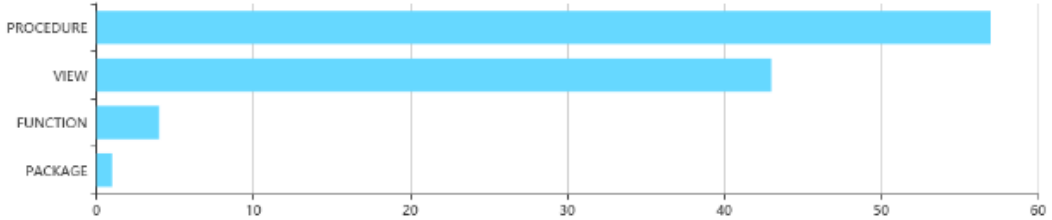
分区表统计

序号	表名称	分区数	子分区数
1	LOCAL_ZF_SUM	5	0

触发器统计

序号	触发器名称	触发器类型	触发事件	触发对象类型	触发表名

PLSQL对象数量统计



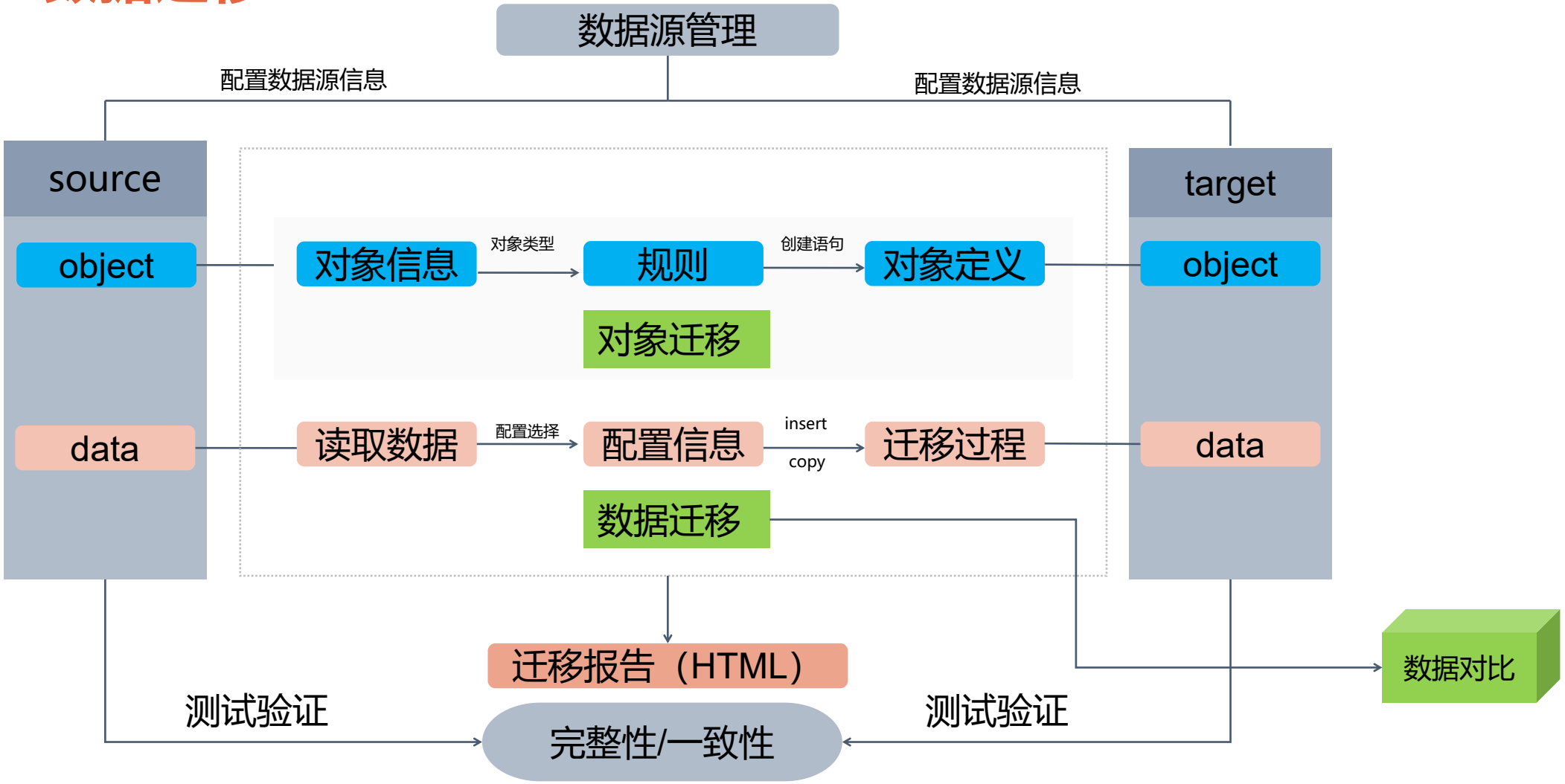
PLSQL对象数量统计

序号	对象名称	对象数量
1	PROCEDURE	57
2	VIEW	43
