Tehnici de programare cu baze de date

#12 PL/SQL
Pachete în PL/SQL

Adrian Runceanu
www.runceanu.ro/adrian

Curs 12

Pachete în PL/SQL

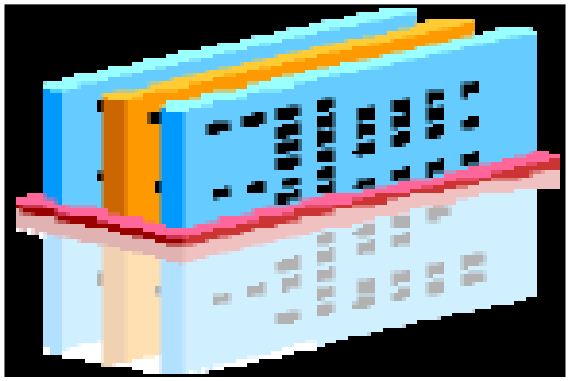
Cuprins

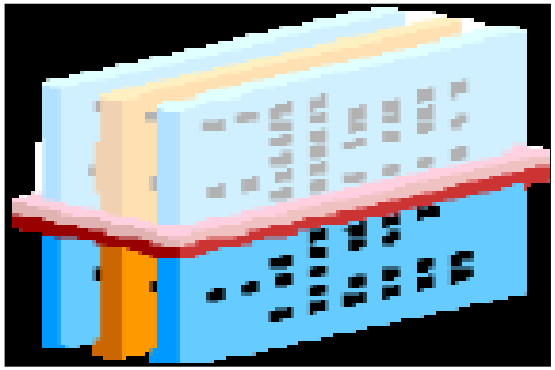
Pachete în PL/SQL

1. Crearea pachetelor
   1.1. Specificatia pachetului
   1.2. Corpul pachetului
2. Gestionarea conceptelor pachetului
3. Concepte avansate despre pachete

1. Crearea pachetelor

- Am studiat pana acum cum sa cream si cum sa folosim procedurile si functiile.
- Sa presupunem ca vrem sa cream cateva proceduri si/sau functii care sunt in relatie unele cu altele.
- O aplicatie poate sa le foloseasca fie pe toate, fie pe nici una dintre ele.





1. Crearea pachetelor

- Nu ar fi mai usor sa creati si sa administrati toate subprogramele ca un singur obiect al bazei de date: un pachet?
- In continuare vom studia ce este un pachet si care sunt componentele sale.
- De asemenea, vom studia crearea si utilizarea pachetelor.

1. Crearea pachetelor

Ce sunt pachetele PL/SQL?

- *Pachetele PL/SQL sunt containere care ne permit sa grupam impreuna subprograme, variabile, cursori si exceptii PL/SQL in relatie unele cu altele.*
- De exemplu, un pachet pentru Resurse Umane poate contine:
  - proceduri de angajare si concediere
  - functii pentru calcularea comisioanelor si bonusurilor
  - variabile folosite pentru scutirea de impozit

1. Crearea pachetelor

Componentele unui pachet PL/SQL

Un pachet este format din doua parti stocate separat in baza de date:

1. Specificatia pachetului – *interfata catre aplicatie*

2. Corpul pachetului – *contine codul executabil al subprogramelor care au fost declarate in specificatia pachetului*

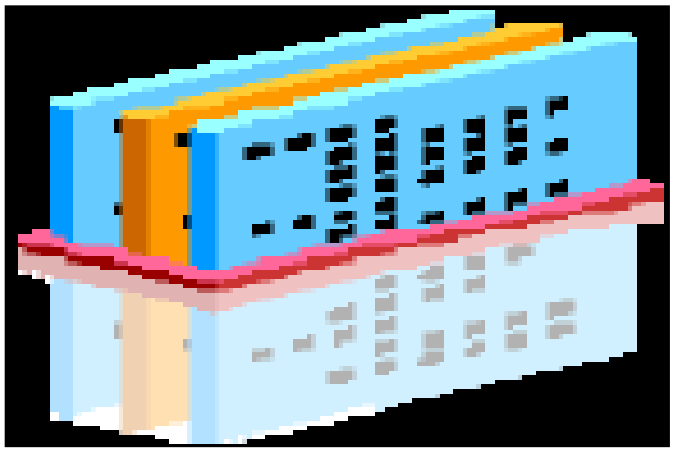
1. Crearea pachetelor

Componentele unui pachet PL/SQL

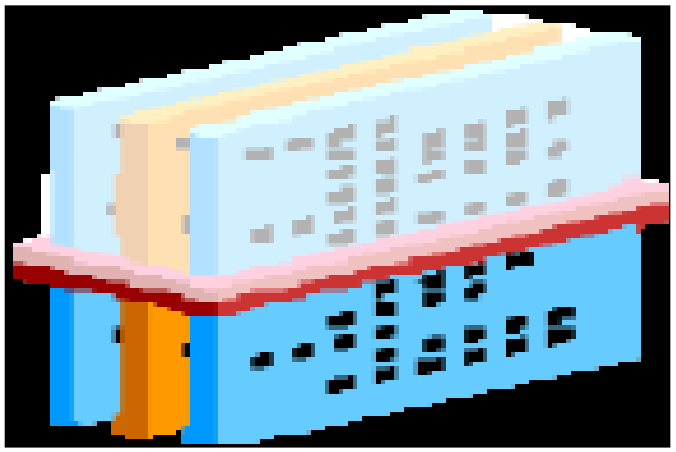
1. Specificatia pachetului – *interfata catre aplicatie*. Aceasta trebuie creata prima data. Se declara constructorii (proceduri, functii, variabile, etc.) care sunt vizibili in mediul apelant

2. Corpul pachetului – *contine codul executabil al subprogramelor care au fost declarate in specificatia pachetului*. De asemenea, poate contine propriile declaratii de variabile.

Specificatia pachetului



Corpul pachetului



- *Corpul detaliat al codului corpului corespunzator pachetului nu este vizibil in mediul apelant, acesta putand vedea doar specificatia.*
- Daca sunt necesare modificari in cod, corpul de instructiuni poate fi editat si recompilat fara a fi necesara editarea sau recompilarea specificatiei.
- Aceasta structura formata din doua parti este un exemplu al principiului programarii structurate numit ***încapsulare*.**

Cuprins

Pachete în PL/SQL

1. Crearea pachetelor
   1.1. Specificatia pachetului
   1.2. Corpul pachetului
2. Gestionarea conceptelor pachetului
3. Concepte avansate despre pachete

1.1. Specificatia pachetului

Sintaxa pentru crearea specificatiei pachetului

Pentru crearea pachetelor, declaram toti constructorii publici in specificatia pachetului.

```
CREATE [OR REPLACE] PACKAGE package_name
IS|AS
        public type and variable declarations
        public subprogram specifications
END [package_name];
```

- Optiunea OR REPLACE sterge si recreaza specificatia pachetului.
- Variabilele declarate in specificatia pachetului sunt initializate implicit cu NULL.
- Toti constructorii declarati in specificatia pachetului sunt vizibili utilizatorilor carora li s-a acordat privilegiul EXECUTE asupra pachetului.
- ***package_name*** – specifica un nume pentru pachet care trebuie sa fie unic printre obiectele din propria schema. Includerea denumirii pachetului dupa cuvantul cheie END este optionala.
- ***public type and variable declarations*** – declara variabilele, constantele, cursorii, exceptiile, tipurile si subtipurile definite de utilizator care sunt publice.
- ***public subprogram specifications*** – declara procedurile si/sau functiile publice ale pachetului.

- ***"Public"*** presupune ca, constructorii pachetului (variabile, procedurii, functii etc.) pot fi vazuti si executati dinafara pachetului.
- Toti constructorii declarati in specificatia pachetului sunt automat constructori publici.
- Specificatia pachetului ar trebui sa contina antete de proceduri si functii terminate prin punct si virgula, fara cuvantul cheie IS (sau AS) si blocul sau PL/SQL.

