# plsql2plpgsql 转换补充

目前LightDB使用java工具，兼容ora2pg的函数、存储过程转换功能，完成plsql2plpgsql的转换。

转换过程中，由于oracle语法并不兼容，导致某些语法、函数无法进行一对一进行转换。本文就是对这些问题进行一些汇总，将一些需要手动进行转换的语法、函数一一进行列举，方便用户进行转换。

## 1. merge into

由于LightDB-A目前不支持oracle merge into语法，因此需要进行手动修改，确保插入或者更新的准确性。

### 转换方案

就是通过LightDB-A CTE语法结构将oracle merge into的作用进行替代。

update target t set t.year = s.year from source s where t.id = s.aid 负责对满足t.id = s.aid条件的表进行更新，并且通过returning s.\* 返回满足t.id = s.aid条件的元组。首先判断where not exists (select 1 from up u where s.id = u.id)是否存在满足s.id = u.id条件的元组，如不存在则插入，存在则不插入。

Oracle merge into语法：

merge into target t using source s on (t.id = s.aid)when matched thenupdate set t.year = s.yearwhen not matched theninsert values(s.aid, s.name, s.year);

LightDB-A CTE语法：

with up as(

update target t set t.year = s.year from source s where t.id = s.aid returning s.\*) --update

insert into target(id,name,year) select aid, name, year from source s --insert

where not exists (

select 1 from up u

where s.id = u.id);

## 2. connect by

由于LightDB-A目前不支持oracle connect by语法，因此需要进行手动修改，确保插入或者更新的准确性。

### 转换方案

就是通过LightDB-A CTE语法结构将oracle connect by的作用进行替代。

主要有以下三种情况：

（1）prior在表达式左边，例如prior id = parent；

Oracle connect by语法：

select \* from sr\_menu

start with id = 1

connect by prior id = parent;

LightDB-A CTE语法：

with recursive cte\_connect\_by as (

select s.\* from sr\_menu s where id = 1

union all

select s.\* from cte\_connect\_by r inner join sr\_menu s on r.id = s.parent

)

select \* from cte\_connect\_by

;

（2）prior在表达式右边，例如prior id = parent；

Oracle connect by语法：

select \* from sr\_menu

start with id = 1

connect by id = prior parent;

LightDB-A CTE语法：

with recursive cte\_connect\_by as (

select s.\* from sr\_menu s where id = 1

union all

select s.\* from cte\_connect\_by r inner join sr\_menu s on s.id = r.parent

)

select \* from cte\_connect\_by ;

（3）加connect\_by\_root

Oracle connect by语法：

select connect\_by\_root id, parent, title from sr\_menu

start with id = 1

connect by prior id = parent

LightDB-A CTE语法：

with recursive cte\_connect\_by(id, connect\_by\_root\_id, parent, title) as (

select id, id as connect\_by\_root\_id, parent, title from sr\_menu s where id = 1

union all

select s.id, r.connect\_by\_root\_id, s.parent, s.title from cte\_connect\_by r inner join sr\_menu s on r.id = s.parent

)

select id, connect\_by\_root\_id, parent, title from cte\_connect\_by ;

## 3. rownum

Rownum是oracle的伪列，多用于进行分页查询。LightDB-A未兼容该伪列。

### 转换方案

LightDB-A使用row\_number() over () 分析函数替代oracle rownum伪列，并且使用rownum作为别名。

## 4. table() 函数

Oracle table函数将嵌套表的结果，转换成表格形式，显示出结果。

### 转换方案

LightDB-A不支持创建table函数，因为table在LightDB-A作为保留关键字，不能被作为函数名称使用。LightDB-A存在函数unnest()，功能与Oracle相同，存在调用table的地方使用unnest替换。

## 5. exception类型

在Oracle plsql中支持用户自定义异常，并对异常进行初始化，赋错误码sqlcode。Oracle 使用方法如下：

errrecord exception；

exception\_int(errrecord, -30001);

之后使用raise errrecoed；抛出错误。

postgres plpgsql不支持oracle以相同的方式进行自定义异常，因此在迁移的过程中我们需要对这部分内容进行手动修改，以达到相同的目的。

### 转换方案

LightDB-A支持直接使用raise exception抛出异常，因此，我们在处理时通常转换为以下形式：

raise EXCEPTION '(%) An attached analytic workspace is blocking this command', 'errrecode';

errrecode为使用exception声明的变量。

## 6. insert all

Oracle支持复合相同条件的元组，同时插入到不同的表中。目前LightDB-A不支持 insert all语法。

### 转换方案

实际使用过程中，将需要插入数据的表单独按照相同的条件进行插入。

如下面方式进行修改：

Oracle insert all 语法：

insert all

into t1(object\_name,object\_id)

into t2(object\_name,object\_id)

select \* from t;

