Procedures

Programação em Banco de Dados Computação@UFCG

Prof. Carlos Eduardo Santos Pires

* Características das Procedures
* Operações com Procedures: (re-)criação, execução, visualização e remoção.
* Procedures que se chamam
* Uso de Parâmetros (IN OUT e OUT)
* Procedures Locais
* Uma procedure é essencialmente uma função tendo *void* como tipo de dados de retorno.
* *Void* é explicitamente definido em outras LPs como C, C#, Java e C++.
* Uma procedure **NÃO** pode ser considerada um operando da direita em uma expressão de atribuição.

my\_var := my\_proc(10);

* Uma procedure **NÃO** pode ser chamada a partir de comandos SQL.

SELECT my\_proc(…) FROM…

* Uma procedure pode conter funções ou procedures locais na seção de declaração.
* Uma procedure pode ou não ter parâmetros formais.
* Uma procedure suporta parâmetros nos modos **IN** (default), **OUT** e **IN OUT**.
* Os parâmetros formais podem ser variáveis passadas por **valor** (IN) ou por **referência** (OUT e IN OUT).
* Uma procedure pode retornar valores através de sua lista de parâmetros formais quando passados por referência.
* **Dica**: uma procedure com vários parâmetros de modo IN e um parâmetro OUT deve ser transformada em uma função.
* Características das Procedures
* Operações com Procedures: (re-)criação, execução, visualização e remoção.
* Procedures que se chamam
* Uso de Parâmetros (IN OUT e OUT)
* Procedures Locais

CREATE OR REPLACE [{EDITIONABLE | NONEDITIONABLE}]

[schema.] **PROCEDURE *procedure\_name***

( *parameter* [IN][OUT] [NOCOPY] {*sql\_data\_type* | *plsql\_data\_type*} [,*parameter* [IN][OUT] [NOCOPY] {*sql\_data\_type* | *plsql\_data\_type*}] [, ... ] )

**[ACCESSIBLE BY**

**( [{FUNCTION | PROCEDURE | PACKAGE | TYPE}] [schema.]*unit\_name*) [,[{FUNCTION | PROCEDURE | PACKAGE | TYPE}] [schema.]*unit\_name*)] [,... ]]])**

[ AUTHID DEFINER | CURRENT\_USER ] **IS**

*declaration\_statements*

**BEGIN**

***execution\_statements;***

[EXCEPTION]

*exception\_handling\_statements*

**END** [procedure\_name]**;**

CREATE OR REPLACE PROCEDURE **add\_student**

(p\_id NUMBER, p\_nome VARCHAR2, p\_dob IN DATE) IS

BEGIN

INSERT INTO student\_tab VALUES (p\_id, p\_nome, p\_dob); COMMIT;

END;

CREATE TABLE **socio\_tab**

(socio\_id NUMBER GENERATED AS IDENTITY

CONSTRAINT socio\_id\_pk PRIMARY KEY, socio\_nome VARCHAR2(30));

CREATE TABLE **dependente\_tab**

(dependente\_id NUMBER GENERATED AS IDENTITY

CONSTRAINT dependente\_id\_pk PRIMARY KEY,

socio\_id NUMBER,

dependente\_nome VARCHAR2(30), CONSTRAINT socio\_id\_fk FOREIGN

KEY(socio\_id)

REFERENCES socio\_tab(socio\_id));

CREATE OR REPLACE PROCEDURE adiciona\_socio **(pv\_socio\_nome VARCHAR2, pv\_dependente\_nome VARCHAR2)** IS

lv\_socio\_id NUMBER; BEGIN

**INSERT INTO socio\_tab (socio\_nome) VALUES (pv\_socio\_nome) RETURNING socio\_id INTO lv\_socio\_id;**

INSERT INTO dependente\_tab (socio\_id, dependente\_nome) VALUES (**lv\_socio\_id**, pv\_dependente\_nome);

**COMMIT;**

END;

* A partir de um bloco anônimo:

BEGIN

**adiciona\_socio('Airbender','Epis');**

END;

* A partir do prompt do SQL\*Plus:

EXECUTE

**adiciona\_socio('Airbender','Epis');**

ou

CALL

**adiciona\_socio('Airbender','Epis');**

* A partir de outra procedure:

CREATE PROCEDURE teste IS

BEGIN

**adiciona\_socio('Airbender','Epis');**

END;

* A partir de uma função:

CREATE FUNCTION teste RETURN INTEGER

IS BEGIN

**adiciona\_socio('Airbender','Epis');**

RETURN 1; END;

SQL> **DESCRIBE user\_objects**

Nome Nulo? Tipo

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

**OBJECT\_NAME VARCHAR2(128)**

SUBOBJECT\_NAME VARCHAR2(128)

OBJECT\_ID NUMBER

DATA\_OBJECT\_ID NUMBER

**OBJECT\_TYPE VARCHAR2(23)**

CREATED DATE

LAST\_DDL\_TIME DATE

...

...

TIMESTAMP VARCHAR2(19)

**STATUS VARCHAR2(7)**

TEMPORARY VARCHAR2(1)

GENERATED VARCHAR2(1)

SECONDARY VARCHAR2(1)

NAMESPACE NUMBER

EDITION\_NAME VARCHAR2(128)

SHARING VARCHAR2(13)

EDITIONABLE VARCHAR2(1)

ORACLE\_MAINTAINED VARCHAR2(1)

SELECT object\_name, status FROM **user\_objects**

WHERE object\_type = 'PROCEDURE';

OBJECT\_NAME STATUS

ATUALIZA VALID

ATUALIZAEMAIL VALID

FORMATCEL VALID

GETCEL VALID

...

