◆ collections (index by table & nested table & bulk collect)

در ادامه مباحث مطرح شده با موضوع PL/SQL در این بخش به معرفی نوع های داده ای ترکیبی (composit data type) ها میپردازم

و در این بخش به معرفی record و در بخش بعد به معرفی collection خواهم پرداخت

- نوع داده ای ترکیبی چیست و چه کاربردی دارد ؟
- میتوان گفت که نوع داده ای ترکیبی از کنار هم قرار دادن یک یا چند نوع داده ای شـــمارش پذیر (scalar data type)ها مانند date ، number، varchar2 ایجاد میشود که به منظور تعریف یک ساختار جهت نگهداری داده های مرتبط با هم تعریف میشـود ، در واقع انواع داده ای ترکیبی شـبیه به سـاختار های داده ای زبان های برنامه نویسی مانند class یا structure میباشند.

چند نوع داده ای ترکیبی در PL/SQL وجود دارد ؟

- **دو نوع داده ای ترکیبی در PL/SQL وجود دارد که شامل**

1. **record**
   - ➞ **user-defined record**
   - ➞ **% rowtype**
2. **collection**
   - ➞**index by table**
   - ➞ **nested table**
   - ➞ **varray**

- در چه مواقعی باید از record و در چه مواقعی باید از collection استفاده کنیم ؟
- هرگاه نیاز داشـــتیم تا از نوع داده ای ترکیبی برای نگهداری یک ســـطر داده اســـتفاده کنیم میتوانیم از recordاستفاده کنیم
- هرگاه نیاز دا شتیم تا از نوع داده ای ترکیبی برای نگهداری بیش از یک رکورد ا ستفاده کنیم میتوانیم از collectionاستفاده کنیم

- در چه مواقعی باید از index by table استفاده کنیم ؟
- اگر میخواهیم مجموعه ای از رکوردها را در حافظه بارگذاری کرده و بر اساس کلیدی در داخل آن جستجو انجام دهیم باید از index by table استفاده کنیم
- چگونه و در کدام قسمت میتوانیم یک index by table تعریف کنیم؟
- برای تعریف index by table باید در قسمت declaration دو گام ذیل را طی کنیم

گام : 🅰 تعریف نوع داده ای با استفاده از کلمهtype

گام :🅱 تعریف متغیری از نوع داده ای تعریف شده گام🅰

```
declare
🅰 type <<type_name>> is table of
data_type | table_name.column_name % type|
table_name % rowtype | user_defined_record_name
index by pls_integer|binary_integer|varchar2(size);

🅱 variable_name <<type_name>>;
```

به نکات مربوط به index by table دقت نمایید☜☜

نکته۱: مجوعه هایی که از نوع index by table هستند شامل یک ستون کلید و ستون (های) داده است ، جستجو در این مجموعه ها بر اساس کلید تخصیص داده شده به هر رکورد انجام میشود

نکته۲: به منظور جستجوی بهینه در مجموعه هایی که از نوع index by table تعریف میشود داده های وارد شده بصورت اتوماتیک بر اساس کلید تعریف شده مرتب سازی میشوند

نکته۳: در صورت استفاده از collection ها میتوان از متدهای first ، last، count، prior، next و exists استفاده کرد

به مثال ذیل دقت نمایید 👇👇👇👇👇👇

```
declare

type ename is table of emp_info.last_name%type index by pls_integer;
v_ename ename;

begin
v_ename(-100) := 'ali';
v_ename(1) := 'aliReza';
v_ename(2) := 'mohammad';
v_ename(5) := 'mohammadReza';
v_ename(-150) := 'mohammadReza';
v_ename(7) := 'mohammadReza';
/* collection sort after insert*/
dbms_output.put_line(v_ename.first); /*get first index after sort*/
dbms_output.put_line(v_ename.last); /*get last index after sort*/
dbms_output.put_line(v_ename.count); /*get count of collection*/
dbms_output.put_line(v_ename.prior(5)); /*get prior index 5*/
dbms_output.put_line(v_ename.next(5)); /*get next index 5*/
dbms_output.put_line(v_ename(v_ename.first));

for i in v_ename.first .. v_ename.last loop
if v_ename.exists(i) then
dbms_output.put_line(v_ename(i));
end if;
end loop;

end;
```

