

SQL/PSM tárolt modulok --- 1.rész

Tankönyv: Ullman-Widom:
Adatbázisrendszerek Alapvetés
Második, átdolgozott kiadás,
Panem, 2009



9.3. Az SQL és a befogadó nyelv
közötti felület (sormutatók)
9.4. SQL/PSM Sémában
tárolt függvények és eljárások

-- itt: PSM1modulok: utasítások, modulok, PSM-kivételek
-- később lesz: PSM2kurzorok: lekérdezések PSM-ben

SQL programnyelvi környezetben

- Milyen problémák merülnek fel, amikor egy alkalmazás részeként, programban használjuk az SQL utasításokat?
- 1.) **Osztott változók használata:** közös változók a nyelv és az SQL utasítás között (ott használható SQL utasításban, ahol kifejezés használható).
- 2.) **A típuseltérés problémája:** Az SQL magját a relációs adatmodell képezi. Tábla – gyűjtemény, sorok multihalmaza, mint adattípus nem fordul elő a magasszintű nyelvekben. A lekérdezés eredménye hogyan használható fel? Megoldás: kurzorral, az eredmény soronkénti bejárása.

Háromféle programozási megközelítés

- 1.) **SQL kiterjesztése procedurális eszközökkel**, az adatbázis séma részeként tárolt kódrészekkel, tárolt modulokkal (pl. **PSM** = Persistent Stored Modules, **Oracle PL/SQL**).
- 2.) **Beágyazott SQL** (sajátos előzetes beágyazás EXEC SQL. - Előfordító alakítja át a befogadó gazdanyelvre/host language, pl. C)
- 3.) **Hívásszintű felület**: hagyományos nyelvben programozunk, függvénykönyvtárat használunk az adatbázishoz való hozzáféréshez (pl. CLI = call-level interface, JDBC, PHP/DB)

PSM – Persistent Stored Procedures

- SQL/PSM is a part of the latest revision to the SQL standard, called SQL:2003
- PSM, or “*persistent stored modules*,” allows us to store procedures as database schema elements.
- PSM = a mixture of conventional statements (if, while, etc.) and SQL statements.
- Lets us do things we cannot do in SQL alone.
- Each commercial DBMS offers its own extension of PSM, e.g. Oracle: PL/SQL (see, in practice)

PSM tárolt eljárások

- **Tárolt eljárások** (SQL objektumok)

CREATE PROCEDURE eljárás-név (
paraméter-lista)

[DECLARE ... deklarációk]

BEGIN

az eljárás utasításai;

END;

- **Paraméter lista** (tárolt eljárásban)

[IN | OUT | INOUT] paraméternév értéktípus

PSM eljárások paramétereit

- **Mód – Név – Típus** hármast
- The parameters of a PSM procedure:
Unlike the usual name-type, PSM uses mode-name-type triples, where the *mode* can be:
 - IN = procedure uses value, does not change value.
 - OUT = procedure changes, does not use.
 - INOUT = both.

PSM tárolt függvények

➤ Tárolt függvények

CREATE FUNCTION függvény-név (
paraméter-lista) RETURNS értéktípus

[DECLARE ... deklarációk]

BEGIN

utasítások ...

END;

➤ Függvények paraméterei:

➤ may only be of mode **IN**

(PSM forbids side-effects in functions)

Példa: Stored Procedure

- Let's write a procedure that takes two arguments b and p , and adds a tuple to **Sells(bar, beer, price)** that has $\text{bar} = \text{'Joe''s Bar'}$, $\text{beer} = b$, and $\text{price} = p$.
- Used by Joe to add to his menu more easily.

```
CREATE PROCEDURE JoeMenu (  
  IN  b  CHAR(20) ,  
  IN  p  REAL  
)  
INSERT INTO Sells  
VALUES ('Joe''s Bar', b, p);
```

Parameters are both read-only, not changed

The body a single insertion

Legfontosabb utasítások --- 1

1. Eljáráshívás: The call statement

```
CALL <procedure name> (<argument  
list>);
```

Use SQL/PSM statement CALL, with the name of the desired procedure and arguments.

➤ Example:

```
CALL JoeMenu ('Moosedrool', 5.00);
```

Legfontosabb utasítások --- 2

2. The return statement

Függvényhívás: Functions used in SQL expressions wherever a value of their return type is appropriate.

RETURN <expression> sets the return value of a function.

- Unlike C, etc., RETURN *does not* terminate function execution.

3. Változók deklarációja

DECLARE <name> <type>

used to declare local variables.

Legfontosabb utasítások --- 3

4. Értékadás - Assignment statements

SET <variable> = <expression>;

- Example: SET b = 'Bud' ;

5. Statement group

BEGIN . . . END for groups of statements.

- Separate statements by semicolons.

6. Statement labels

- give a statement a label by prefixing a name and a colon.

7. SQL utasítások

- DELETE, UPDATE, INSERT, MERGE
- (de SELECT nem, azt később nézzük)

IF Statements

- Simplest form:
IF <condition> THEN
<statements(s)>
END IF;
- Add ELSE <statement(s)> if desired, as
IF ... THEN ... ELSE ... END IF;
- Add additional cases by ELSEIF <statements(s)>:
IF ... THEN ... ELSEIF ... THEN ...
ELSEIF ... THEN ... ELSE ... END IF;

Example: IF

- Let's rate bars by how many customers they have, based on `Frequents(drinker,bar)`.
 - <100 customers: 'unpopular'.
 - 100-199 customers: 'average'.
 - ≥ 200 customers: 'popular'.
- Function `Rate(b)` rates bar b.

