SQL_EXEC_PROC

Akcia SQL_EXEC_PROC

Funkcia

Akcia vykoná daný SQL príkaz (uloženú procedúru alebo funkciu) so zoznamom parametrov.

Deklarácia

```
SQL_EXEC_PROC handleIdent_Int, retCodeIdent_Int, stringExpr BIND _locVar1,
_locVar2, ...


SQL_EXEC_PROC handleIdent_Int, retCodeIdent_Int, stringExpr BIND
_locVarRowIdent


SQL_EXEC_PROC handleIdent_Int, retCodeIdent_Int, stringExpr BIND
_locVarRecordIdent
```

Parametre

handleIdent_Int	in	Identifikátor typu *Int* - jednoznané íslo (handle) spojenia s databázou.
retCodeIdent_Int	out	Identifikátor typu *Int* - návratový kód.
strExpr	in	Výraz typu *Text* - SQL procedúra alebo funkcia.
_locVar1, _locVar2, ...	in/out	Zoznam lokálnych premenných.
_locVarRowIdent	in/out	Odkaz na riadok lokálnej premennej typu *Record.*
_locVarRecordIdent	in/out	Identifikátor lokálnej premennej typu *Record.*

Návratový kód

Hodnota parametra *retCodeIdent_Int* - pozri tabuku chybových stavov. Je možné získa rozšírenú informáciu o chybe.

Popis

Akcia vykoná SQL príkaz nad databázou, ktorá bola otvorená akciou SQL_CONNECT. Príkaz je reprezentovaný hodnotou výrazu *stringExpr*. SQL príkaz má vstupné, výstupné alebo vstupno-výstupné parametre, ktorých hodnoty udávajú parametre za kúovým slovom *BIND*.
Pokia sa hodnoty parametrov po vykonaní SQL príkazu zmenia, proces D2000 DBManager nastaví príslušné lokálne premenné na zmenenú hodnotu (s aktuálnou asovou znakou).
Pokia je udaný identifikátor typu *Record*, SQL príkaz sa volá pre každý riadok štruktúry. Ak vráti volanie chybu, s alšími riadkami sa už nepokrauje.

Poznámka

Syntax SQL príkazu je rôzna pre *ODBC* a *OCI* verziu procesu D2000 DBManager:

- **ODBC verzia procesu D2000 DBManager:**
  Poda konvencie ODBC je parameter oznaený otáznikom, syntax volania uloženej procedúry je *"{ call NAZOV_PROCEDURY(?,?,..) }"*
  a syntax volania uloženej funkcie je
  "{ ? = call NAZOV_FUNKCIE(?,?,..) }"

  Príklady:
  Funkcia s dvomi parametrami: *"{ ? = call TEST_FUNC(?,?) }"*

  Procedúra s tromi parametrami: *"{ call TEST_PROC(?,?,?) }"*

  Procedúra s tromi parametrami, druhý parameter je konštanta: *"{ call TEST_PROC(?,5,?) }"*

  Nasledujúci príklad funguje pre Sybase SQL Anywhere, nefunguje pre Oracle (ten v ODBC vyžaduje, aby všetky parametre boli bindované ako premenné, nie ako konštanty). Toto obmedzenie sa dá obís použitím OCI syntaxe aj v ODBC verzii: *"BEGIN TEST_PROC(:par1,5,:par3); END;"*.
  Na druhej strane, Sybase SQL Anywhere na rozdiel od Oracle podporuje výstupné parametre iba pre procedúry, nie pre funkcie.

  Poznámka: patche z 7.3.2023 a novšie automaticky doplnia pre Oracle BEGIN/END aj bodkoiarky - kvôli štandardne zapnutej ochrane pred vykonaním viacerých príkazov v rámci jedného volania - vi ladiacu kategóriu DBG.DBMANAGER.SANITIZE), takže funguje nasledovná syntax pre uloženú procedúru:

Príklad pre Oracle:
"BEGIN TEST_PROC(?); END;"
resp.
"TEST_PROC(?)"

Príklad pre PostgreSQL:
"call TEST_PROC(IN ?)"

- **OCI verzia procesu D2000 DBManager:**
  Parameter je oznaený dvojbodkou a menom, syntax volania uloženej procedúry je *"BEGIN NAZOV_PROCEDURY(:par1,:par2,..); END;"* a syntax volania uloženej funkcie je *"BEGIN :vysledok := NAZOV_FUNKCIE(:par1,:par2,..); END;"*
  Poznámka: patche z 7.3.2023 a novšie automaticky doplnia BEGIN/END aj bodkoiarky - kvôli štandardne zapnutej ochrane pred vykonaním viacerých príkazov v rámci jedného volania - vi ladiacu kategóriu DBG.DBMANAGER.SANITIZE), takže funguje nasledovná syntax pre uloženú procedúru:
  NAZOV_PROCEDURY(:par1,:par2,..);
  a pre volanie uloženej funkcie:
  :vysledok := NAZOV_FUNKCIE(:par1,:par2,..)

  Príklady:
  Funkcia s dvomi parametrami: *"BEGIN :res := TEST_FUNC(:par1,:par2); END;"*
  resp.
  :res := TEST_FUNC(:par1,:par2)
  Procedúra s tromi parametrami: *"BEGIN TEST_PROC(:par1,:par2,:par3); END;"*
  resp.
  TEST_PROC(:par1,:par2,:par3)

  Procedúra s tromi parametrami, druhý parameter je konštanta: *"BEGIN TEST_PROC(:par1,5,:par3); END;"*
  resp.
  TEST_PROC(:par1,5,:par3)

  Poradie parametrov je urené poradím v reazci (v predchádzajúcom prípade *:res*, *:par1* a *:par2*). Pokia sú parametre nazvané rovnakým menom, chápu sa ako jediný parameter, t.j. SQL príkaz *" BEGIN :res := TEST_FUNC(:parX,:parX); END;"* má dva parametre a to *:res* (výstup funkcie) a *:parX* (dva vstupné parametre funkcie s tou istou hodnotou).