- *Implementarea (scrierea detaliata a codului) unei proceduri sau functii care este declarata in specificatia pachetului se face in corpul pachetului.*

Exemple:

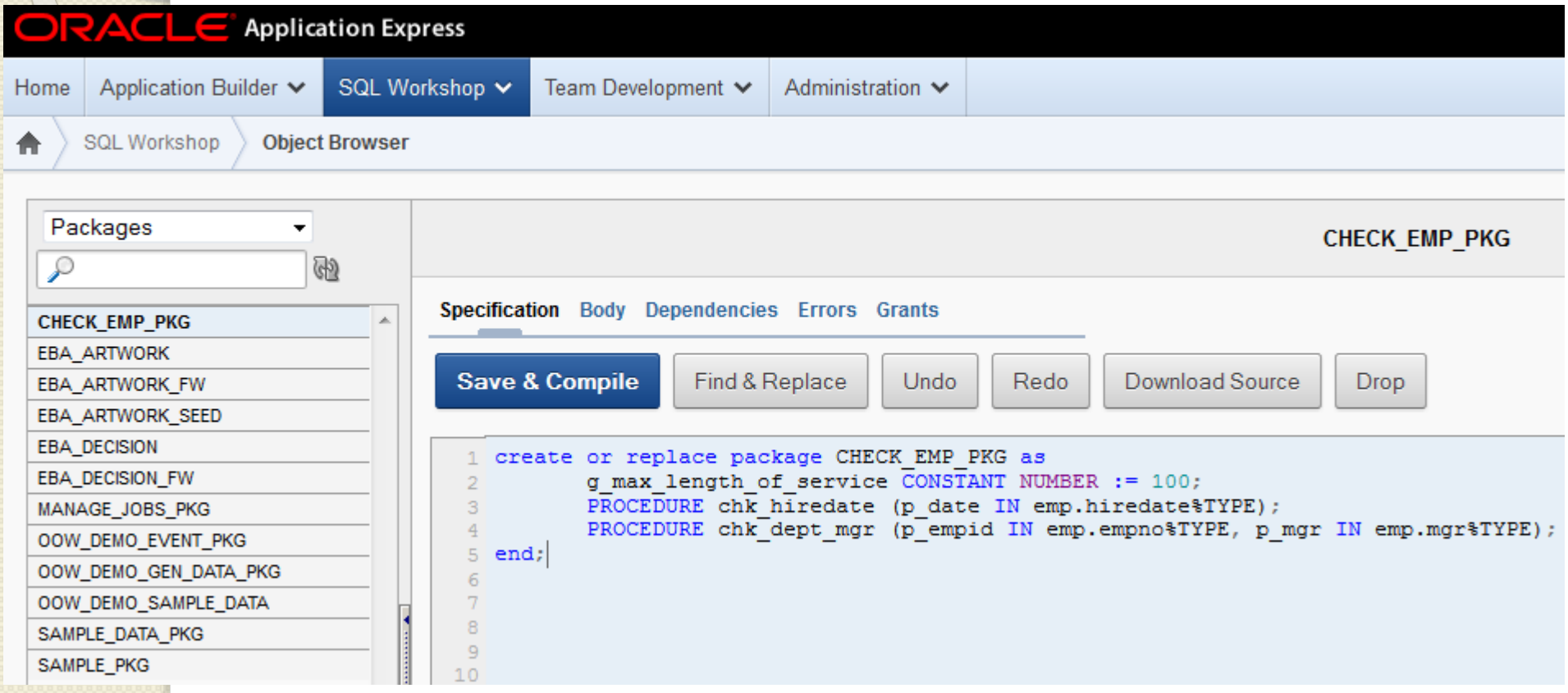
Specificarea pachetului ***check_emp_pkg***

```
create or replace package CHECK_EMP_PKG as
      g_max_length_of_service CONSTANT NUMBER
   := 100;
      PROCEDURE chk_hiredate (p_date IN
   emp.hiredate%TYPE);
      PROCEDURE chk_dept_mgr (p_empid IN
   emp.empno%TYPE, p_mgr IN emp.mgr%TYPE);
end;
```

g_max_length_of_service este o constanta declarata si initializata in specificatie

chk_hiredate si **chk_dept_mgr** sunt doua proceduri publice declarate in specificatie. Codul detaliat este scris in corpul de instructiuni al pachetului.

Exemple:
Specificarea pachetului ***check_emp_pkg***



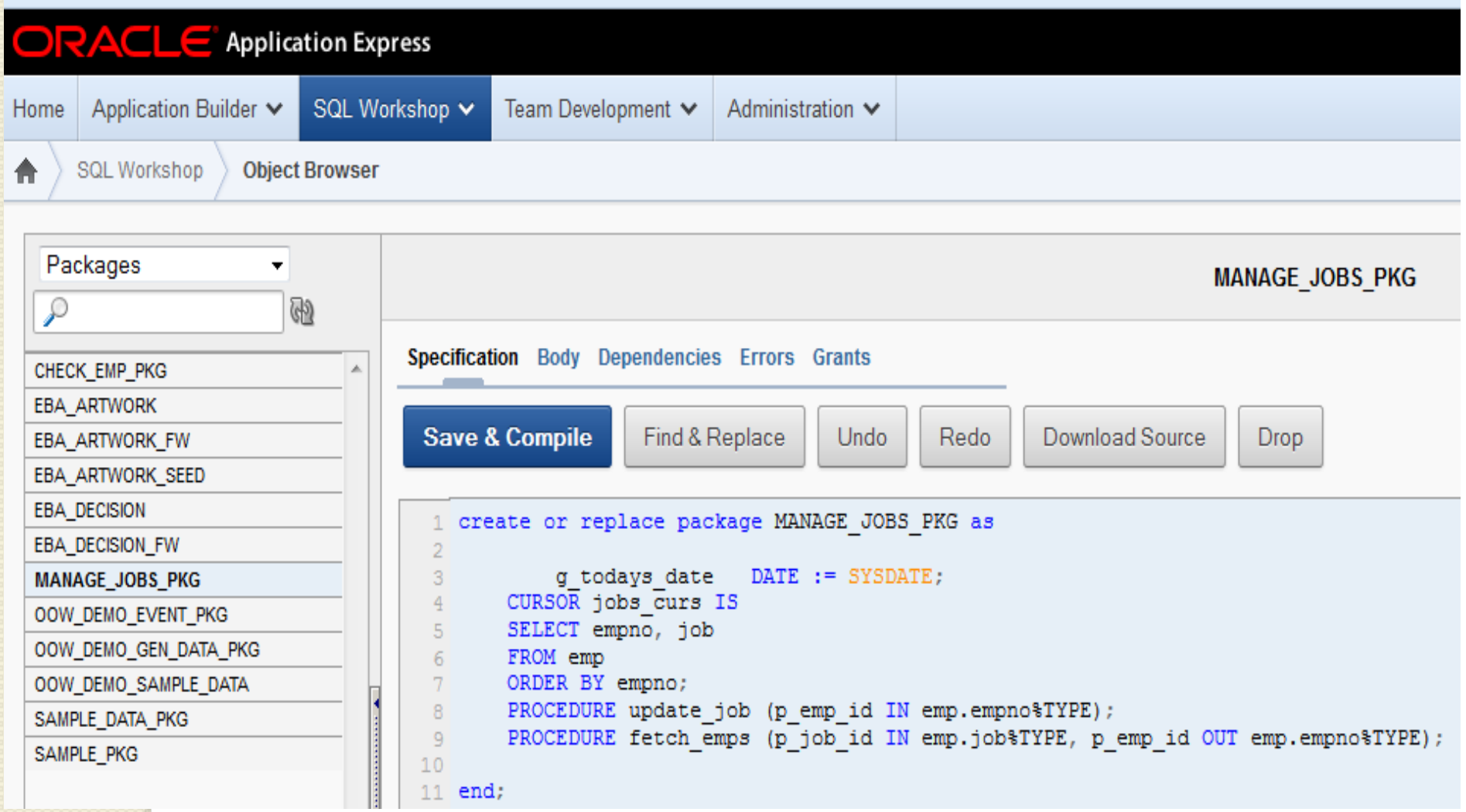
Sa ne reamintim ca un tip de variabila este si cursorul.

```
create or replace package MANAGE_JOBS_PKG as

    g_todays_date   DATE := SYSDATE;
    CURSOR jobs_curs IS
       SELECT empno, job
       FROM emp
       ORDER BY empno;
    PROCEDURE update_job (
       p_emp_id IN emp.empno%TYPE);
    PROCEDURE fetch_emps (
       p_job_id IN emp.job%TYPE,
       p_emp_id OUT emp.empno%TYPE);

end;
```

Specificarea pachetului **MANAGE_JOBS_PKG**



Cuprins

Pachete în PL/SQL

1. Crearea pachetelor
   1.1. Specificatia pachetului
   1.2. Corpul pachetului
2. Gestionarea conceptelor pachetului
3. Concepte avansate despre pachete

1.2. Corpul pachetului

Sintaxa pentru crearea corpului pachetului

```
CREATE [OR REPLACE] PACKAGE BODY
package_name
IS|AS
        private type and variable declarations
        subprogram bodies
[BEGIN initialization statements]
END [package_name];
```

1.2. Corpul pachetului

- Optiunea OR REPLACE sterge si recreaza corpul de instructiuni al pachetului.
- „Subprogram bodies” trebuie sa contina corpul tuturor subprogramelor declarate in specificatia pachetului
- ***package_name*** specifica un nume pentru pachet care trebuie sa fie acelasi ca si la specificatia pachetului. Folosirea denumirii pachetului dupa cuvantul cheie END este optionala.
- ***subprogram bodies*** – specifica implementarea completa (codul PL/SQL detaliat) al tuturor procedurilor si functiilor publice si/sau private.

1.2. Corpul pachetului

Atunci cand cream corpul unui pachet, trebuie sa facem urmatoarele:

- Specificam optiunea OR REPLACE pentru a suprascrie un corp de pachet existent
- Definim subprogramele intr-o ordine corespunzatoare.
- *Principiul de baza este ca trebuie sa declaram o variabila sau un subprogram inainte ca acestea sa fie referite de alte componente ale aceluiasi pachet.*
- Fiecare subprogram declarat in specificatia pachetului trebuie de asemenea sa fie inclus in corpul pachetului.