3	FUNCTION	4
4	PACKAGE	1

TOP 10 PLSQL对象行数

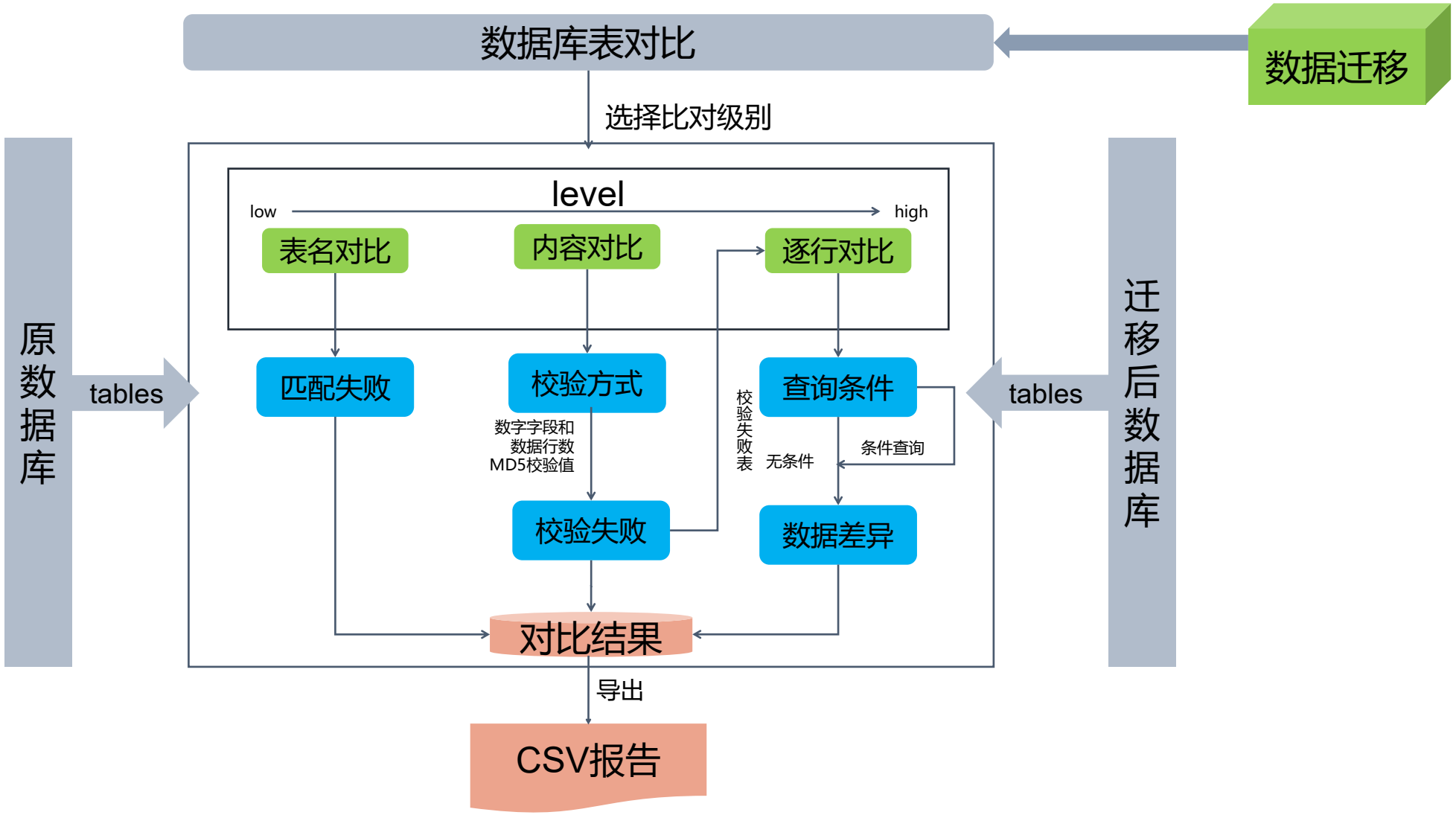
序号	类型	名称	行数
1	PROCEDURE	PROC_TONGJI_TZRYBZXX	1629
2	PROCEDURE	PROC_SMZ_TONGJI_RYBZXX	1317
3	PROCEDURE	PROC_RELOAD_TONGJI_REPORT_DATA	1245