LightDB-A insert语法：

with temp as (select \* from t;),

Ins1 as (insert into t1(object\_name,object\_id) select \* from temp)

insert into t2(object\_name,object\_id) select \* from temp;

## 7. Oracle（+）

Oracle支持（+）表示连接。目前LightDB-A不支持 oracle (+)语法。

### 转换方案

实际使用过程中，使用left join 或者 right join进行替换。

如下面方式进行修改：

Oracle oracle (+) 语法：

Select a.Pro\_lightdb\_version\_number,

b.Pro\_em\_release\_date,

c.Pro\_O45\_publisher，

from hs\_lightDB a, hs\_em b, hs\_O45 c

where 1 = 1

and a.Pro\_lightdb\_version\_number(+) = c.Pro\_O45\_version\_number

and b.Pro\_em\_publisher = c. Pro\_O45\_publisher(+);

LightDB-A oracle (+) 语法：

select a.Pro\_lightdb\_version\_number,

b.Pro\_em\_release\_date,

c.Pro\_O45\_publisher

from hs\_lightDB a

right join

hs\_O45 c on a.Pro\_lightdb\_version\_number = c.Pro\_O45\_version\_number

right join

hs\_em b on b.Pro\_em\_publisher = c.Pro\_O45\_publisher

where 1 = 1

**order by a.Pro\_lightdb\_version\_number asc;**

转换时需要加上排序条件。

## 8. SQLCODE

Oracle支持使用SQLCODE返回错误码，放回类型为int，且SQLCODE使用场景不局限于exception when结构体内，初始化值为0。目前LightDB-A不支持 SQLCODE，但提供SQLSTATE作为错误码的返回值，其类型为字符类型，并且使用范围局限于exception when结构体内。

### 转换方案

实际使用过程中，使用SQLSTATE 替换 SQLCODE。

## 9. MINUS

Oracle支持使用MINUS去做结果集的减法。A minus B就意味着将结果集A去除结果集B中所包含的所有记录后的结果，即在A中存在，而在B中不存在的记录。Oracle的minus是按列进行比较的，所以A能够minus B的前提条件是结果集A和结果集B需要有相同的列数，且相同列索引的列具有相同的数据类型。Oracle会对minus后的结果集进行去重，即如果A中原本多条相同的记录数在进行A minus B后将会只剩一条对应的记录。

目前LightDB-A不支持 MINUS，但提供EXCEPT支持该功能，

### 转换方案

实际使用过程中，使用EXCEPT替换 MINUS。

## 10. sys\_guid()函数

Oracle支持使用sys\_guid()去产生并返回一个全球唯一的标识符。

目前LightDB-A不支持 sys\_guid()，插件uuid-ossp提供函数uuid\_generate\_v4()，

### 转换方案

实际使用过程中，使用uuid\_generate\_v4替换 sys\_guid。

## 11. q转义字符

Oracle支持使用q转义字符进行字符转义。目前LightDB-A不支持 q转义字符。

### 转换方案

实际使用过程中，使用E’’替换 q’[]’。

Oracle q转义字符：

select q'[this isn't a good news $$$$]' from dual;

LightDB-A q转义字符：

select e'this isn\'t a good news $$$$';

可以看到除不可见字符，LightDB-A需要使用\进行转义外，单引号(‘)本身也需要使用\进行转换。

## 12. 嵌套表

Oracle支持使用嵌套表，常见的使用场景有2种：

（1）在存储过程使用；

（2）在ddl种使用，作为类型使用。

### 转换方案

Oracle 嵌套表单一类型：

CREATE OR REPLACE TYPE type1 AS TABLE OF VARCHAR2(30);

/

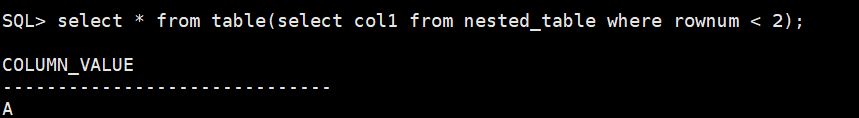
CREATE TABLE nested\_table (id NUMBER, col1 type1)

NESTED TABLE col1 STORE AS col1\_tab;

