

Aplicații cu BD

Persistent Stored Modules (PSM)

Embedded SQL

Utilizare SQL în aplicații

1. Codul scris într-un limbaj specializat este stocat în BD (de exemplu PSM, PL/SQL – Oracle sau Transact-SQL – Microsoft SQL Server).
2. Instrucțiunile SQL sunt incluse într-un *limbaj gazdă* (cum este C).
3. Se folosesc unelte de conectare pentru a permite unui limbaj convențional să acceseze o bază de date (de exemplu CLI, JDBC, PHP/DB).

Proceduri Stocate

- PSM, sau "*persistent stored modules*," permit stocare de proceduri ca elemente ale schemei BD.
- PSM = un amalgam de instrucțiuni convenționale (if, while, etc.) și SQL.
- Oferă posibilități altfel inexistente în SQL.

Formatul de bază PSM

```
CREATE PROCEDURE <nume> (  
    <listă de parametri> )  
    <declarații locale opționale>  
    <corp>;
```

□ Alternativa este Funcția:

```
CREATE FUNCTION <nume> (  
    < listă de parametri > ) RETURNS <tip>
```

Parametri în PSM

- Spre deosebire de alte limbaje cum este C unde există perechi nume-tip, PSM folosesc triplete *mod-nume-tip*, unde *mod* poate fi:
 - IN = procedura folosește valoarea, dar nu o poate modifica.
 - OUT = procedura modifică valoarea, nu o utilizează.
 - INOUT = ambele.

Exemplu: Procedură Stocată

- Vom scrie o procedură ce primește două argumente b și p , iar acțiunea constă în adăugarea unei tuple la relația **Sells(bar, beer, price)** ce are "bar = 'Joe's Bar'", "beer = b " și "price = p ".
 - Este utilizată de Joe pentru a-și alcătui mai ușor meniul.

Procedura

```
CREATE PROCEDURE JoeMenu (
```

```
  IN b    CHAR(20),  
  IN p    REAL
```

Parametrii sunt amândoi
read-only, nu pot fi modificați

```
)
```

```
  INSERT INTO Sells  
  VALUES('Joe"s Bar', b, p);
```

Corpul:
o singură adăugare

Apelul Procedurilor

- Se folosește instrucțiunea EXECUTE (CALL) urmată de numele procedurii și de argumente.

- Exemplu:

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

- Funcțiile sunt folosite în expresii SQL oriunde se potrivește valoarea returnată (ca tip de dată).

Instrucțiuni specifice PSM

- RETURN <expresie> setează valoarea returnată de funcție.
 - Spre deosebire de C, etc., RETURN *NU* termină execuția funcției.
- DECLARE <nume> <tip> se folosește pentru a declara variabile locale.
- BEGIN . . . END grupează instrucțiuni.
 - Instrucțiunile se separă cu “;”.

Instrucțiuni specifice PSM

□ Instrucțiuni de atribuire:

SET <variabilă> = <expresie>;

□ Exemplu: SET b = 'Bud' ;

□ Etichete: se prefixează numele cu ":".

Instrucțiuni IF

- Forma simplificată:

```
IF <condiție> THEN  
    <instrucțiune(-i)>  
END IF;
```

- Se adaugă ELSE <instrucțiune(-i)>,
IF . . . THEN . . . ELSE . . . END IF;

- Se poate adăuga suplimentar ELSEIF
<instrucțiune(-i)>: IF ... THEN ... ELSEIF ...
THEN ... ELSEIF ... THEN ... ELSE ... END IF;

Exemplu: IF

- Să se noteze barurile după numărul de clienți. Se folosește **Frequents(drinker,bar)**.
 - Clienți < 100 : 'nepopular'.
 - Clienți 100-199 : 'mediu'.
 - Clienți ≥ 200 : 'popular'.
- Funcția **Rate(b)** acordă notă barului b.

Exemplu: IF (continuare)

```
CREATE FUNCTION Rate (IN b CHAR(20) )
```

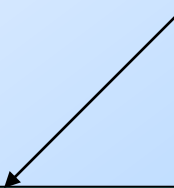
```
  RETURNS CHAR(10)
```

```
  DECLARE cust INTEGER;
```

```
  BEGIN
```

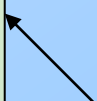
```
    SET cust = (SELECT COUNT(*) FROM Frequents  
                WHERE bar = b);
```

Numărul
clienților
barului b



```
    IF cust < 100 THEN RETURN 'nepopular'  
    ELSEIF cust < 200 THEN RETURN 'mediu'  
    ELSE RETURN 'popular'  
    END IF;
```

Instrucțiune
IF imbricată



```
  END;
```

Return apare în acest loc, nu unde
se utilizează instrucțiunile RETURN



Bucle

□ Forma de bază:

<nume buclă>: LOOP <instrucțiuni>
END LOOP;

□ Ieșirea dintr-o buclă:

LEAVE <nume buclă>

Exemplu: Ieșirea din Buclă

bucla1: LOOP

. . .

