

Procedures

Programação em Banco de Dados

Computação@UFCG

Prof. Carlos Eduardo Santos Pires

Outline

- 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

Stored Procedures (ou Procedures)

- 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);



Stored Procedures (ou Procedures)

- 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.

Stored Procedures (ou Procedures)

- 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).

Stored Procedures (ou Procedures)

- 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.

Outline

- 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

Sintaxe

```
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];
```


Criando uma Procedure - Exemplo 1

```
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;
```

Criando uma Procedure - Exemplo 2

```
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));
```

Criando uma Procedure - Exemplo 2

```
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;
```

Executando uma Procedure

- 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');
```

Executando uma Procedure

- 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;
```

Visualizando Procedures Criadas

SQL> DESCRIBE user_objects

Nome	Nulo? Tipo	...	
-----	-----		
OBJECT_NAME	VARCHAR2(128)	STATUS	VARCHAR2(19)
SUBOBJECT_NAME	VARCHAR2(128)	TEMPORARY	VARCHAR2(1)
OBJECT_ID	NUMBER	GENERATED	VARCHAR2(1)
DATA_OBJECT_ID	NUMBER	SECONDARY	VARCHAR2(1)
OBJECT_TYPE	VARCHAR2(23)	NAMESPACE	NUMBER
CREATED	DATE	EDITION_NAME	VARCHAR2(128)
LAST_DDL_TIME	DATE	SHARING	VARCHAR2(13)
		EDITIONABLE	VARCHAR2(1)
		ORACLE_MAINTAINED	VARCHAR2(1)
...			

Visualizando Procedures Criadas

```
SELECT object_name, status  
FROM user_objects  
WHERE object_type = 'PROCEDURE';
```

OBJECT_NAME	STATUS
-----	-----
ATUALIZA	VALID
ATUALIZAEMAIL	VALID
FORMATCEL	VALID
GETCEL	VALID
...	

Removendo uma Procedure

- `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.

Removendo uma Procedure

```
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
-----------	----------------

Atribuindo Privilégio de Execução

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

Outline

- 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

Procedures que se Chamam

```
CREATE OR REPLACE PROCEDURE proc5
IS
BEGIN
  DBMS_OUTPUT.PUT_LINE('proc5');
  -- proc6;
END;
```

Procedimento criado.



```
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.




Procedures que se Chamam

```
SQL> SELECT status  
      FROM user_objects  
      WHERE object_name IN ('PROC5','PROC6');
```

STATUS

INVALID

INVALID



Invalidadas
pelo SGBD
Oracle

Outline

- 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

Procedure com Parâmetro no Modo IN OUT

```
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

Procedure com Parâmetro no Modo OUT

```
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

Resumo dos Modos de Parâmetros

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

Outline

- 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

Procedure Local (Interna) – Exemplo 1

```
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

Procedure Local (Interna) – Exemplo 2

```
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

Procedure Local (Interna) – Exemplo 2

```
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]