数据迁移



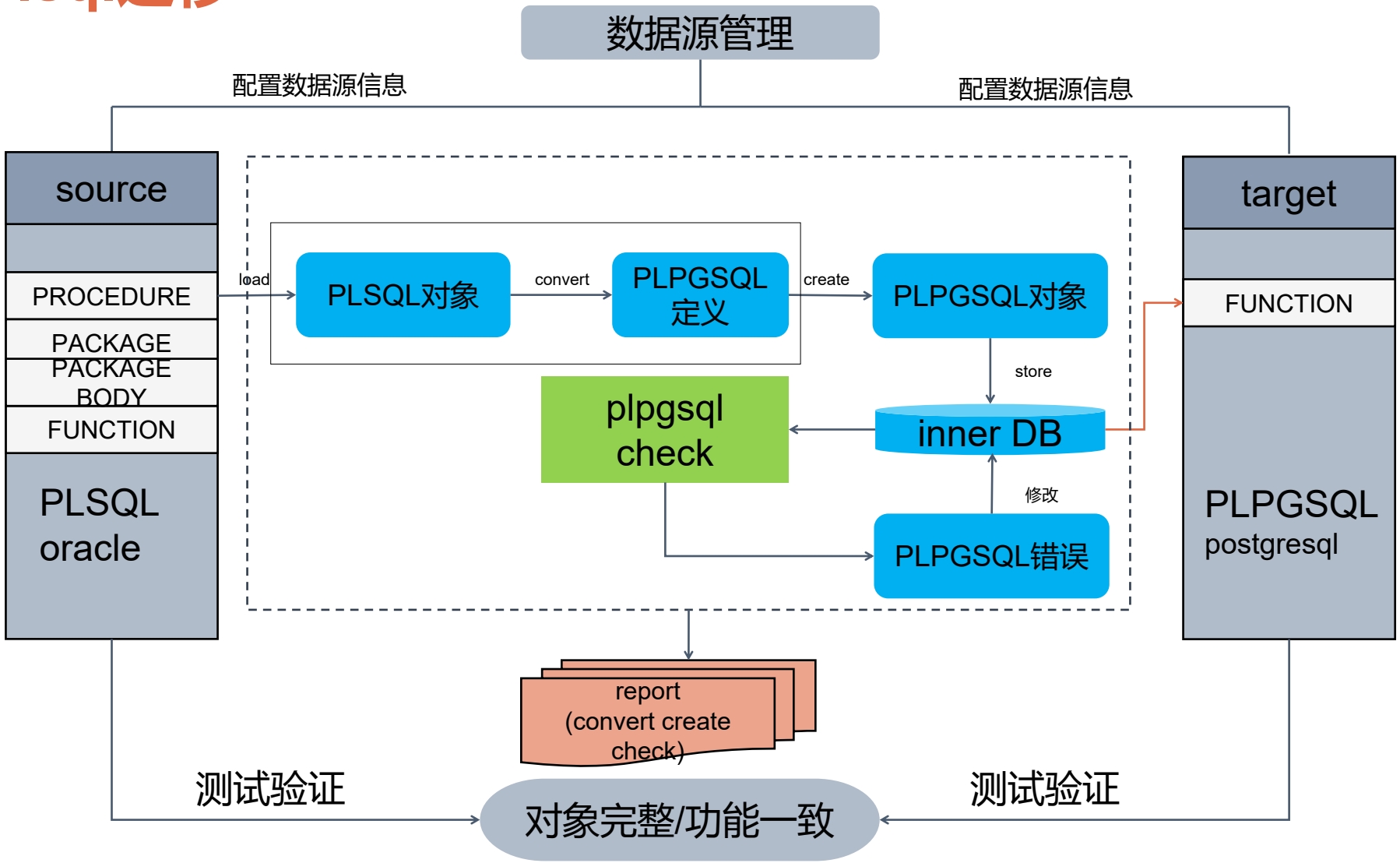
数据库基本对象及数据迁移--架构图

数据对比



数据库表对比--架构图

Plsql迁移



PLSQL转换迁移--架构图

应用代码扫描

文件修改个数	4
文件总个数	49

序号	类	行数	SQL内容	建议
1	com.itsv.oa.auditadvice.dao.AuditadviceDao	95	select a from Auditinfo a , Auditadvice v where 1=1 and v.auditid=a.auditid and (a.finish=' ? ' or a.finish = ' ? ') and v.userid=' ? ' and v.isaudit=' ? ' and a.infoname like '% ? %' and a.registerdate>=to_date(' ? 00:00:00','yyyy-mm-dd hh24:mi:ss') and a.registerdate<=to_date(' ? 23:59:59','yyyy-mm-dd hh24:mi:ss') order by a.registerdate	ERROR expression :to_date(to_char(now(),'yyyyMMdd'),'yyyy-MM-dd') ERROR expression :to_date(to_char(now(),'yyyyMMdd'),'yyyy-MM-dd')
2	com.itsv.oa.auditadvice.dao.AuditadviceDao	95	select a from Auditinfo a , Auditadvice v where 1=1 and v.auditid=a.auditid and v.userid=' ? ' and v.isaudit=' ? ' and a.infoname like '% ? %' and a.registerdate>=to_date(' ? 00:00:00','yyyy-mm-dd hh24:mi:ss') and a.registerdate<=to_date(' ? 23:59:59','yyyy-mm-dd hh24:mi:ss') order by a.registerdate	ERROR expression :to_date(to_char(now(),'yyyyMMdd'),'yyyy-MM-dd') ERROR expression :to_date(to_char(now(),'yyyyMMdd'),'yyyy-MM-dd')