1.2. Corpul pachetului

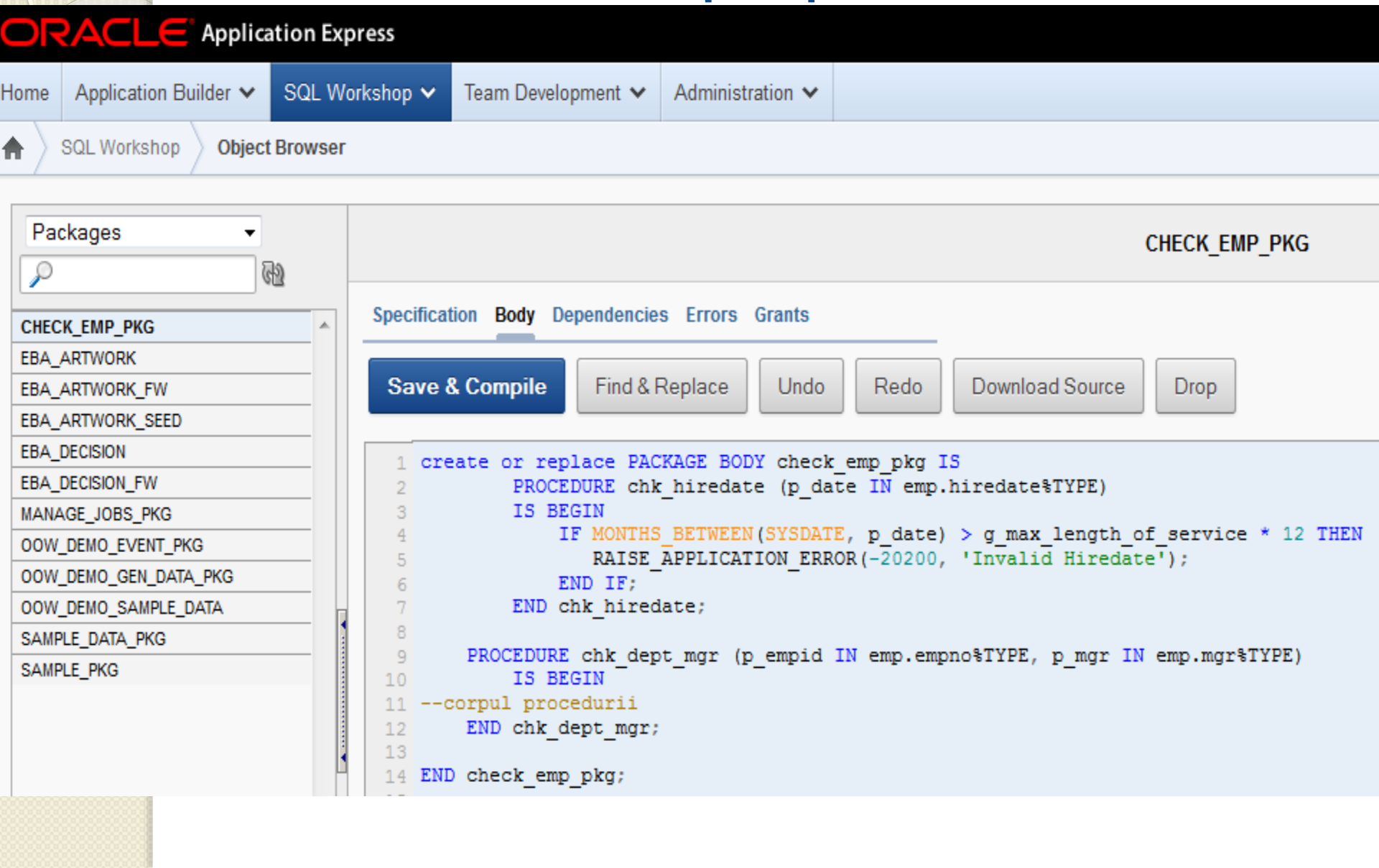
Exemplu de corp al unui pachet: ***check_emp_pkg***

```
CREATE OR REPLACE PACKAGE BODY
   check_emp_pkg IS
        PROCEDURE chk_hiredate
        (p_date IN emp.hiredate%TYPE)
        IS BEGIN
                IF MONTHS_BETWEEN(SYSDATE,
   p_date) > g_max_length_of_service * 12
                THEN
                RAISE_APPLICATION_ERROR(-
   20200, 'Invalid Hiredate');
                END IF;
        END chk_hiredate;
```

1.2. Corpul pachetului

```
    PROCEDURE chk_dept_mgr
        (p_empid IN emp.empno%TYPE,
        p_mgr IN emp.mgr%TYPE)
        IS BEGIN
...
    END chk_dept_mgr;
END check_emp_pkg;
```

1.2. Corpul pachetului



1.2. Corpul pachetului

Modificarea codului din corpul pachetului

- Sa presupunem ca vrem sa facem o modificare in procedura ***chk_hiredate***, de exemplu sa punem un mesaj de eroare diferit.
- Trebuie sa editam si sa recompilam corpul pachetului, dar nu trebuie sa recompilam specificatia.
- Sa nu uitam *ca specificatia poate exista fara corp, dar corpul nu poate exista fara specificatie*.
- Deoarece specificatia nu este recompilata, nu este necesar sa recompilati nici o aplicatie (sau subprogram PL/SQL) care deja apeleaza procedurile pachetului.

1.2. Corpul pachetului

Recompilarea corpului pachetului: check_emp_pkg

```
CREATE OR REPLACE PACKAGE BODY
   check_emp_pkg IS
   PROCEDURE chk_hiredate
         (p_date IN emp.hiredate%TYPE)
         IS BEGIN
         IF MONTHS_BETWEEN(SYSDATE, p_date) >
         g_max_length_of_service * 12 THEN
         RAISE_APPLICATION_ERROR(-20201,
   'Hiredate Too Old');
         END IF;
END chk_hiredate;
```

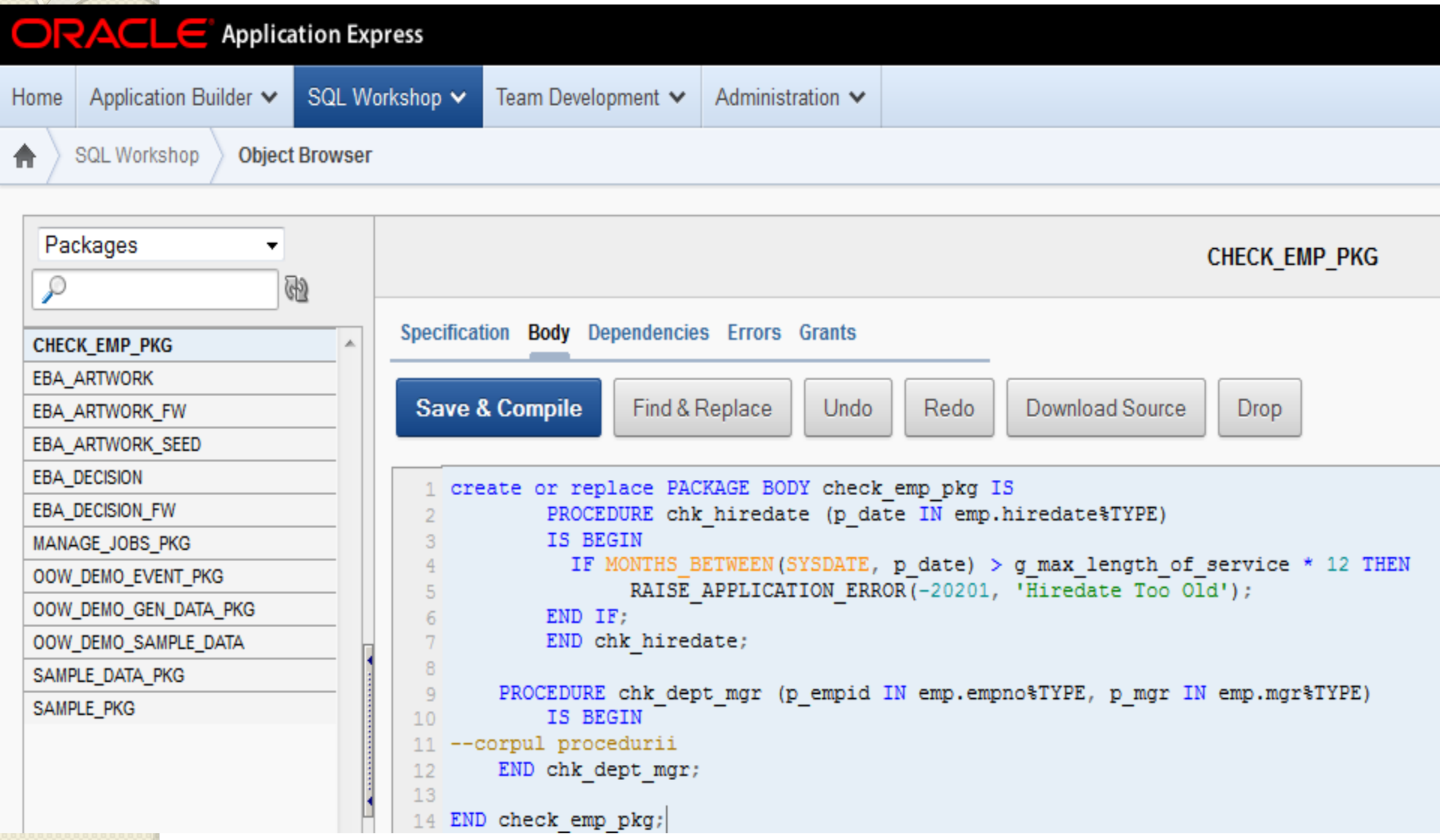
1.2. Corpul pachetului

```
PROCEDURE chk_dept_mgr
(p_empid IN      emp.empno%TYPE,
p_mgr IN   emp.mgr%TYPE)

IS BEGIN
        ...
END chk_dept_mgr;

END check_emp_pkg;
```

