Subprograme în PL/SQL

Stocarea subprogramelor

Subprogramele scrise în ***PL/SQL***, spre deosebire de *codurile anonime*, sunt stocate în baza de date. Din acest motiv se mai numesc și ***"proceduri stocate"***.

Dacă lucrați cu *proceduri stocate* pe *calculatoare publice* (cum ar fi, de exemplu, cele de la facultate), este important să nu uitați să le ștergeți din baza de date după ce ați terminat.

Crearea unei proceduri

Procedura va fi *verificată sintactic*, după care *codul ei va fi memorat în baza de date* în cazul în care verificarea nu identifică nici o problemă. Codul nu va mai fi recompilat și atunci când se dorește executarea procedurii.

În *SQL Developer*, în coloana din partea stângă, unde se află și lista tabelelor, aveți o secțiune denumită ***"Procedures"***. Procedura pe care tocmai ați compilat-o va putea fi regăsită în acea secțiune. Inclusiv codul sursă se poate vedea.

Exemplu: ***Example_1.sql***.

Apelarea unei proceduri

Procedurile pot fi *apelate* numai din cadrul unui cod PL/SQL. *Funcțiile* vor putea fi apelate și dintr-o comandă de tip ***select***, spre exemplu.

Exemplu: ***Example_2.sql***.

Ștergerea unei proceduri

Exemplu: ***Example_3.sql***.

Tabela "USER_SOURCE"

Codul sursă al procedurilor scrise este introdus în tabela ***"USER_SOURCE"***.

Exemplu 1: ***Example_4.sql***.

Exemplu 2: ***Example_5.sql***.

Tabela "USER_OBJECTS"

Exemplu: ***Example_6.sql***.

Tabela "USER_PROCEDURES"

Exemplu: ***Example_7.sql***.

Tipuri de subprograme

Există *două tipuri de subprograme* pe care le puteți construi: ***proceduri*** și ***funcții***.

Proceduri

Blocurile pe care le-ați executat până acum au fost denumite ***blocuri anonime*** - tocmai pentru că nu au un nume cu care să poată fi apelate. Blocurile anonime nu sunt stocate în tabela "USER_SOURCE" pentru că nu au un nume și nu pot fi refolosite.

În procedurile anonime exista o secțiune numită ***"DECLARE"*** în care erau precizate *variabilele* ce erau utilizate în blocul anonim.

Această secțiune era opțională și cuvântul *DECLARE* putea fi omis, în cazul în care nu era nevoie de variabile.

Pentru a *declara o procedură* se va utiliza următoarea sintaxă:

```
CREATE [OR REPLACE] PROCEDURE name_of_procedure [(parameter_1 [mode]
data_type_1, parameter_2 [mode] data_type_2, ..., parameter_N [mode]
data_type_N)] AS|IS

     [variable_1 type_1; …]

BEGIN

     // The Code of the Procedure

END [name_of_procedure];
```

După ***"CREATE"*** urmează, opțional cuvintele ***"OR REPLACE"***. Aceste cuvinte au rolul de a *înlocui procedura* în cazul în care aceasta există deja în baza de date. Ați putea face de fiecare dată ***"DROP"*** la procedură după care să o construiți din nou cu ***"CREATE"***, dar acest lucru ar fi mai anevoios.

Cuvântul ***"PROCEDURE"*** indică *tipul de subprogram* ce este construit (vom construi și funcții, iar acestea vor avea cuvântul ***"FUNCTION"*** ca și tip al subprogramului).

Numele prin care va fi apelată procedura este dat în continuare în locul parametrului ***"name_of_procedure"*** din sintaxa de mai sus.

O procedură poate avea mai mulți *parametri de intrare*, mai multe *valori de ieșire* sau *parametri care pot fi modificați în interiorul procedurii* (care sunt în același timp și de intrare și de ieșire). Aceștia sunt dați între paranteze rotunde, despărțiți prin virgulă. Secțiunea opțională ***"mode"*** din definiţia sintactică de mai sus poate avea oricare din valorile ***"IN"***, ***"OUT"*** sau ***"IN OUT"***. Dacă modul lipsește, varianta implicită este ***"IN"***. Atunci când se apelează procedura, trebuie ca pe pozițiile în care se află valori de ieșire (identificate prin ***"OUT"***) să fie neapărat variabile de același tip cu cele declarate în definiția procedurii.