INSERT INTO nested\_table VALUES (1, type1('A'));

INSERT INTO nested\_table VALUES (2, type1('B', 'C'));

INSERT INTO nested\_table VALUES (3, type1('D', 'E', 'F'));



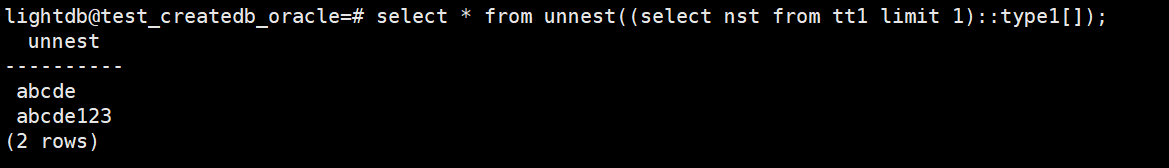
LightDB-A 兼容嵌套表单一类型：

create domain type1 as varchar2(30);

--数组

create table tt1 (id int, info text, nst type1[]);

insert into tt1 values (1,'test',array['abcde'::type1, 'abcde123'::type1]);



Oracle 嵌套表复合类型：

--创建对象

CREATE TYPE animal\_ty AS OBJECT (

breed varchar2(25),

name varchar2(25),

birthdate date);

/

--创建类型

CREATE TYPE animals\_nt as table of animal\_ty;

/

--创建表

create table breeder

(breedername varchar2(25),

animals animals\_nt)

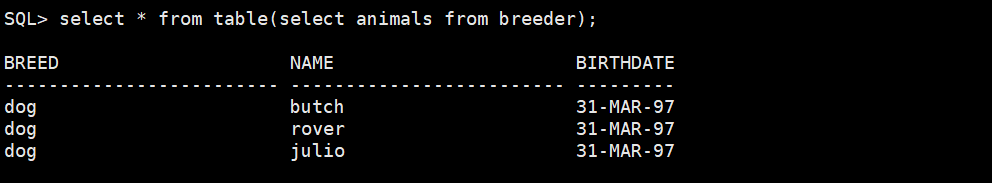
nested table animals store as animals\_nt\_tab;

--插入数据

insert into breeder  
values('mary', animals\_nt(animal\_ty('dog','butch','31-MAR-97'),  
animal\_ty('dog','rover','31-MAR-97'),  
animal\_ty('dog','julio','31-MAR-97')));

--查询

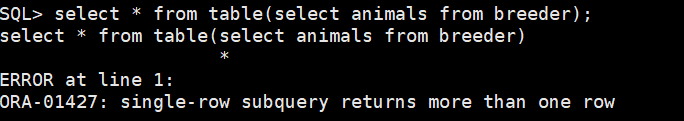
select \* from table(select animals from breeder);



再次插入：

insert into breeder  
values('mary', animals\_nt(animal\_ty('dog','butch','31-MAR-97'),  
animal\_ty('dog','rover','31-MAR-97'),  
animal\_ty('dog','julio','31-MAR-97')));

报错：single-row subquery returns more than one row



LightDB-A使用数组加上复合类型的方法实现嵌套表

--复合类型

create type type1 as (c1 int, c2 int, c3 text, c4 timestamp);

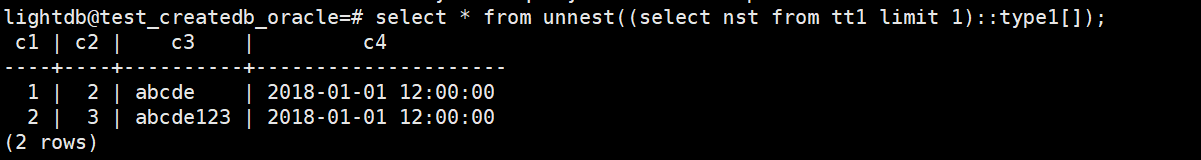
--复合类型数组

create table tt1 (id int, info text, nst type1[]);

--插入数组

insert into tt1 values (1,'test',array['(1,2,"abcde","2018-01-01 12:00:00")'::type1, '(2,3,"abcde123","2018-01-01 12:00:00")'::type1]);

select \* from unnest((select nst from tt1 limit 1)::type1[]);



查询多列报错，与oracle一致，只能查询单列。



## 13. 嵌套表 extend方法

Oracle支持使用extend方法，扩充嵌套表的。目前LightDB-A不支持嵌套表 extend方法。

### 转换方案

由12.嵌套表的转换方案可知，LightDB-A通过数组的形式来兼容oracle嵌套表的，且没有设置数组长度，在内存满足的情况下，数据总能存储到数组中，因此不需要使用extend进行扩容。删掉嵌套表 extend方法。