  V OCI verzii procesu **D2000 DBManager** môže by SQL príkaz aj celá sekvencia, napr. *"BEGIN :res := TEST_FUNC(:par1,:par2); IF :res=0 THEN :res := TEST_FUNC2(:par1,:par2); END IF; END;""*

V rámci volania procedúry je možné špecifikova typ parametra, ktorý sa bude bindova pridaním modifikátora IN, INOUT, OUT pred symbol bindovanej hodnoty.
Príklad:

```
"{ call TEST_PROC(IN ?, INOUT?, OUT?) }"

"TEST_PROC(IN :par1, INOUT :par2, OUT :par3)"
```

Uvedenie modifikátorov nie je povinné (prednastavená hodnota je INOUT). Modifikátory nerozlišujú malé a veké písmená.
Pozn: v prípade databázy PostgreSQL je nutné uvies modifikátor IN pre všetky vstupné parametre, odporúame uvies aj INOUT a OUT modifikátory.

Príklad

1. ODBC verzia procesu D2000 DBManager

Príklad vytvorenia uložených procedúr v SQL Anywhere:

```
/* par1 je vstupno/výstupný parameter, par2 je vstupný a par3 je výstupný */
create procedure TEST_PROC(@par1 varchar(10) output, @par2 integer,@par3 integer output)
as
declare @vysl integer
begin
  select @par=@par+'XYZ'
  select @par3=2*@par2
end

/* príklad funkcie s dvoma parametrami (Sybase podporuje iba vstupné parametre funkcie) */
create function TEST_FUNC(in @par1 real,in @par2 integer)
returns real as
begin
  return(@par1*@par2)
end
```

Volanie zo skriptu:

```
 BEGIN
 INT  _myInt
 INT  _iRetCode
 INT  _iHandle
 TEXT _myText
 REAL _myReal
 INT  _myInt1
 INT  _myInt2

 _myText := "ABC"
 _myInt1 := 10

 SQL_CONNECT MyDB, _iHandle, _iRetCode

 ; volanie procedúry
 SQL_EXEC_PROC _iHandle, _iRetCode, "{ call TEST_PROC(?,?,?) }" BIND
_myText, _myInt1, _myInt2
 ;_myText má hodnotu "ABCXYZ", _myInt2 má hodnotu 20 (2 * 10)

 ; volanie procedúry s konštantným parametrom
 SQL_EXEC_PROC _iHandle, _iRetCode, "{ call TEST_PROC(?,3,?) }" BIND
_myText, _myInt1
 ;_myText má hodnotu "ABCXYZXYZ", _myInt2 má hodnotu 6 (2 * 3)

 ; volanie funkcie
 SQL_EXEC_PROC _iHandle, _iRetCode, "{ ? = call TEST_FUNC(?,?) }" BIND
_myReal, _myInt1, _myInt2
 ; _myReal má hodnotu 60 (10 * 6)

 ; volanie funkcie s konštantným parametrom
 SQL_EXEC_PROC _iHandle, _iRetCode, "{ ? = call TEST_FUNC(?,3.3) }" BIND
_myReal, _myInt1
 ; _myReal má hodnotu 33 (10 * 3.3)
```

2. OCI verzia procesu D2000 DBManagera

Príklad vytvorenia uložených procedúr v Oracle 9i:

```
/* par1 je vstupno/výstupný parameter, par2 je vstupný a par3 je výstupný */
CREATE OR REPLACE  PROCEDURE "MYUSER"."TEST_PROC" (
 par1 in out varchar,par2 integer, par3 out integer
)
as
begin
 par1 := par1 || 'XYZ';
 par3 := 2 * par2;
end;

/* par1,par2 sú vstupné parametre, succ je výstupný */
CREATE OR REPLACE  FUNCTION "MYUSER"."TEST_FUNC" (
 par1 in float, par2 in float, succ out integer
)
return float
as
begin
 if par2 = 0.0 then
  succ := 0;
  return 0;
 else
  succ := 1;
  return par1/par2;
 end if;
end;
```

Volanie zo skriptu:

```
 BEGIN
 INT  _myInt
 INT  _iRetCode
 INT  _iHandle
 TEXT _myText
 REAL _myReal
 INT  _myInt1
 INT  _myInt2
 INT  _Succ

 _myText := "ABC"
 _myInt1 := 10

 SQL_CONNECT MyDB, _iHandle, _iRetCode

 ; volanie procedúry
 SQL_EXEC_PROC _iHandle, _iRetCode, "TEST_PROC(:p1,:p2,:p3)" BIND _myText,
_myInt1, _myInt2
 ;_myText má hodnotu "ABCXYZ", _myInt2 má hodnotu 20 (2 * 10)

 ; volanie funkcie
 SQL_EXEC_PROC _iHandle, _iRetCode, ":ret := TEST_FUNC(:par1,:par2,:par3)"
BIND _myReal, _myInt1, _myInt2, _Succ
 ; _myReal má hodnotu 0.5 (10 / 20), _Succ má hodnotu 1
```

Súvisiace odkazy

DB_TRANS_OPEN
DB_TRANS_COMMIT
DB_TRANS_ROLLBACK
DB_TRANS_CLOSE

SQL_CONNECT
SQL_DISCONNECT
SQL_EXEC_DIRECT

SQL_PREPARE
SQL_BINDIN
SQL_FETCH
SQL_FREE

SQL_SELECT

Všetky databázové akcie

Súvisiace stránky:

Akcie v skriptoch