1.2. Corpul pachetului

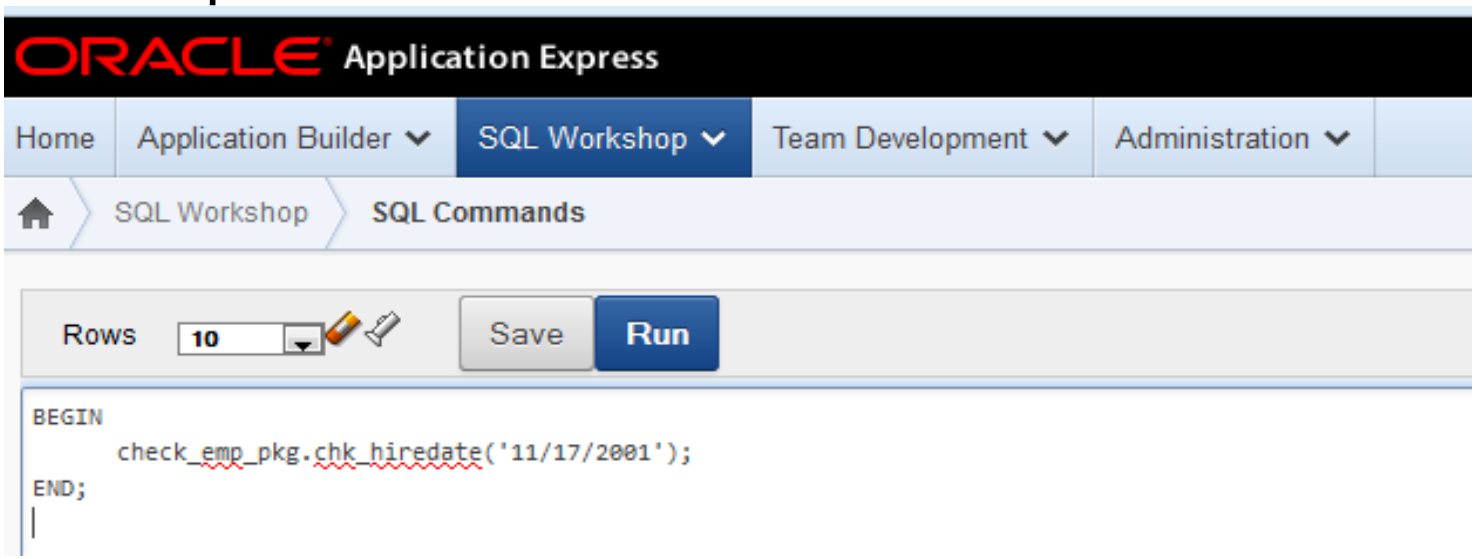


1.2. Corpul pachetului

Apelarea subprogramelor in pachete

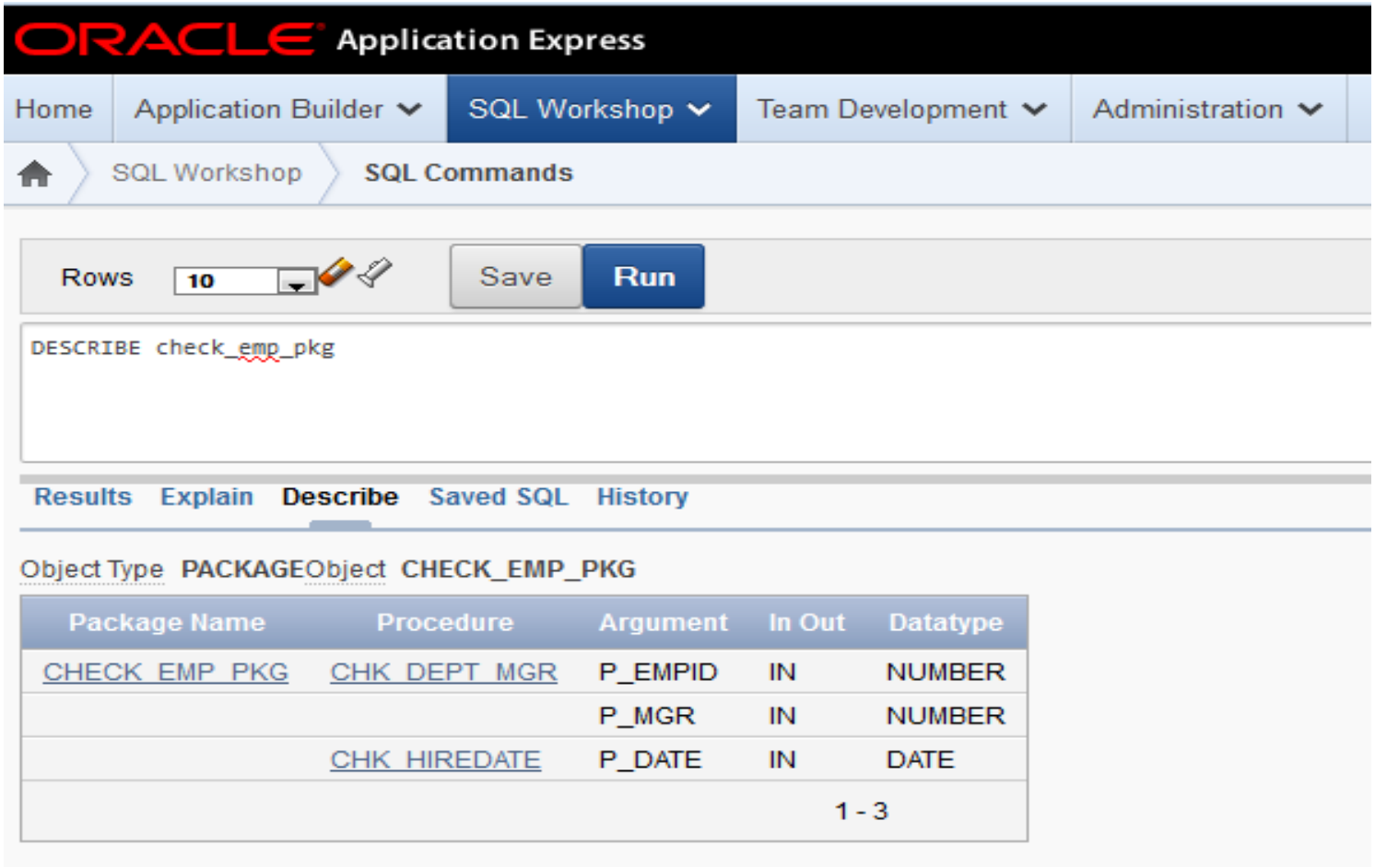
Procedurile si functiile din pachete se apeleaza in acelasi mod ca si subprogramele care nu fac parte din pachete, cu exceptia faptului ca *trebuie sa prefixam numele subprogramului cu numele pachetului urmat de caracterul punct*.

De exemplu:



- Dar daca se intampla sa uitati denumirile procedurilor sau ce parametri trebuie sa le transmiteti?
- In astfel de situatii puteti folosi DESCRIBE in acelasi mod in care folositi pentru o tabela sau pentru un view.

DESCRIBE check_emp_pkg



1.2. Corpul pachetului

Argumente pentru folosirea pachetelor

1. ***Modularitate*** – Programele si variabilele relationate pot fi grupate impreuna.
2. ***Ascunderea informatiei*** – doar declaratiile din specificatia pachetului sunt vizibile la apel. Dezvoltatorii de aplicatii nu au nevoie sa stie detaliile codului din corpul pachetului.
3. ***O intretinere mai usoara*** – Puteti modifica si recompila codul din corpul pachetului fara a fi necesara recompilarea specificatiei. De aceea, aplicatiile care deja folosesc pachetul nu este nevoie sa fie recompilate.