## 14. 嵌套表 count方法

Oracle支持使用count方法，记录当前嵌套表的长度，常用来插入最新的行。目前LightDB-A不支持嵌套表 count方法。

### 转换方案

Oracle 嵌套表 count方法：

--创建嵌套表

CREATE OR REPLACE TYPE TP\_STRING is table of VARCHAR2(1000);

/

--创建函数

CREATE OR REPLACE FUNCTION F\_PUB\_GET\_STRLIST

(

p\_inlist\_string char DEFAULT ' ',

p\_separator\_str char DEFAULT ','

)

return tp\_string

as

v\_inlist\_string char(256) := nvl(trim(p\_inlist\_string),' '); --in字符串列表

v\_separator\_str char(256) := nvl(trim(p\_separator\_str),','); --分隔符

v\_sstr char(256);

v\_data tp\_string;

v\_i int;

v\_j int;

v\_len int;

v\_len2 int;

begin

v\_sstr := ' ' ;

v\_data := tp\_string() ;

v\_i := 1 ;

v\_j := 0 ;

v\_len := 0 ;

v\_len2 := 0 ;

--20150807 zhangxd modify for 支持分隔符以变量的形式传入

v\_len := length(v\_inlist\_string);

v\_len2 := length(v\_separator\_str);

while v\_i <= v\_len + 1 loop

v\_j := instr(v\_inlist\_string, v\_separator\_str, v\_i);

if v\_j = 0 then

v\_j := v\_len + 1;

end if;

v\_sstr := substr(v\_inlist\_string, v\_i, v\_j - v\_i);

v\_i := v\_j + v\_len2;

dbms\_output.put\_line(v\_sstr);

v\_data.extend;

v\_data(v\_data.count) := v\_sstr;

end loop;

return v\_data;

end f\_pub\_get\_strlist;

/

;

LightDB-A嵌套表 count方法：

--创建数组

create domain TP\_STRING as VARCHAR(1000)[];

--创建函数

CREATE OR REPLACE FUNCTION F\_PUB\_GET\_STRLIST

(

p\_inlist\_string char(256) DEFAULT ' ', --in字符串列表

p\_separator\_str char(256) DEFAULT ',' --分隔符

)

returns tp\_string

as $$

declare

v\_inlist\_string char(256) = nvl(trim(p\_inlist\_string),' '); --in字符串列表

v\_separator\_str char(256) := nvl(trim(p\_separator\_str),','); --分隔符

v\_sstr char(256);

v\_data tp\_string;

v\_i int;

v\_j int;

v\_len int;

v\_len2 int;

begin

v\_sstr := ' ' ;

v\_i := 1 ;

v\_j := 0 ;

v\_len := 0 ;

v\_len2 := 0 ;

--20150807 zhangxd modify for 支持分隔符以变量的形式传入

v\_len := length(v\_inlist\_string);

v\_len2 := length(v\_separator\_str);

while v\_i <= v\_len + 1 loop

v\_j := instr(v\_inlist\_string, v\_separator\_str, v\_i);

if v\_j = 0 then

v\_j := v\_len + 1;

end if;

v\_sstr := substr(v\_inlist\_string, v\_i, v\_j - v\_i);

v\_i := v\_j + v\_len2;

raise notice '%',v\_sstr;

v\_data = array\_append(v\_data, v\_sstr::VARCHAR);

raise notice '%',v\_data[1];

end loop;

return v\_data;

end;

$$ language plpgsql

;

使用表达式v\_data = array\_append(v\_data, v\_sstr::VARCHAR);去替换v\_data(v\_data.count) := v\_sstr;

注意使用array\_append函数时，需要保证参数左右两边基础类型相同，可以使用::去进行强制转换。如定义数组时使用的varchar，array\_append右边参数也为varchar.