LEAVE bucla1; — Dacă este executată această
instrucțiune . . .

. . .

END LOOP;

← Controlul trece la acest punct (după END LOOP)

Alte Forme de Bucle

- WHILE <condiție>
 DO <instrucțiuni>
 END WHILE;
- REPEAT <instrucțiuni>
 UNTIL <condiție>
 END REPEAT;

Interogări

- În general interogările SELECT-FROM-WHERE *NU* sunt permise în PSM.
- Există trei moduri pentru a obține efectul unei interogări:
 1. Interogările ce produc o singură valoare pot fi utilizate ca expresie într-o atribuire.
 2. SELECT . . . INTO având rezultat 1 tuplă.
 3. Cursoare.

Exemplu: Atribuire/Interogare

- Se folosește variabila locală p și **Sells(bar, beer, price)**, pentru a obține prețul la care "Joe" vinde "Bud":

```
SET p = (SELECT price FROM Sells  
        WHERE bar = 'Joe''s Bar' AND  
               beer = 'Bud' );
```

SELECT . . . INTO

- O altă cale pentru a obține valoarea unei interogări ce returnează 1 tuplă este să se folosească **INTO** <variabilă> după clauza SELECT.

- **Exemplu:**

```
SELECT price INTO p FROM Sells
WHERE bar = 'Joe''s Bar' AND
       beer = 'Bud';
```

Cursoare

□ Un *cursor* este în esență o variabilă de tuplă ce parcurge toate tuplele din rezultatul unei anumite interogări.

□ Se declară cursorul *c* în felul următor:

```
DECLARE c CURSOR FOR <interogare>;
```

Deschiderea și Închiderea de Cursoare

- Pentru a folosi cursorul c , trebuie emisă comanda:

OPEN c ;

- Interogarea din definiția c este evaluată și c este setat să facă referire la prima tuplă a rezultatului.

- La terminarea lucrului cu c , este emisă comanda:

CLOSE c ;

Extragerea Tuplelor dintr-un Cursor

- Pentru a obține următoarea tuplă din cursorul c , este emisă comanda:

FETCH FROM c INTO x_1, x_2, \dots, x_n ;

- x -urile sunt o listă de variabile, câte una pentru fiecare componentă a tuplei referite de către c .
- c se deplasează automat la următoarea tuplă.

Ieșirea din bucla unui Cursor

- În mod obișnuit un cursor este folosit într-o buclă creată cu instrucțiunea FETCH, pentru fiecare tuplă extrasă se execută o anumită acțiune.
- Se pune întrebarea cum se iese din buclă atunci când nu mai există tuple de prelucrat?

Ieșirea din bucla unui Cursor

- Fiecare operație SQL returnează o *stare*, ce este un șir de caractere format din 5 cifre.
 - De exemplu, 00000 = "Este în regulă," și 02000 = "Regăsirea unei tuple a eșuat."
- În PSM, se poate obține valoarea stării într-o variabilă numită SQLSTATE.

Ieșirea din bucla unui Cursor

□ Se poate declara o *condiție*, care să fie o variabilă logică ce este "true" numai dacă SQLSTATE are o valoare particulară.

□ **Exemplu:** Se poate declara condiția NeGasit pentru a reprezenta 02000 :

```
DECLARE NeGasit CONDITION FOR  
SQLSTATE '02000';
```

Ieșirea din bucla unui Cursor

- Structura buclei unui cursor arată în felul următor:

```
Buclacursor: LOOP
```

```
...
```

```
FETCH c INTO ... ;
```

```
IF NeGasit THEN LEAVE Buclacursor;
```

```
END IF;
```

```
...
```

```
END LOOP;
```

Exemplu: Cursor

- Presupunem că dorim să scriem o procedură care să examineze `Sells(bar, beer, price)` și să mărească cu 1 (\$) prețul berilor vândute de “Joe’s Bar” și care au prețul mai mic decât 3 (\$).
- Obs. Se putea rezolva cu un simplu UPDATE, dar scopul este de a vedea soluția folosind un cursor.