Cuprins

Pachete în PL/SQL

1. Crearea pachetelor
   1.1. Specificatia pachetului
   1.2. Corpul pachetului
2. Gestionarea conceptelor pachetului
3. Concepte avansate despre pachete

2. Gestionarea conceptelor pachetului

Vom studia:

- Crearea subprogramelor private in interiorul pachetului.
- Stergerea pachetelor
- Vizualizarea pachetelor in Data Dictionary
- Beneficiile suplimentare ale pachetelor

2. Gestionarea conceptelor pachetului

Componentele unui pachet PL/SQL

1. Componente publice – cele care sunt declarate in specificatia pachetului.

- Componentele publice pot fi apelate din orice mediu apelant, cu conditia ca utilizatorului sa-i fi fost acordat privilegiul EXECUTE asupra pachetului.

2. Componente private - sunt declarate doar in corpul pachetului si pot fi referite doar de alt constructor (public sau privat) din acelasi corp al pachetului.

- Componentele private pot referi componentele publice ale pachetului.

2. Gestionarea conceptelor pachetului

Vizibilitatea componentelor pachetului

- Vizibilitatea unei componente descrie daca acea componenta poate fi vazuta, referita si folosita de alte componente sau obiecte.
- Vizibilitatea unei componente depinde de locul unde este declarata.

2. Gestionarea conceptelor pachetului

Vizibilitatea componentelor pachetului

Puteti declara componentele in 3 locuri din interiorul pachetului:

1. Global, in specificatie – aceste componente sunt vizibile in tot corpul pachetului si de catre mediul apelant.

2. Local, in corpul pachetului, dar in afara oricarui subprogram – aceste componente sunt vizibile pe parcursul intregului corp al pachetului, dar nu si de mediul apelant

3. Local, in corpul pachetului, in interiorul unui anumit subprogram – aceste componente sunt vizibile doar in acel subprogram.

Toate componentele publice sunt globale, iar componentele private sunt locale.

Care este totusi diferenta intre public si global, intre privat si local? Intr-adevar nici una.

Dar se foloseste:

- public/privat cand vorbim despre proceduri si functii
- global/local cand este vorba despre alte componente (variabile, constante, cursori).

2. Gestionarea conceptelor pachetului

Vizibilitatea componentelor publice (globale)

Componentele declarate global sunt vizibile in interiorul si in exteriorul pachetului, astfel:

- O variabila globala declarata in specificatia pachetului poate fi referita si modificata in afara pachetului
- Un subprogram public declarat in specificatie poate fi apelat din surse de cod extern

2. Gestionarea conceptelor pachetului

Vizibilitatea componentelor private (locale)

Componentele locale sunt vizibile doar in structura in care sunt declarate, astfel:

- Variabilele locale declarate intr-un anumit subprogram pot fi referite doar de acel subprogram si nu sunt vizibile de catre componentele externe
- Variabilele locale care sunt declarate in corpul unui pachet pot fi referite de celelalte componente din acelasi corp al pachetului. Nu sunt vizibile nici unui subprogram sau obiect care sunt in afara pachetului.

Exemplu – specificatia pachetului *sal_pkg*

Sa presupunem sa avem o regula in domeniul de activitate ca salariul nici unui angajat sa nu creasca cu mai mult de 20% la un moment dat.

```
CREATE OR REPLACE PACKAGE sal_pkg
IS
        g_max_sal_raise CONSTANT NUMBER := 0.20;
        PROCEDURE update_sal
        (p_empno emp.empno%TYPE, p_new_sal
   emp.sal%TYPE);
   END sal_pkg;
```

g_max_sal_raise este o constanta globala initializata cu 0.20

update_sal este o procedura publica care actualizeaza salariul angajatului.

Exemplu de corp – pachet sal_pkg

```
CREATE OR REPLACE PACKAGE BODY sal_pkg IS
   FUNCTION validate_raise -- private function
   (p_old_sal emp.sal%TYPE,
   p_new_sal emp.sal%TYPE)
   RETURN BOOLEAN IS
   BEGIN
         IF p_new_sal > (p_old_sal * (1 +
   g_max_sal_raise)) THEN RETURN FALSE;
         ELSE
                  RETURN TRUE;
         END IF;
   END validate_raise;

… -- in continuare procedura publica
…
```

```
PROCEDURE update_sal -- public procedure
(p_empno emp.empno%TYPE,
p_new_sal emp.sal%TYPE)
IS
v_old_sal emp.sal%TYPE; -- local variable
BEGIN
   SELECT sal INTO v_old_sal
   FROM emp
   WHERE empno = p_empno;
   IF validate_raise(v_old_sal, p_new_sal) THEN
         UPDATE emp SET sal = p_new_sal
         WHERE empno = p_empno;
   ELSE
         RAISE_APPLICATION_ERROR(-20210, 'Raise too
   high');
   END IF;
 END update_sal;
END sal_pkg;
```

SAL_PKG

Specification **Body** Dependencies Errors Grants

Save & Compile | Find & Replace | Undo | Redo | Download Source | Drop

```
1  create or replace package body "SAL_PKG" is
2  FUNCTION validate_raise -- private function
3      (p_old_sal emp.sal%TYPE,
4      p_new_sal emp.sal%TYPE)
5      RETURN BOOLEAN IS
6      BEGIN
7          IF p_new_sal > (p_old_sal * (1 + g_max_sal_raise)) THEN RETURN FALSE;
8          ELSE
9              RETURN TRUE;
10         END IF;
11     END validate_raise;
12
13
14 PROCEDURE update_sal -- public procedure
15 (p_empno emp.empno%TYPE,
16 p_new_sal emp.sal%TYPE)
17 IS
18 v_old_sal emp.sal%TYPE; -- local variable
19 BEGIN
20     SELECT sal INTO v_old_sal
21     FROM emp
22     WHERE empno = p_empno;
23     IF validate_raise(v_old_sal, p_new_sal) THEN
24         UPDATE emp SET sal = p_new_sal
25         WHERE empno = p_empno;
26     ELSE
27         RAISE_APPLICATION_ERROR(-20210, 'Raise too high');
28     END IF;
29  END update_sal;
```

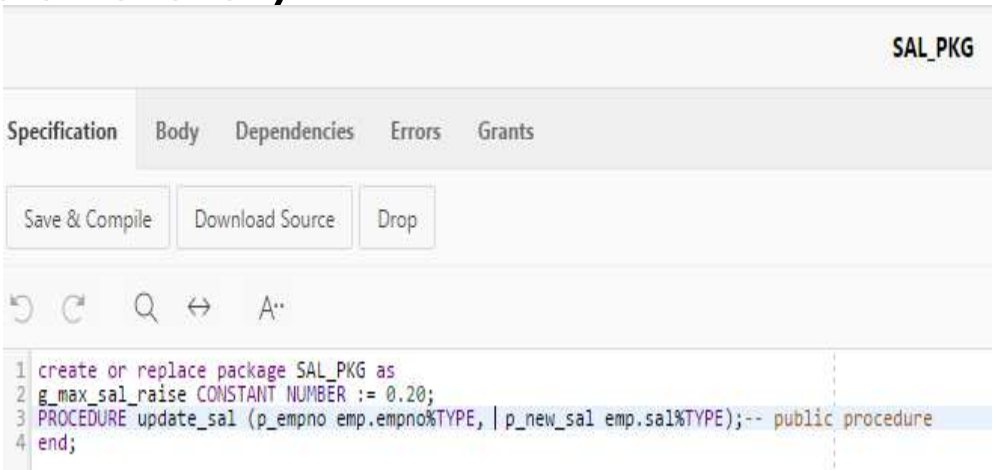
2. Gestionarea conceptelor pachetului

Apelarea subprogramelor pachetului

- Dupa ce pachetul este stocat in baza de date, puteti apela subprogramele stocate in acelasi pachet sau in alte pachete.