Aceste variabile vor primi după apel valorile variabilelor respective din procedură. Tipul variabilelor poate fi definit și cu ***“%TYPE”***.
Recomandare: prefixați parametrii cu ***"p_ "***.

La sfârșitul declarației procedurii se poate afla oricare dintre cuvintele ***“AS”*** sau ***“IS”***. După acestea urmează variabilele (fără a mai fi nevoie de cuvântul ***“DECLARE”***) și, obligatoriu, *codul executabil al procedurii* (ce poate conține și o secțiune de tratare a excepțiilor).

O procedură poate fi apelată *dintr-un bloc anonim*, *din altă procedură* sau *dintr-o aplicaţie ce poate interacţiona cu server-ul* (de exemplu, *dintr-o aplicație PHP sau Java*). O procedură *nu poate fi apelată* dintr-o comandă de tip ***“SELECT”***.

Parametrii care sunt definiți în antetul procedurii se numesc *parametri formali* (vor fi utilizați în subprogram și din acest motiv este recomandat să îi prefixați cu ***"p_ "***) iar parametrii cu care este apelată procedura (valori sau variabile) se numesc *parametrii actuali*. Când procedura este apelată, fiecărui parametru formal îi va fi atribuită valoarea parametrului actual din apel, în ordine: primului parametru formal îi va fi atribuită valoarea primului parametru actual și așa mai departe.

Puteți trimite un șir de caractere către procedura ce incrementează valoarea și ea va funcționa corect. Acest lucru se întâmplă datorită faptului că PL/SQL știe să facă anumite conversii în mod automat. Din punct de vedere al eficienței, este mai bine ca parametrii actuali (valorile trimise către procedură) să fie de același tip cu parametrii formali (cum au fost declarați în antetul procedurii). În cazul în care nu știți cum au fost declarați parametrii (nu știți tipul parametrilor formali), puteți executa comanda ***“DESCRIBE name_of_procedure”*** pentru a obține aceste informații.

Exemplu 1: ***Example_8.sql***.

Exemplu 2: ***Example_9.sql***.

Exemplu 3: ***Example_10.sql***.

Exemplu 4: ***Example_11.sql***.

Exemplu 5: ***Example_12.sql***.

Unele limbaje de programare permit inițializarea variabilelor formale în mod automat în cazul în care acestea nu primesc valori. Acest lucru este posibil și în PL/SQL.

Exemplu: ***Example_13.sql***.

În acest cod, parametrii de intrare sunt inițializați. În cazul în care aceștia nu sunt transmiși din blocul anonim, valorile predefinite (din definiția funcției) le sunt automat asociate. Pentru a preciza doar un anumit parametru (sau daca doriți să daţi parametrii în altă ordine decât cea din definiția procedurii), puteți să îi asociați în apel sub forma ***cheie=>valoare***. De exemplu, următoarele trei script-uri vor avea același efect:

Exemplu 1: ***Example_14.sql***.

Exemplu 2: ***Example_15.sql***.

Exemplu 3: ***Example_16.sql***.

Următoarele trei script-uri demonstrează faptul că putem transmite parametrii în orice combinație de moduri:

Exemplu 1: ***Example_17.sql***.

Exemplu 2: ***Example_18.sql***.

Exemplu 3: ***Example_19.sql***.

Iată un exemplu în care doar baza și variabila în care trebuie să fie întors rezultatul sunt precizate:

Exemplu 1: ***Example_20.sql***.

Exemplu 2: ***Example_21.sql***.

O procedură poate conține o comandă de tip ***"RETURN"*** pentru a forța ieșirea din acea procedură.

Exemplu 1: ***Example_22.sql***.

Exemplu 2: ***Example_23.sql***.

Mai Multe Exemple

Exemplu 1: ***Example_24.sql***.

Exemplu 2: ***Example_25.sql***.

Exemplu 3: ***Example_26.sql***.

Exemplu 4: ***Example_27.sql***.

Exemplu 5: ***Example_28.sql***.

Exemplu 6: ***Example_29.sql***.

Tot codul se poate găsi aici:

https://github.com/TudorGalatan/DBMS_Practice/tree/main/Project