3	com.itsv.oa.auditadvice.dao.AuditadviceDao	95	select a from Auditinfo a , Auditadvice v where 1=1 and v.auditid=a.auditid and (a.finish=' ? ' or a.finish = ' ? ') and 1=2 and v.isaudit=' ? ' and a.infoname like '% ? %' and a.registerdate>=to_date(' ? 00:00:00','yyyy-mm-dd hh24:mi:ss') and a.registerdate<=to_date(' ? 23:59:59','yyyy-mm-dd hh24:mi:ss') order by a.registerdate	ERROR expression :to_date(to_char(now(),'yyyyMMdd'),'yyyy-MM-dd') ERROR expression :to_date(to_char(now(),'yyyyMMdd'),'yyyy-MM-dd')
4	com.itsv.oa.auditadvice.dao.AuditadviceDao	95	select a from Auditinfo a , Auditadvice v where 1=1 and v.auditid=a.auditid and 1=2 and v.isaudit=' ? ' and a.infoname like '% ? %' and a.registerdate>=to_date(' ? 00:00:00','yyyy-mm-dd hh24:mi:ss') and a.registerdate<=to_date(' ? 23:59:59','yyyy-mm-dd hh24:mi:ss') order by a.registerdate	ERROR expression :to_date(to_char(now(),'yyyyMMdd'),'yyyy-MM-dd') ERROR expression :to_date(to_char(now(),'yyyyMMdd'),'yyyy-MM-dd')

激

1、字符类型迁移

ORACLE 2 PG DATATYPE CAST	
ORACLE	PG
CHAR(n)	CHAR(n)
NCHAR	CHAR(n)
VARCHAR2	VARCHAR/TEXT
NVARCHAR2	VARCHAR/TEXT

差异

Oracle

- char类型的最大长度是2000字节，varchar2是4000个字节。
- nchar，nvarchar2类型的最大长度根据数据集不同而不同。最终的byte数不能超过2000和4000。
- char类型，不带字符数的时候，是1个字符（char(1)）
- varchar2必须有字符数。

Postgresql

- 理论最大数据长度是1G，默认值是10 * 1024 * 1024=10485760
- char类型，不带字符数的时候，是默认一个字符(char(1))
- varchar不带字符数的时候，没有限制。
- Text变长，最大长度没有限制