Example: IF (continued)

```
CREATE FUNCTION Rate (IN b CHAR(20) )
  RETURNS CHAR(10)
  DECLARE cust INTEGER;
BEGIN
  SET cust = (SELECT COUNT(*)
              FROM Frequents WHERE bar = b);
  IF cust < 100 THEN RETURN 'unpopular'
  ELSEIF cust < 200 THEN RETURN 'average'
  ELSE RETURN 'popular'
  END IF;
END;
```

-- Number of
-- customers of
-- bar b

-- Return occurs here,
-- not at one of the RETURN --
statements

-- Nested
-- IF statement

Ciklusok

- Basic form:

```
<loop label>: LOOP <statements>  
                END LOOP;
```

- Exit from a loop by:

```
LEAVE <loop label>
```

Example: Exiting a Loop

címke: LOOP

...

LEAVE címke;

...

END LOOP;

Other Loop Forms

- WHILE <condition>
DO <statements>
END WHILE;

- REPEAT <statements>
UNTIL <condition>
END REPEAT;

PSM kivételek

- Az SQL-rendszer a hibákat egy ötjegyű SQLSTATE nevű karakterlánc beállításával jelzi, például '02000' jelzi, hogy nem talált sort.
- Kivételek kezelése, kivételek nem feltétlen hiba, hanem a normálistól való eltérés kezelése
- DECLARE <hova menjen>
HANDLER FOR <feltétel lista>
<utasítás>
- <hova menjen> lehetőségek:
CONTINUE, EXIT, UNDO

Tankönyv 9.16 példa (9.18 ábra)

PL/SQL

- Oracle uses a variant of SQL/PSM which it calls PL/SQL.
- PL/SQL not only allows you to create and store procedures or functions, but it can be run from the *generic query interface* (sqlplus), like any SQL statement.
- Triggers are a part of PL/SQL.

Trigger Differences

- Compared with SQL standard triggers, Oracle has the following differences:
 1. Action is a PL/SQL statement.
 2. New/old tuples referenced automatically.
 3. Strong constraints on trigger actions designed to make certain you can't fire off an infinite sequence of triggers.

SQLPlus

- In addition to stored procedures, one can write a PL/SQL statement that looks like the body of a procedure, but is executed once, like any SQL statement typed to the generic interface.
- Oracle calls the generic interface “sqlplus.”
- PL/SQL is really the “plus.”

Form of PL/SQL Statements

DECLARE

<declarations>

BEGIN

<statements>

END;

.

run

➤ The DECLARE section is optional.

Form of PL/SQL Procedure

CREATE OR REPLACE PROCEDURE

<name> (<arguments>) **AS**

← Notice AS
needed here

<optional declarations>

BEGIN

<PL/SQL statements>

END;

·
run

← Needed to store
procedure in database;
does not really run it.

PL/SQL Declarations és Assignments

- The word DECLARE does not appear in front of each local declaration.
 - Just use the variable name and its type.
- There is no word SET in assignments, and := is used in place of =.
 - **Example:** `x := y;`

PL/SQL Procedure Parameters

- There are several differences in the forms of PL/SQL argument or local-variable declarations, compared with the SQL/PSM standard:
 1. Order is name-mode-type, not mode-name-type.
 2. INOUT is replaced by IN OUT in PL/SQL.
 3. Several new types.

PL/SQL Types

- In addition to the SQL types, NUMBER can be used to mean INT or REAL, as appropriate.
- You can refer to the type of attribute x of relation R by $R.x\%TYPE$.
 - Useful to avoid type mismatches.
 - Also, $R\%ROWTYPE$ is a tuple whose components have the types of R 's attributes.

Example: JoeMenu

- Recall the procedure **JoeMenu(b,p)** that adds beer *b* at price *p* to the beers sold by Joe (in relation Sells).
- Here is the PL/SQL version.

```
CREATE OR REPLACE PROCEDURE JoeMenu (  
    b IN Sells.beer%TYPE,  
    p IN Sells.price%TYPE  
) AS  
BEGIN  
    INSERT INTO Sells  
    VALUES ('Joe''s Bar', b, p);  
END;
```

run

PL/SQL Branching Statements

- Like IF ... in SQL/PSM, but:
- Use ELSIF in place of ELSEIF.
- Viz.: IF ... THEN ... ELSIF ... THEN ... ELSIF ... THEN ... ELSE ... END IF;

PL/SQL Loops

- LOOP ... END LOOP as in SQL/PSM.
- Instead of LEAVE ... , PL/SQL uses
EXIT WHEN <condition>
- And when the condition is that cursor *c* has found no tuple, we can write *c%NOTFOUND* as the condition.

Tuple-Valued Variables

- PL/SQL allows a variable x to have a tuple type.
- $x \text{ R}\%ROWTYPE$ gives x the type of R 's tuples.
- R could be either a relation or a cursor.
- $x.a$ gives the value of the component for attribute a in the tuple x .