Declarațiile necesare

```
CREATE PROCEDURE JoeGouge( )
```

```
    DECLARE Berea CHAR(20);
```

```
    DECLARE Pretul REAL;
```

```
    DECLARE NeGasit CONDITION FOR  
        SQLSTATE '02000';
```

```
    DECLARE c CURSOR FOR
```

```
        (SELECT beer, price FROM Sells  
         WHERE bar = 'Joe's Bar');
```

Se vor folosi
pentru a păstra
perechi bere-preț
la extragerea
datelor folosind
cursorul c

Se returnează
meniul la "Joe's Bar"

Corpul Procedurii

BEGIN

OPEN c;

Buclo_meniu: LOOP

FETCH c INTO Berea, Pretul;

Verifică dacă cel mai recent FETCH nu a regăsit nici o tuplă

IF NeGasit THEN LEAVE Buclo_meniu END IF;

IF Pretul < 3.00 THEN

UPDATE Sells SET price = Pretul + 1.00

WHERE bar = 'Joe's Bar' AND beer = Berea;

END IF;

END LOOP;

CLOSE c;

END;

Dacă "Joe's Bar" are un preț mai mic de 3 (\$) pentru bere, mărește prețul acelei mărci de bere la barul "Joe's Bar" cu 1 (\$).

PL/SQL

- ❑ Oracle folosește o variantă de SQL/PSM numită PL/SQL.
- ❑ PL/SQL permite crearea de proceduri stocate și funcții, ce pot fi executate din "*generic query interface*" (sqlplus), ca orice instrucțiune SQL.
- ❑ În sqlplus se poate scrie o instrucțiune PL/SQL asemănător cu corpul unei proceduri, dar aceasta este executată o singură dată.

Instrucțiuni PL/SQL

DECLARE

<declarații>

BEGIN

<instrucțiuni>

END;

.

run

□ Secțiunea DECLARE este opțională.

Procedura în PL/SQL

CREATE OR REPLACE PROCEDURE

<nume> (<argumente>) **AS**

De notat
prezența "AS"

<declarații opționale>

BEGIN

<instrucțiuni PL/SQL>

END;

•
run

Efectul este stocarea
procedurii în BD;
nu execuția ei.

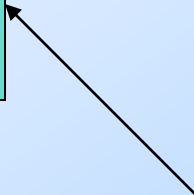
Exemplu:JoeMenu

- Se va defini în PL/SQL procedura **JoeMenu(b,p)** ce adaugă berea b cu prețul p la lista berilor vândute de barul “Joe’s Bar” (în relația Sells).

Procedura "JoeMenu" în PL/SQL

```
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;
```

De notat aceste
precizări
generice de tip.



·
run

Procedură stocată în Microsoft SQL Server

- Următoarea procedură returnează un cursor ce conține descrierea valutelor:

```
USE AdventureWorks;
GO
IF OBJECT_ID ( 'dbo.uspCurrencyCursor', 'P' ) IS NOT NULL
    DROP PROCEDURE dbo.uspCurrencyCursor;
GO
CREATE PROCEDURE dbo.uspCurrencyCursor
    @CurrencyCursor CURSOR VARYING OUTPUT
AS
    SET @CurrencyCursor = CURSOR
    FORWARD_ONLY STATIC FOR
    SELECT CurrencyCode, Name
    FROM Sales.Currency;
    OPEN @CurrencyCursor;
GO
```

Procedură stocată în Microsoft SQL Server

- Următoarea secvență de instrucțiuni prezintă utilizarea cursorului creat de procedura stocată uspCurrencyCursor într-un script SQL:

```
USE AdventureWorks;
GO
DECLARE @MyCursor CURSOR;
EXEC dbo.uspCurrencyCursor @CurrencyCursor = @MyCursor OUTPUT;
WHILE (@@FETCH_STATUS = 0)
BEGIN;
    FETCH NEXT FROM @MyCursor;
END;
CLOSE @MyCursor;
DEALLOCATE @MyCursor;
GO
```

Embedded SQL

- **Ideea de bază:** Un preprocesor translatează instrucțiunile SQL în apeluri de procedură ce se încadrează în codul limbajului gazdă.
- Toate instrucțiunile “embedded SQL” încep cu EXEC SQL, în așa fel încât preprocesorul să le descopere cu ușurință.

Variabile Partajate

- Pentru a “închega” SQL cu programul în limbaj gazdă, cele două părți trebuie să partajeze anumite variabile.
- Declarațiile variabilelor partajate sunt încadrate de:

 **EXEC SQL** BEGIN DECLARE SECTION;

<declarații limbaj gazdă>

 **EXEC SQL** END DECLARE SECTION;

Utilizarea Variabilelor Partajate

- În SQL, variabilele partajate trebuie să fie precedate de ":".
- Ele pot fi folosite ca și constante furnizate de programul limbaj gazdă.
- Ele pot primi valori prin instrucțiuni SQL și pot transfera aceste valori programului limbaj gazdă.
- În limbajul gazdă, variabilele partajate se comportă ca orice altă variabilă.

Exemplu: Căutarea Prețului

- Se va utiliza limbajul "C" cu "embedded SQL" pentru a schița părțile importante ale unei funcții de căutare a prețului cunoscându-se denumirea berii și denumirea barului.
- Se folosește relația **Sells(bar, beer, price)**.