2、数字类型迁移

ORACLE 2 PG DATATYPE CAST	
ORACLE	PG
NUMBER(p,0), NUMBER(p)	NUMERIC/INTEGER/DECIMAL
NUMBER(p,s)	NUMERIC(p,s)/DECIMAL(p,s)
BINARY_FLOAT	REAL/DOUBLE PRECISION
BINARY_DOUBLE	DOUBLE PRECISION
FLOAT	REAL/DOUBLE PRECISION

PG的数字类型有三类。

整数类型：**smallint，integer和bigint**。

任意精度类型：**numeric和decimal**两种(其中decimal等同于numeric)，

浮点数类型：**real和double**。

number(p, s)

当小数位数(s)=0时

可以对应PG的smallint，integer，bigint，numeric(p)等多种数据类型。由于smallint，Integer，bigint的算术运算效率比numberic高的多，所以迁移时建议如下处理：

- 当p <= 4时，可以使用smallint和Integer，推荐使用Integer，因为它在取值范围、存储空间、性能之间最为平衡。只有磁盘空间紧张的情况下，才建议使用smallint。
- 当4 < p <= 9时，应该使用integer
- 当9 < p <= 18时，使用bigint
- 当p > 18时，使用numeric(p，0)

3、日期类型迁移

ORACLE 2 PG DATATYPE CAST	
ORACLE	PG
DATE	TIMESTAMP [(0)]
TIMESTAMP(p)	TIMESTAMP [(p)]
TIMESTAMP WITH TIME ZONE	TIMESTAMP [(p)] WITH TIME ZONE
TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMP [(p)] WITH TIME ZONE
INTERVAL YEAR(p) TO MONTH、 INTERVAL DAY(p) TO SECOND(s)	INTERVAL(s)

差异

Date类型

Oracle的Date类型包括年、月、日、时、分、秒六个字段，

PG中，可以使用Timestamp(0) without time zone来对应。

注意：PG中的Date类型仅仅包含日期，不包括时、分、秒信息。因此不能使用Date类型来对应Oracle的Date类型

Timestamp(p)类型

TIMESTAMP 类型的参数 s：

在 Oracle 中， 0<=s<=9,缺省为 6;

在 PG 中， 0<=s<=6,缺省为 6。

迁移过程中，若 S 大于 6 时取 6，小于 6 时取原始数值。

4、大对象类型

ORACLE 2 PG DATATYPE CAST

ORACLE	PG
CLOB	TEXT
NCLOB	TEXT
BLOB	BYTEA
BFILE	BYTEA
LONG	TEXT
LONG RAW	BYTEA

PG的大数据类型是存储二进制数据的**bytea**和存储字符类型的**text**。

5、其他类型

ORACLE 2 PG DATATYPE CAST	
ORACLE	PG
ROWID	CHAR
UROWID	VARCHAR(n)
RAW	BYTEA
XMLTYPE	XML

语法兼容

1. limit -- oracle 中的rownum对应PG的limit

```
ORACLE
select * from test where rownum<=10
PG
select * from test where limit<=10
Select name from test limit 2 offset 1
```

2. 左链接 右链接

PG不支持+，支持 left join、right join

3. || 在oracle中 null || ‘abc’ 结果为abc， PG 结果为null，需要加判断COALESCE（,’’）。

Postgresql使用coalesce实现类似oracle的NVL方法

语法兼容

3. PG数据类型的兼容比较严格

固定字段查询后类型为unknown,但在关联是可能有比较unknown=text 这种情况就会报错

```
highgo=# select 1 ;
 ?column?
----------
        1
(1 行记录)
```

```
highgo=# select 1::text;
 text
------
 1
(1 行记录)
```

```
highgo=# select decode(sex, 'M','男','F','女', '其他') from person;
ERROR:  42883: could not identify an equality operator for type unknown

highgo=# select decode(sex, 'M'::varchar,'男'::varchar,'F'::varchar,'女'::varchar,
'其他'::varchar) from person;
 decode
--------
 男
 女
 其他
(3行记录)
```

语法兼容

4. 数字运算符

Oracle

```
取 模 ： select mod(6/4) from dual;
幂    ： select power(2,3)from dual;
平 方 根 ： select sqrt(25) from dual;
立 方 根 ： select power(27,1/3) from
dual;
求绝对值:  select abs(-5.0)from dual;
```