- در چه مواقعی باید از nested table استفاده کنیم ؟
- اگر میخواهیم مجموعه ای از رکوردها را در حافظه بارگذاری کرده و از آن به عنوان یک ظرف برای پر کردن داده ها استفاده کنیم میتوانیم از nested table استفاده کنیم

چگونه و در کدام قسمت میتوانیم یک nested table تعریف کنیم؟

برای تعریف nested table باید در قسمت declaration دو گام ذیل را طی کنیم

گام : 🅰 تعریف نوع داده ای با استفاده از کلمهtype

گام :🅱 تعریف متغیری از نوع داده ای تعریف شده گام🅰

```
declare
🅰 type <<type_name>> is table of
data_type | table_name.column_name % type|
table_name % rowtype | user_defined_record_name;

🅱 variable_name <<type_name>>;
```

به نکات مربوط به nested table دقت نمایید👈👈

نکته۱: مجوعه هایی که از نوع nested table هستند شامل یک ستون کلید و ستون (های) داده است ، ستون کلید در واقع یک شمارنده است که از یک شمرده میشود

نکته۲: میتوانید از متدهای first ، last، count، prior، next و exists استفاده کرد

نکته۳: قبل از ا ستفاد از nested table باید آن را intialize کنید یا با ا ستفاده از متد extend سطر جدید برای آن در نظر بگیرید

به مثال ذیل دقت نمایید👇👇👇👇👇👇

```
decalare
type emp_coll_type is table of employees % rowtype;
v_emp_coll emp_coll_type := emp_coll_type();
begin
```

```
for r in (select * from employees e where e.department_id = 50) loop
v_emp_coll.extend();
v_emp_coll(v_emp_coll.last) := r;

end loop;
dbms_output.put_line(v_emp_coll.count);
dbms_output.put_line(v_emp_coll(43).first_name);

end;

/---------------------------------------/

decalare
type location_type is table of locations.city % type;
v_loc_type location_type;
begin
v_loc_type := location_type('tehran', 'mashhad', 'gazvin');

for i in 1 .. v_loc_type.count loop
dbms_output.put_line(v_loc_type(i));
end loop;

end;

/---------------------------------------/
```

- در چه مواقعی میتوانیم از bulk collect استفاده کنیم ؟
- با استفاده از این امکان میتوان مجموعه ای از رکوردها را یکجا در مجموعه ای بارگذاری کرد و همچنین میتوان مجموعه ای از دستورات DML را ایجاد کرده و بصورت دسته ای به پایگاه داده ارسال کرد

به مثال های ذیل دقت نمایید 👇👇👇👇👇👇

در مثال فوق به ازای هر باری که دستور fetch صدا زده میشود ۱۰ رکورد از داده های جدول employees در قرار داده میشود

```
decalare
type emp_type is table of employees % rowtype;
v_emp emp_type;
cursor c is
select * from employees;
begin

open c;
loop
fetch c bulk collect
into v_emp limit 10;

forall i in v_emp.first .. v_emp.last
insert into employees_1 values v_emp(i);

commit;

exit when c % notfound;
end loop;
end;

/---------------------------------------/
```

در مثال ذیل نیز مجموعه ای از دستورات update ساخته شده و بصورت دسته ای به پایگاه داده ارسال میشود

```
declare
type tp_id_table is table of
employees.employee_id % type index by binary_integer;

v_id_table tp_id_table;

begin
v_id_table(1) := 100;
v_id_table(2) := 1300;
v_id_table(3) := 102;
v_id_table(4) := 103;
```

```
forall i in v_id_table.first .. v_id_table.last
update employees emp
set emp.salary = emp.salary + 100
where emp.employee_id = v_id_table(i);

commit;

end;
```