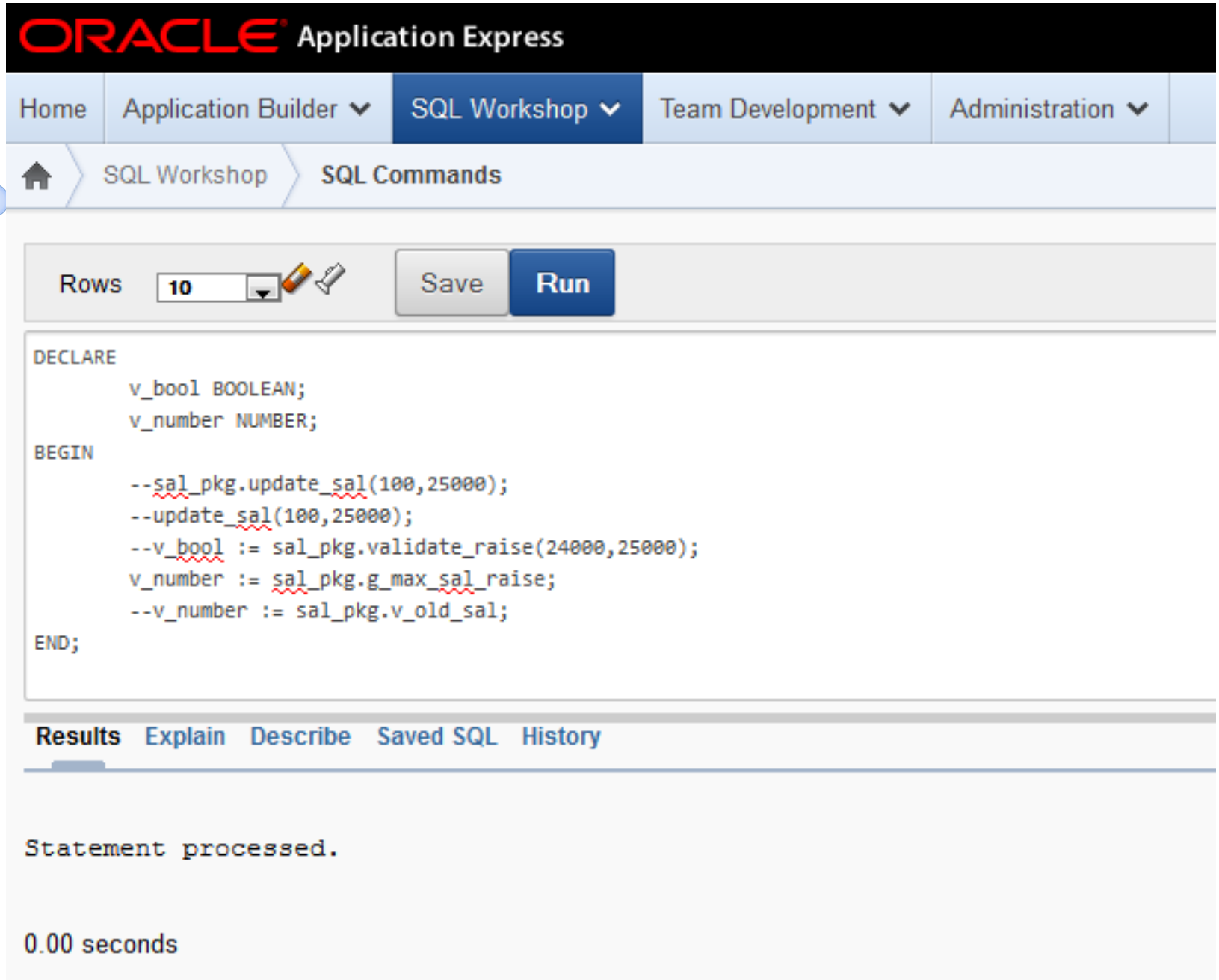
Within the same package	Specify the subprogram name subprogram; You can fully qualify a subprogram within the same package, but this is optional **package_name.subprogram**
External to the package	Fully qualify the (public) subprogram with its package name **package_name.subprogram**

Care dintre urmatoarele apeluri realizate din afara pachetului sal_pkg este valida (presupunand fie ca apelantul detine pachetul, fie are privilegiul EXECUTE asupra pachetului).




```
DECLARE
   v_bool BOOLEAN;
   v_number NUMBER;

BEGIN
a)    sal_pkg.update_sal(100,25000);
b)    update_sal(100,25000);
c)    v_bool := sal_pkg.validate_raise(24000,25000);
d)    v_number := sal_pkg.g_max_sal_raise;
e)    v_number := sal_pkg.v_old_sal;
END;
```


ORACLE Application Express
Home
Application Builder
SQL Workshop
Team Development
Administration
SQL Workshop
SQL Commands
Rows
10
Save
Run
DECLARE
v_bool BOOLEAN;
v_number NUMBER;
BEGIN
--sal_pkg.update_sal(100,25000);
--update_sal(100,25000);
--v_bool := sal_pkg.validate_raise(24000,25000);
v_number := sal_pkg.g_max_sal_raise;
--v_number := sal_pkg.v_old_sal;
END;
Results
Explain
Describe
Saved SQL
History
Statement processed.
0.00 seconds

2. Gestionarea conceptelor pachetului

Stergerea pachetelor

- Pentru a sterge intreg pachetul, specificatia si corpul se foloseste sintaxa:

DROP PACKAGE package_name;

- Pentru a sterge doar corpul pachetului, se foloseste urmatoarea sintaxa:

DROP PACKAGE BODY package_name;

- Nu se poate elimina doar specificatia pachetului

2. Gestionarea conceptelor pachetului

Vizualizarea pachetelor in Data Dictionary

Codul sursa pentru pachetele PL/SQL este mentinut si este vizibil prin tabelele USER_SOURCE si ALL_SOURCE in Data Dictionary.

- Pentru a vizualiza specificatia pachetului se foloseste:

```
SELECT text
FROM user_source
WHERE name = 'sal_PKG' AND type = 'PACKAGE'
ORDER BY line;
```

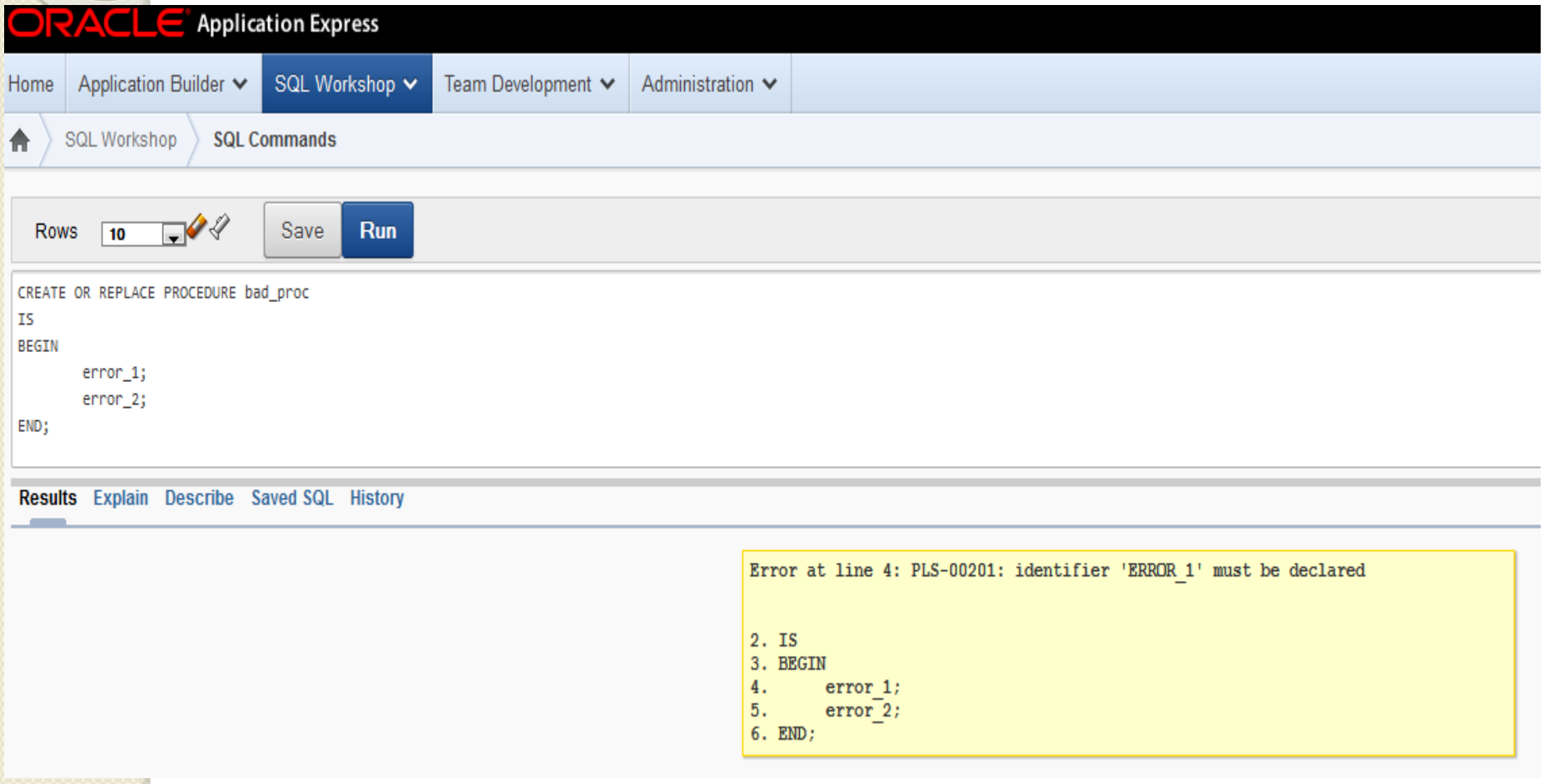
2. Gestionarea conceptelor pachetului

- Pentru a vizualiza corpul pachetului se foloseste:

```
SELECT text
FROM user_source
WHERE name = 'sal_PKG' AND type =
  'PACKAGE BODY'
ORDER BY line;
```

Cum folosim USING_ERRORS?