Postgresql

```
取 模 ： select 6%4; 也 可 用 select mod(6,4);
幂    ： select 2^3; 也 可 用 select power(2,3);
平 方 根 ： select |/25; 也 可 用 select sqrt(25)
立 方 根 ： select ||/27; 也 可 用 select cbrt(27);
求绝对值:  select @ -5.0;也可用 select abs(-5.0);
```

5. 比较运算符

PG 比较操作符之间不允许有空格， Oracle 比较操作符之间可以有空格

语法兼容

6. 列转行

```
cqdb=> select * from test ;
 name
------
 AA
 BB
 CC
(3 rows)

cqdb=> select string_agg(name,',') from test;
 string_agg
------------
 AA,BB,CC
(1 row)
```

行转列

```
cqdb=> select * from test ;
   name
-----------
 A,B,C,D,E
(1 row)

cqdb=> select  regexp_split_to_table(name,',')
from test;
 regexp_split_to_table
-----------------------
 A
 B
 C
 D
 E
(5 rows)
```

语法兼容

7. 序列

Oracle 序列 默认的序列的最大值是： 9999999999999999999999999999

```
create sequence test2_seq
 start with 1
 minvalue 1
 increment by 1
 maxvalue 100
 nocache
 noorder
 cycle;
```

Oracle的序列创建的时候，可以指定起始值 (**start with**)，最大值 (**maxvalue**，默认是10^28)，最小值 (**minvalue**，默认1)，每次自增值 (**increment by**)、是否使用Cache (**cache | nocache**)、是否保证按请求顺序获得序列值 (**order**) 以及是否循环 (**cycle | nocycle**) 等属性

语法兼容

PG 序列 默认的序列的最大值是： 9223372036854775807

```
create sequence test3_seq
start with 1
minvalue 1
increment by 1
maxvalue 100
cache 1
cycle;
```

PG的序列除了以下三点，别的属性和Oracle都是相同的。

①order 属性：Oracle有order属性，PG则没有order属性。

②cache属性：Oracle的cache的最小值是2。小于最小值就是nocache。而PG没有nocache值，cache的最小值和默认值都等于1，就是不使用缓存。

③maxvalue属性：Oracle序列的 maxvalue 的最大值和默认值都是10^28，并且当指定值超过10^28，且没超过number类型最大值 (10^126) 的时候，都自动采用10^28。而PG的序列的 maxvalue 的最大值和默认值都是BIGINT类型的最大值9223372036854775807.

语法兼容

操作序列的函数

Oracle操作序列的函数，主要是nextval， currval。用于取得下一个值和当前值。没有重置序列值的方法。

```
select test1_seq.nextval from dual;
select test1_seq.currval from dual;
```

PostgreSQL中操作序列的函数，除了nextval, currval，还提供了重置序列值的方法setval，以及一个lastval（返回会话内上一次调用序列的序列值）。

```
select nextval('test1_seq');
select setval(‘test1_seq’, 1) ;
select currval('test1_seq');
select lastval();
```

PostgreSQL提供了三种序列的数据类型，分别叫smallserial、serial、bigserial。它们三个实际上不是真正意义的数据类型，而是为该数字型字段生成一个序列，每当该字段数据增加的时候，都自动调用该序列的 nextval方法获得一个自动增大的、不重复的数字值

语法兼容

8. instr函数

PG中，集成了instr函数，但是必须使用四个参数。对用oracle两个参数的情况，可以使用position(substring in string) 函数来对应它。

```
SQL> select instr('helloll','ll') from dual;

INSTR('HELLOLL','LL')
---------------------
                    3

SQL> select instr('helloll','ll',1,1) from dual;

INSTR('HELLOLL','LL',1,1)
-------------------------
                        3

SQL> select instr('helloll','ll',1,2) from dual;

INSTR('HELLOLL','LL',1,2)
-------------------------
                        6
```

oracle

```
highgo=# select position('ll' in 'helloll');
 position
----------
        3
(1 行记录)

highgo=# select instr('helloll','ll',1,1);
 instr
-------
     3
(1 行记录)

highgo=# select instr('helloll','ll');
ERROR:  42883: function instr(unknown, unknown) d
第1行select instr('helloll','ll');
           ^
提示:  No function matches the given name and arg
add explicit type casts.
highgo=# select instr('helloll','ll',1,2);
 instr
-------
     6
```

postgresql

语法兼容

9. 伪列

Oracle

```
select rowid from student;
```

Postgresql

```
select ctid from student;
--PG中没有rowid伪列 有ctid实现类似rowid的作用
```

```
select rownum from score;
```

```
select row_number() over() as rownum from score;
--PG中没有rownum伪列，可以通过函数得到
```

