PROGRAMSKI SQL

Uvod

Programski SQL se implementira i izvršava na nivou servera baze podataka, odnosno sastavni je deo SUBP. Javljaju se u tri vida programskih celina:

- SQLFunkcija — Izvršna SQL komponenta sa ili bez ulaznih argumenata koja kao rezultat vraća jednu povratnu vrednost, pri čemu može samo da očitava pojedine tabele, poglede i kursore.
- SQL Procedura — Izvršna SQL komponenta sa ili bez argumenata koja nema povratnu vrednost, pri čemu u odnosu na pojedine tabele, poglede i kursore može da ima odnos učitavanja i ažuriranja.
- SQL Modul — SQL komponenta koja sadrži više SQL procedura i/ili SQL funkcija, uz mogućnost deklarisanja zajedničkih varijabli i kursora.

Definicije programskih celina

```
Funkcija ::=
  FUNCTION Funkcija ( [ { Parametar Tip },.. ] ) RETURNS Tip
           TeloFunkcije ;

Procedura ::=
  PROCEDURE Procedura ( [ { IN|OUT|INOUT Parametar Tip },.. ] )
            TeloProcedure ;

Modul ::=
  MODULE Modul TeloModula

TeloFunkcije ::=
  { RETURN Izraz }|{ BEGIN SQLNaredba SQLNaredba... RETURN Izraz END }

TeloProcedure ::=
  BEGIN SQLNaredba SQLNaredba... [ RETURN ] END

TeloModula ::=
  BEGIN [ Deklaracije ] { Funkcija|Procedura }... END

Parametar ::= :NazivParametra
```

Kod definisanja funkcije, procedure i modula kao prva javlja se ključna reč **CREATE**.
Prilikom navođenja funkcija i procedura unutar modula to nije slučaj.

Deklaracije može da sadrži:

- deklaracije varijabli,
- kreiranje privremenih tabela
- deklaracije kursora.

Naredbe deklaracije varijabli i dodele vrednosti

```
DeklaracijaVarijabli ::=
  DECLARE Varijabla,.. Tip [ DEFAULT Konstanta ] ;
```

Varijable deklarisane bez DEFAULT opcije imaju vrednost NULL.

```
DodelaVrednosti ::=
  SET Varijabla|Parametar = Konstanta |SkalarniIzraz | NULL ;
```

Unutar **SkalarniIzraz** mogu se javiti varijable, parametri, konstante i pozivi funkcija.

Naredbe kontrole toka

```
SekvencaNaredbi ::= [ Labela: ] BEGIN
                                  SQLNaredba
                                  1..
                    [ Labela: ] END
```

```
UslovnoGrananje ::=   IF LogickiIzraz THEN
                         SQLNaredba
                         1..
                    [ ELSE IF LogickiIzraz THEN
                         SQLNaredba
                         1.. ]
                      ...
                    [ ELSE ]
                         SQLNaredba
                         1..
                      END IF
```

Za sve petlje koje slede: ako se koriste obe labele, one moraju biti iste.

```
PetljaLOOP ::= [ Labela: ] LOOP
                              SQLNaredba
                              1..
               [ Labela: ] END LOOP
```

Unutar tela petlje LOOP mora se pojaviti bar jedna uslovljena naredba iskoka.

```
PetljaWHILE ::= [ Labela: ] WHILE LogickiIzraz DO
                               SQLNaredba
                               1..
                [ Labela: ] END WHILE
```

```
PetljaREPEAT ::= [ Labela: ] REPEAT
                                SQLNaredba
                                1..
                              UNTIL LogickiIzraz
                 [ Labela: ] END REPEAT
```

```
PetljaFOR ::= [ Labela: ] FOR Varijabla,.. AS Upit | Kursor DO
                              SQLNaredba
                              1..
              [ Labela: ] END FOR
```

Ova petlja iterira kroz redove upita ili kursora bez mogućnosti izmene podataka u redu.

```
IskokIzPetlje ::= EXIT | { LEAVE [ Labela ] }
```

EXIT može da napusti samo petlju u kojoj se neposredno nalazi, a LEAVE sa opcijom labele može da napusti proizvoljan broj obuhvatajućih petlji.

Naredba poziva procedure

```
PozivProcedure ::=
  { Procedura ( Argument,.. ) } | CALL Procedura USING Argument,..
```

Naredbe izmene tabele

```
UbacivanjeJedanRed ::=
  INSERT INTO Tabela [ ( Kolona,.. ) ]
         VALUES ( { Varijabla | Parametar | Konstanta },.. ) ;
```

Ako se neka vrednost izostavi, za odgovarajuću kolonu se unosi NULL.

```
BrisanjeRedova ::=
  DELETE FROM Tabela [ Nadimak ]
       [ WHERE R-Predikat ] ;
```

```
IzmenaRedova ::=
  UPDATE Tabela [ Nadimak ]
         SET { Kolona = S-Izraz },..
       [ WHERE R-Predikat ] ;
```

Naredba upita reda

```
UpitJedanRed ::= SELECT { S-Izraz | G-Izraz },..
                   INTO { Varijabla | Parametar },..
                   FROM { Tabela [ Nadimak ] },..
                 [ WHERE R-Predikat ]
                 [ GROUP BY Kolona,..
                 [ HAVING G-Predikat ] ] ;
```

Ovaj upit mora biti takav da se očekuje jedan red, u suprotnom je u pitanju greška.
Odmah posle ovog upita treba ispitati da li je uopšte učitan jedan red.

Obrada izuzetka

Preko standardne globalne promenljive SQLCODE se nakon svake izvršene naredbe signalizira ishod. Pri tome je SQLCODE predefinisan (kao i CURRENT_TIME i slično) i ne treba ga deklarisati.

```
DefinicijaObradeIzuzetka ::=
  WHENEVER Ishod CONTINUE | { GOTO Labela } ;
```

Ako nema definicije obrade izuzetka, podrazumeva se CONTINUE.

```
Ishod ::=
  NegativnaNenultaKonstanta | { { SQLWARNING | NOT_FOUND } | SQLERROR }
```

Značenje vrednosti ishoda označavaju:

0	naredba je uspešno izvršena;
NOT_FOUND	upit nije vratio ni jedan red,
SQLWARNING	izmena nije izmenila/obrisala ni jedan red;
SQLERROR	nastupila je greška (SQLCODE < 0).

Savremeniji i detaljniji način je preko SQLSTATUS (struktura sa više polja).