- Atunci cand un subprogram PL/SQL esueaza la compilare, **Application Express** afiseaza numarul de eroare si textul mesajului:



2. Gestionarea conceptelor pachetului

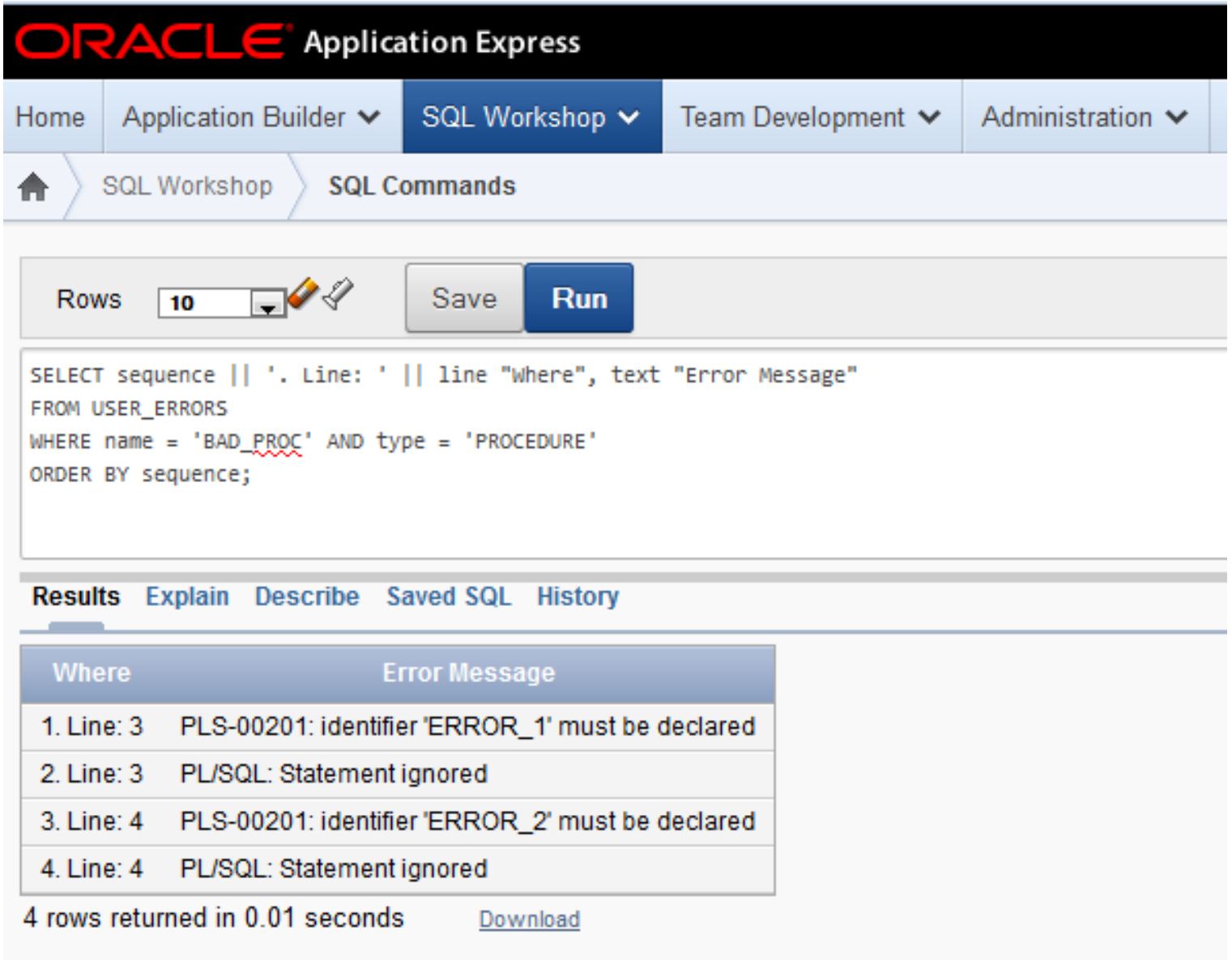
Pentru a vedea toate erorile (nu doar prima), folositi un tabel din dictionarul USER_ERRORS:

```
CREATE OR REPLACE PROCEDURE
  bad_proc
IS
BEGIN
  error_1;
  error_2;
END;
```

2. Gestionarea conceptelor pachetului

```
SELECT sequence || '. Line: ' || line
  "Where", text                    "Error Message"
FROM USER_ERRORS
WHERE name = 'BAD_PROC' AND type
  = 'PROCEDURE'
ORDER BY sequence;
```

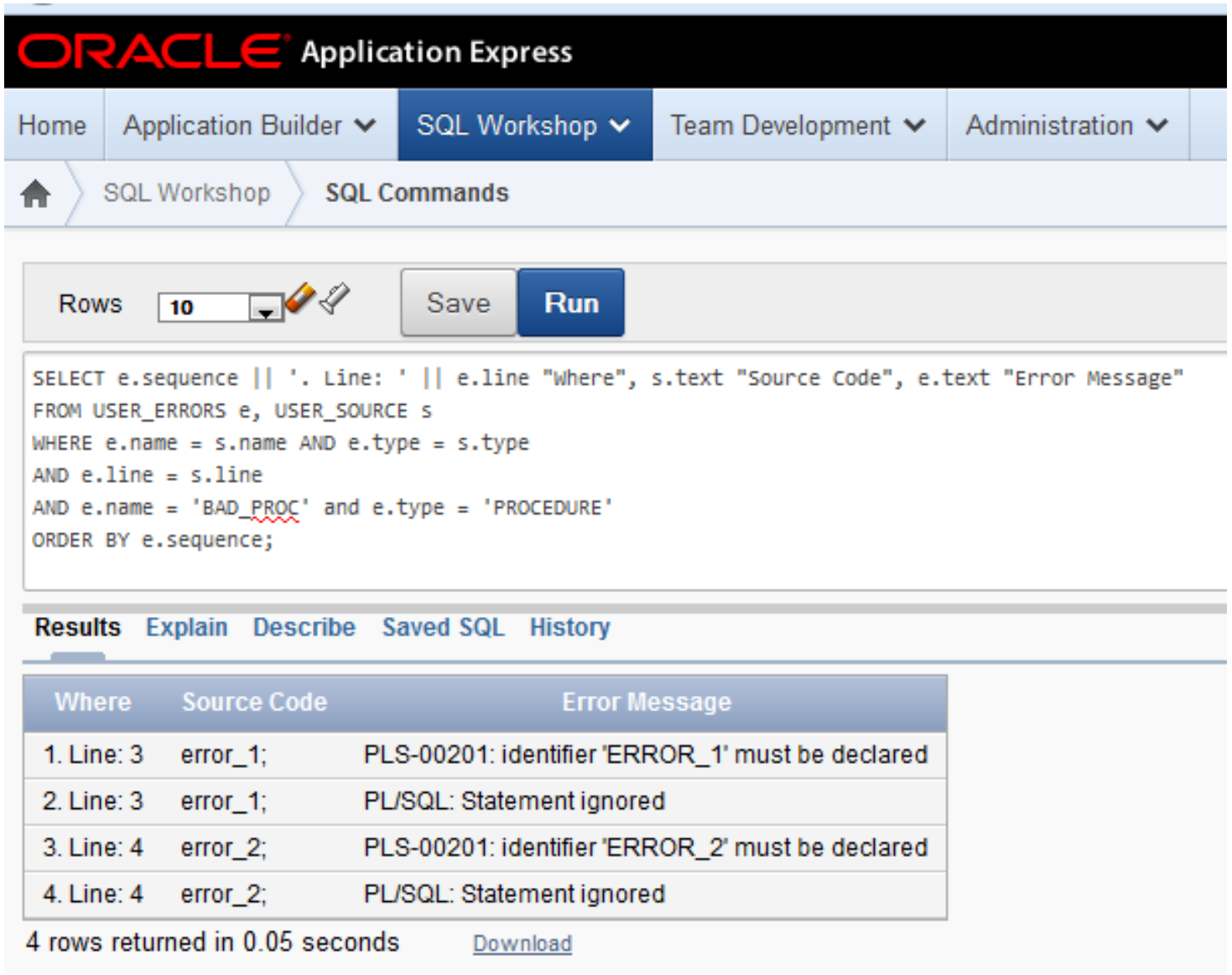
Codul anterior produce urmatoarea iesire:



USER_ERRORS nu indica codul sursa. Dar il putem adauga la USER_SOURCE astfel:

```
SELECT e.sequence || '. Line: ' || e.line
  "Where", s.text  "Source Code", e.text
  "Error Message"
FROM USER_ERRORS e, USER_SOURCE s
WHERE e.name = s.name AND e.type =
  s.type
AND e.line = s.line
AND e.name = 'BAD_PROC' and e.type =
  'PROCEDURE'
ORDER BY e.sequence;
```

Codul anterior produce urmatoarea iesire:



Reguli pentru scrierea pachetelor

- Construiti pachete pentru o utilizare generala
- Creati specificatia pachetului inainte de crearea corpului
- Specificatia pachetului ar trebui sa contina acei constructori care doriti sa fie publici/globali
- Recompilati doar corpul pachetului, daca este posibil, deoarece schimbarile din specificatia pachetului necesita recompilarea tuturor programelor care apeleaza pachetul
- Specificatia pachetului ar trebui sa contina, pe cat posibil, cat mai putini constructori.