10、字符串连接函数concat

Oracle的concat函数类似于字符串连接符（||），但只能够连接两个参数。参数需要是字符串类型，或者可以自动转换成字符串类型。

PG中也内置了这个方法。需要注意的是，Oracle的concat，如果两个参数都是null, 则结果是null。而PG中，如果两个参数都是null，则 结果是空字符串('')。因为PG的concat方法内部对于参数做了coalesce(null, '')处理。

语法兼容

11. 聚合函数

聚合函数是标准SQL语言中的函数，主要包括求计数(count)，求最大值(max)，求最小值(min)，求和(sum)和求平均数(avg)等几个。Oracle和PG都支持聚合函数。但是对求和(sum)和求平均数(avg)而言，Oracle和PG有一些不同。

示例：

在 oracle 下参数类型可以为 text

```
SQL> select sum(‘1’) from dual;
SUM(‘1’)
---------------
1
```

在 PG 只允许 SMALLINT、 INT、 BIGINT、 REAL、 DOUBLE PRECISION、 NUMERIC、 INTERVAL数据类型，或者通过::转意为以上类型。否则会有如下报错：

```
highgo=# select sum('1') -- select sum('1'::int); select sum(cast('1' as int));
错误: 函数 sum(unknown) 不是唯一的
第 1 行 select sum('1');
^
提示: 无法选择最佳候选函数. 你也许需要增加明确的类型转换.
```

语法兼容

12. SYSDATE、CURRENT_TIMESTAMP

- Oracle的SYSDATE是去当前的日期的一个函数。返回值只有年月日。取得当前时间戳的函数是CURRENT_TIMESTAMP，返回当前的时间戳。
- PG中，可以使用CURRENT_DATE来对应SYSDATE。至于CURRENT_TIMESTAMP，PG也支持。也可以使用now()来对应它。

13. 列别名为关键字

- Oracle中比如name，type这样的关键字可以直接作为列的别名，

  比如：select xx name from t

- PG中需要加as，

  比如select xx as name from t

递归迁移

oracle递归大致写法：

```
select * from some_table [where 条件1] connect by [条件2] start with [条件3];
```

其中 connect by 与 start with 语句摆放的先后顺序不影响查询的结果

示例：假如有如下结构的表：test(tid int,pid int)，其中pid保存父记录的id。

```
select tid from test start with tid in(1,2,3) connect by prior tid = pid;
```

connect by prior pid=tid 表示采用自上而下的搜索方式（先找父节点然后找子节点）

connect by pid=prior tid表示采用自下而上的搜索方式（先找叶子节点然后找父节点）。

递归迁移

PG 下需要通过 WITH RECURSIVE r AS()的方法来实现

```
with recursive r as (
  select tid from test where tid in(1,2,3)
  union
select t.tid from test t,r where t.pid = r.tid)
select * from r;
```

存储过程

项目	Oracle	PG

存储过程	create or replace procedure
	test_pro1(str varchar) as
	begin
	  Dbms_output.Put_line('打印信息'||str);
	end;
	call test_pro1('成功');
		create or replace function
		test_pro1(str varchar) returns void
		as
		$$
		begin
		  raise notice '打印信息%',str; --打印信息语句
		end;
		$$ language plpgsql;
		select test_pro1('成功');
		--与之对应的是函数 必须有returns，如果没有可以指定为void

触发器

项目	Oracle	PG

触发器	create or replace trigger
	score_trigger
	after insert
	on score
	for each row
	  begin
	  dbms_output.put_line('成功插入记录');
	  end;
		create function tri_fun() returns trigger
		as $$
		begin
		  raise notice '成功插入数据！！';
		end;
		$$ language plpgsql;
		create trigger score_trigger
		after insert on score
		for each row
		execute procedure tri_fun;
		--触发器不能直接创建，必须先建立触发函数，然后创建触发器调用触发函数

THANKS

Email:liuyanhong@highgo.com