* DROP PROCEDURE getsel;
* Remove código-fonte e código executável da procedure.
* Se código-fonte não estiver salvo em algum script, não é possível recuperar o código-fonte.
* Objetos que dependam da procedure ficarão inválidos.

CREATE PROCEDURE **p1**

IS

BEGIN

null;

END;

CREATE PROCEDURE **p2**

IS

BEGIN

**p1**; END;

SELECT object\_name, status

FROM **user\_objects**

WHERE object\_name IN ('P1','P2');

**P1 VALID**

**P2 VALID DROP PROCEDURE p1;**

SELECT object\_name, status FROM **user\_objects**

WHERE object\_name IN ('P1','P2');

**P2 INVALID**

# User1

10:00 CREATE PROCEDURE x IS BEGIN null; END;

10:05 GRANT **execute** ON x to User2;

# User2

10:10 BEGIN user1.x; END;

Linha do tempo

* Características das Procedures
* Operações com Procedures: (re-)criação, execução, visualização e remoção.
* Procedures que se chamam
* Uso de Parâmetros (IN OUT e OUT)
* Procedures Locais

CREATE OR REPLACE PROCEDURE **proc5**

IS BEGIN

DBMS\_OUTPUT.PUT\_LINE('proc5');

**-- proc6;**

END;

Procedimento criado.

CREATE OR REPLACE PROCEDURE **proc6**

IS BEGIN

DBMS\_OUTPUT.PUT\_LINE('proc6');

**proc5;**

END;

Procedimento criado.

CREATE OR REPLACE PROCEDURE **proc5**

IS BEGIN

**1**

**3**

DBMS\_OUTPUT.PUT\_LINE('proc5');

**proc6;**

END;

Procedimento criado.

**2**

Computação@UFCG - Prof. Carlos E. S. Pires - Programação em 20

SQL> SELECT status

FROM user\_objects

WHERE object\_name IN ('PROC5','PROC6');

STATUS

Invalidadas pelo SGBD Oracle

------- **INVALID INVALID**

* Características das Procedures
* Operações com Procedures: (re-)criação, execução, visualização e remoção.
* Procedures que se chamam
* Uso de Parâmetros (IN OUT e OUT)
* Procedures Locais

CREATE PROCEDURE **format\_cell**

(string\_in **IN OUT** VARCHAR2)

IS BEGIN

string\_in := '+55 83 '||string\_in;

END;

DECLARE

v\_cell VARCHAR2(17);

BEGIN

v\_cell := '98765-8376';

dbms\_output.put\_line('antes: '||v\_cell); **format\_cell(v\_cell);** dbms\_output.put\_line('depois: '||v\_cell); END;

**antes: 98765-8376**

**depois: +55 83 98765-8376**

CREATE PROCEDURE **get\_dependente** (p\_socio\_id NUMBER, p\_depen\_nome **OUT** VARCHAR2) IS

BEGIN

SELECT dependente\_nome INTO p\_depen\_nome FROM dependente\_tab

WHERE socio\_id = p\_socio\_id; END;

DECLARE

v\_socio\_id NUMBER := 1; v\_depen\_nome VARCHAR2(30); BEGIN

**get\_dependente**(v\_socio\_id, v\_depen\_nome); dbms\_output.put\_line(v\_dependente\_nome); END;

Epis

|  |  |  |
| --- | --- | --- |
| **IN** | **OUT** | **IN OUT** |
| É o modo default e pode ser  suprimido | Deve ser explicitado | Deve ser explicitado |
| O valor é passado para o subprograma | O valor é retornado para o ambiente que chamou | O valor é passado para o subprograma; O valor é retornado para o ambiente que chamou |
| Funciona como uma constante | Variável não inicializada | Variável inicializada |
| Pode ser um literal, expressão, constante ou variável inicializada | Deve ser uma variável | Deve ser uma variável |
| Pode ter um valor default | Não pode ter um valor default | Não pode ter um valor default |

* Características das Procedures
* Operações com Procedures: (re-)criação, execução, visualização e remoção.
* Procedures que se chamam
* Uso de Parâmetros (IN OUT e OUT)
* Procedures Locais

DECLARE

**PROCEDURE first(pv\_caller VARCHAR2) IS BEGIN**

**dbms\_output.put\_line('"First" called by '||pv\_caller); END;**

BEGIN

first('Main'); END;

"First" called by Main

DECLARE

PROCEDURE first(pv\_caller VARCHAR2) IS BEGIN

dbms\_output.put\_line('"First" called by ['||pv\_caller||']');

**second('First');**

END;

PROCEDURE **second**(pv\_caller VARCHAR2) IS

BEGIN

dbms\_output.put\_line('"Second" called by ['||pv\_caller||']'); END;

BEGIN

first('Main');

END;

second('First');

\*

ERRO na linha 5:

ORA-06550: linha 5, coluna 5: PLS-00313: 'SECOND' não

declarado nesta abrangência

ORA-06550: linha 5, coluna 5: PL/SQL: Statement ignored

DECLARE

PROCEDURE second(pv\_caller VARCHAR2); -- referencing stub PROCEDURE first(pv\_caller VARCHAR2) IS

BEGIN

dbms\_output.put\_line('"First" called by ['||pv\_caller||']'); second('First');

END;

PROCEDURE second(pv\_caller VARCHAR2) IS BEGIN

dbms\_output.put\_line('"Second" called by ['||pv\_caller||']'); END;

BEGIN

first('Main'); END;

* "First" called by [Main]
* "Second" called by [First]