Avantajele folosirii pachetelor

1. Modularizarea – incapsularea relatiilor dintre constructori

2. O intretinere mai usoara – pastrarea la un loc a elementelor legate

3. Proiectarea cu mai mare usurinta a aplicatiilor – codarea si compilarea separata a specificatiei si corpului pachetului

Avantajele folosirii pachetelor

4. ***Ascunderea informatiei***:
   - Doar declaratiile din specificatia pachetului sunt vizibile si accesibile aplicatiilor
   - Constructorii privati din corpul pachetului sunt ascunsi si inaccesibili
   - Toate codurile sunt ascunse in corpul pachetului

5. O functionalitate suplimentara – se pastreaza variabilele si cursorii

Avantajele folosirii pachetelor

6. Imbunatatirea performantelor:

- Intregul pachet este incarcat in memorie atunci cand pachetul este referit prima data
- Exista o singura copie in memorie pentru toti utilizatorii
- Se simplifica dependent ierarhia

7. Supraincarcarea – mai multe subprograme au acelasi nume

Cuprins

Pachete în PL/SQL

1. Crearea pachetelor
   1.1. Specificatia pachetului
   1.2. Corpul pachetului
2. Gestionarea conceptelor pachetului
3. Concepte avansate despre pachete

3. Concepte avansate despre pachete

Supraincarcarea subprogramelor

- Termenul de supraincarcare in PL/SQL ne permite sa dezvoltam doua sau mai multe subprograme in pachete care au acelasi nume.
- *Supraincarcarea este utila atunci cand vreti ca un subprogram sa accepte o multime de parametri asemanatori, dar care au tipuri de date diferite.*
- De exemplu, functia TO_CHAR are mai multe modalitati de a fi apelata, permitand convertirea unui numar sau a unei date calendaristice intr-un sir de caractere.

Supraincarcarea in PL/SQL

- Ne permite sa cream doua sau mai multe subprograme cu acelasi nume, in acelasi pachet
- Ne permite sa construim modalitati flexibile sa apelam aceleasi subprograme cu date diferite
- Face ca lucrurile sa fie mai usoare pentru dezvoltatorul de aplicatie, care trebuie sa-si aminteasca doar un nume de subprogram
- *Regula de baza este ca puteti folosi acelasi nume pentru subprograme diferite atata timp cat parametrii lor formali difera ca numar, ordine, categorie sau tip de date*.

Observatie:

- *Supraincarcarea poate fi aplicata doar subprogramelor in pachete, dar nu subprogramelor de sine statatoare*.

Supraincarcarea in PL/SQL(continuare)

- *Luati in considerare supraincarcarea atunci cand sensul a doua sau mai multe subprograme este asemanator, dar tipul sau numarul parametrilor folositi variaza.*
- Supraincarcarea poate oferi modalitati alternative pentru a gasi diferite date cu diferite criterii de cautare.
- De exemplu, poate doriti sa gasiti un angajat dupa id-ul angajatului sau dupa id-ul job-ului sau data angajarii.
- Scopul este acelasi, dar difera parametrii sau criteriul de cautare.

3. Concepte avansate despre pachete

```
CREATE OR REPLACE PACKAGE
  emp_pkg IS
PROCEDURE find_emp
(p_empno IN NUMBER, p_last_name OUT
  VARCHAR2);
PROCEDURE find_emp
(p_job_id IN VARCHAR2, p_last_name
  OUT VARCHAR2);
PROCEDURE find_emp
(p_hiredate IN DATE, p_last_name OUT
  VARCHAR2);
END emp_pkg;
```

Specificatia pachetului ***emp_pkg*** contine o procedura supraincarcata numita ***find_emp***.

Argumentele de intrare ale celor trei declaratii au tipuri de date din categorii diferite.

Care din declaratii se executa in urma apelului de mai jos?

```
CREATE OR REPLACE PACKAGE
  emp_pkg IS
PROCEDURE find_emp
(p_empno IN NUMBER, p_last_name OUT
  VARCHAR2);
PROCEDURE find_emp
(p_job_id IN VARCHAR2, p_last_name
  OUT VARCHAR2);
PROCEDURE find_emp
(p_hiredate IN DATE, p_last_name OUT
  VARCHAR2);
END emp_pkg;
```

```
DECLARE v_last_name VARCHAR2(30);
BEGIN
   emp_pkg.find_emp('IT_PROG',
   v_last_name);
END;
```

Restrictiile supraincarcarilor

- Nu se pot supraincarca:

1. Doua subprograme daca parametrii lor formali difera doar prin tipul lor si tipurile de date sunt in aceeasi categorie (NUMBER si INTEGER apartin aceleiasi categorii; VARCHAR2 si CHAR apartin aceleiasi categorii).

Restrictiile supraincarcarilor(continuare)

2. Doua functii care difera doar in tipul de date al rezultatului, chiar daca tipurile respective fac parte din categorii diferite.

3. De asemenea, aceste restrictii se aplica daca numele parametrilor sunt aceleasi. Daca folositi nume diferite pentru parametri, atunci puteti apela subprogramele folosind notatia denumita pentru subprograme.

3. Concepte avansate despre pachete

```
CREATE PACKAGE sample_pack IS

    PROCEDURE sample_proc
        (p_char_param IN CHAR);

    PROCEDURE sample_proc
        (p_varchar_param IN VARCHAR2);

END sample_pack;
```

3. Concepte avansate despre pachete

- Acum puteti apela o procedura folosind notatia pozitionala.

```
BEGIN
  sample_pack.sample_proc('Smith');
END;
```

- Aceasta esueaza deoarece ‘Smith’ poate fi atat CHAR sau VARCHAR2. Dar urmatorul apel va avea success:

```
BEGIN
  sample_pack.sample_proc(p_char_param
  =>'Smith');
END;
```

3. Concepte avansate despre pachete

- In urmatorul exemplu specificatia pachetului ***dept_pkg*** contine o procedura supraincarcata numita ***add_department***.

- Prima declaratie are trei parametri care sunt folositi pentru a oferi date pentru o noua inregistrare a departamentului inserata in tabela department.
- A doua declaratie are doi parametri deoarece aceasta versiune genereaza intern id-ul departamentului printr-o secventa Oracle.

```
CREATE OR REPLACE PACKAGE BODY dept_pkg    IS
  PROCEDURE add_department (
    p_deptno NUMBER,
    p_name VARCHAR2:='unknown',
    p_loc NUMBER:=1700) IS
  BEGIN
    INSERT INTO dept (deptno, dname, loc)
   VALUES (p_deptno, p_name, p_loc);
  END add_department;

 PROCEDURE add_department (
   p_name VARCHAR2:='unknown',
   p_loc NUMBER:=1700) IS
  BEGIN
    INSERT INTO dept (deptno, dname, loc)
    VALUES (dept_seq.NEXTVAL, p_name, p_loc);
 END add_department;

END dept_pkg;
```

- Daca apelati ***add_department*** cu un id de departament furnizat explicit, atunci PL/SQL foloseste prima versiune a procedurii.
- Fie urmatorul exemplu:

```
BEGIN
dept_pkg.add_department(980,'Education',2
  500);
END;
SELECT *
FROM dept
WHERE deptno = 980;
```

deptno	dname	mgr	loc
980	Education	-	2500

- Daca apelati ***add_department*** fara nici un id de departament, atunci PL/SQL foloseste a doua versiune.

```
BEGIN
dept_pkg.add_department ('Training',
  2500);
END;
SELECT *
FROM dept
WHERE dname = 'Training';
```

deptno	dname	mgr	loc
290	Training	-	2500

Supraincarcarea si pachetul STANDARD

- Un pachet numit **STANDARD** defineste mediul PL/SQL si functiile built-in.
- Cele mai multe functii built-in sunt supraincarcate.
- Ati studiat functia TO_CHAR ca exemplu. Alt exemplu este functia UPPER:

```
FUNCTION UPPER (ch VARCHAR2)
  RETURN VARCHAR2;
FUNCTION UPPER (ch CLOB) RETURN
  CLOB;
```

- Subprogramele pachetului **STANDARD** nu se prefixeaza cu numele pachetului.

Supraincarcarea si pachetul STANDARD

- Ce este intampla daca va creati propria functie cu acelasi nume ca o functie din pachetul standard?
- De exemplu va creati propria functie UPPER.
- Apoi o apelati ca UPPER(argument).
- Care dintre ele se executa?

Supraincarcarea si pachetul STANDARD

Raspuns:

- Chiar daca functia este in schema voastra, se va executa functia STANDARD built-in.
- Pentru a apela functia proprie, este necesar sa o prefixati cu numele schemei voastre.

…

```
BEGIN
   v_return_value := your-schema-
   name.UPPER(argument);
END;
```

Întrebări?