## 15. unpivot行转列

Oracle支持使用unpivot行转列。目前LightDB-A不支持unpivot行转列。

### 转换方案

创建表：

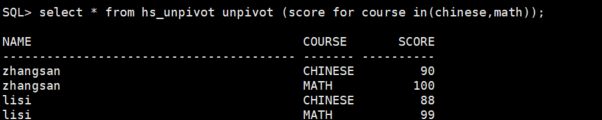
create table hs\_unpivot(name varchar(40),chinese int,math int);

insert into hs\_unpivot values('zhangsan',90,100);

insert into hs\_unpivot values('lisi',88,99);

Oracle unpivot行转列：

select \* from hs\_unpivot unpivot (score for course in(chinese,math));



LightDB-A unpivot行转列：

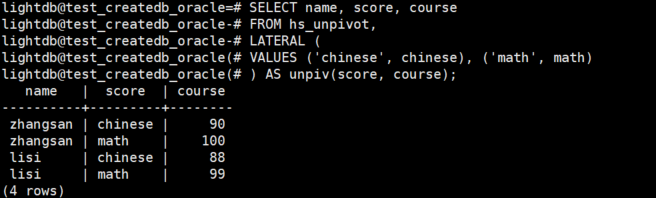
SELECT name, score, course

FROM hs\_unpivot,

LATERAL (

VALUES ('chinese', chinese), ('math', math)

) AS unpiv(score, course);



## 16. pivot列转行

Oracle支持使用pivot列转行。目前LightDB-A不支持pivot列转行。

### 转换方案

创建表：

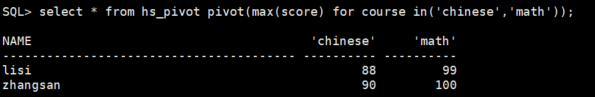
create table hs\_pivot(name varchar(40),chinese int,math int);

insert into hs\_pivot values('zhangsan',90,100);

insert into hs\_pivot values('lisi',88,99);

Oracle pivot列转行：

select \* from hs\_pivot pivot (sum(score) for course in('chinese','math'));;



LightDB-A pivot列转行：

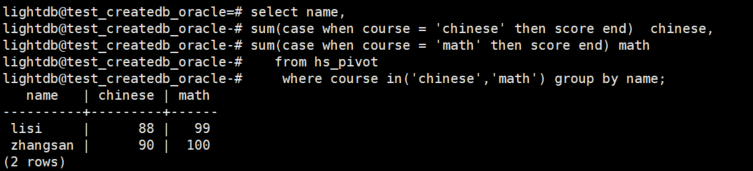
select name,

sum(case when course = 'chinese' then score end) chinese,

sum(case when course = 'math' then score end) math

from hs\_pivot

where course in('chinese','math') group by name;



## 17. goto

Oracle支持使用goto，跳转到指定的标签。目前LightDB-A不支持goto跳转到指定的标签。

### 转换方案

Oracle goto:

create or replace function hs\_goto(i int) return int as

ind int :=0;

begin

ind := i +1;

if i = 1 then

goto flag;

end if;

ind := i+2;

<<flag>>

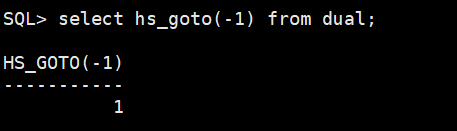
dbms\_output.put\_line(ind);

return ind;

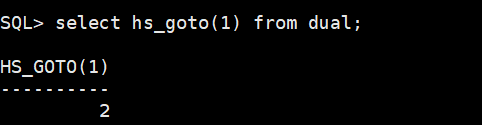
end;

/

select hs\_goto(-1) from dual;



select hs\_goto(1) from dual;



Postgres:

create or replace function hs\_goto(i int) return int as

goto\_flag int := 0;

ind int := 0;

begin

ind := i +1;

if i = 1 then

goto\_flag := 1;

end if;

if goto\_flag != 1 then

ind := i+2;

end if;

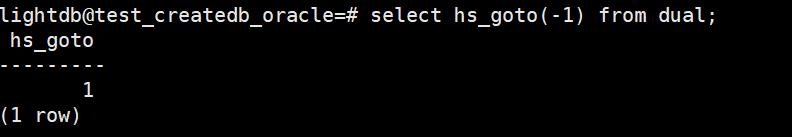
dbms\_output.put\_line(ind);

return ind;

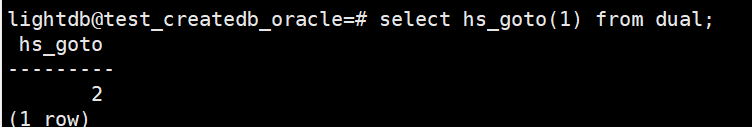
end;

/

select hs\_goto(-1) from dual;



select hs\_goto(1) from dual;



解决方案思路：使用变量goto\_flag去替换goto语句，goto到标签范围内的代码，使用if条件进行判断。!goto\_flag为真，则执行该范围内的代码。