Exemplu: C și SQL

```
EXEC SQL BEGIN DECLARE SECTION;
```

```
char Barul[21], Berea[21];
```

```
float Pretul;
```

```
EXEC SQL END DECLARE SECTION;
```

```
/* obținerea valorilor Barul și Berea */
```

```
EXEC SQL SELECT price INTO :Pretul
```

```
FROM Sells
```

```
WHERE bar = :Barul AND beer = :Berea;
```

```
/* anumite operații cu Pretul */
```

De notat, tablourile
au 21 caractere
pentru 20
caractere +
endmarker

SELECT-INTO
asemănător
cu PSM

Embedded Queries

- “Embedded SQL” are aceleași limitări ca și PSM cu privire la interogări:
 - SELECT-INTO pentru o interogare trebuie să garanteze că produce o singură tuplă.
 - În caz contrar, trebuie folosit un cursor.
 - Există mici diferențe sintactice, dar ideea rămâne aceeași.

Instrucțiuni Cursor

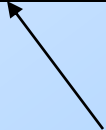
- Declararea unui cursor *c* se face cu:
`EXEC SQL DECLARE c CURSOR FOR <interogare>;`
- Se deschide și se închide cursorul *c* cu:
`EXEC SQL OPEN CURSOR c;`
`EXEC SQL CLOSE CURSOR c;`
- Datele se extrag din cursorul *c* cu:
`EXEC SQL FETCH c INTO <variabilă(-e)>;`
 - Macroul NOT FOUND este true dacă și numai dacă FETCH eșuează în a mai găsi o tuplă.

Exemplu: Tipărirea meniului “Joe’s Bar”

- Vom scrie C + SQL pentru a tipări meniul “Joe’s Bar” – lista perechilor bere-preț pe care o regăsim din relația **Sells(bar, beer, price)** cu condiția “bar = Joe’s Bar”.
- Un cursor va vizita fiecare tuplă Sells ce are bar = “Joe’s Bar”.

Exemplu: Declarațiile

```
EXEC SQL BEGIN DECLARE SECTION;  
    char Berea[21]; float Pretul;  
EXEC SQL END DECLARE SECTION;  
EXEC SQL DECLARE c CURSOR FOR  
    SELECT beer, price FROM Sells  
    WHERE bar = 'Joe''s Bar';
```



Cursorul se declară în exteriorul
secțiunii de declarații

Exemplu: Partea Executabilă

```
EXEC SQL OPEN CURSOR c;
```

```
while(1) {
```

```
    EXEC SQL FETCH c
```

```
        INTO :Berea, :Pretul;
```

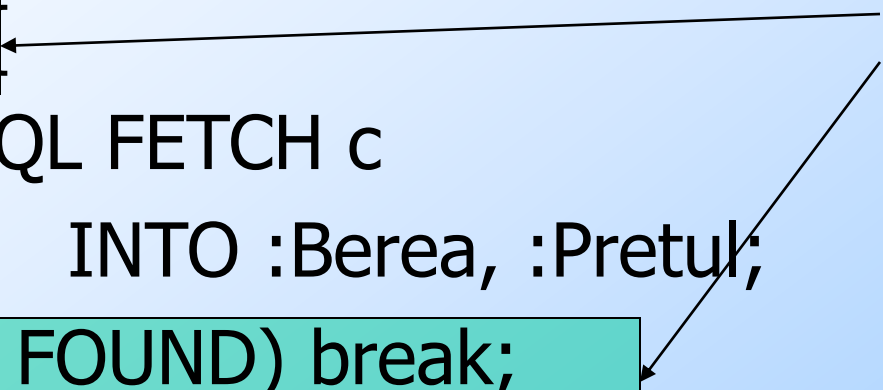
```
    if (NOT FOUND) break;
```

```
    /* se formatează și se tipărește Berea și Pretul */
```

```
}
```

```
EXEC SQL CLOSE CURSOR c;
```

Stilul C pentru
terminarea
repetiției



Necesitatea SQL Dinamic

- Majoritatea aplicațiilor folosesc interogări specifice și instrucțiuni de actualizare ce interacționează cu BD.
 - SGBD-ul compilează instrucțiuni EXEC SQL ... în apeluri procedură specifice și produce un program obișnuit în limbaj gazdă ce folosește o bibliotecă.
- Sqlplus nu știe ceea ce are nevoie să facă până la execuție.

SQL Dinamic

- Pregătirea ("prepare") unei interogări:
EXEC SQL PREPARE <nume-interogare>
FROM <textul interogării>;

- Execuția unei interogări:
EXEC SQL EXECUTE <nume-interogare>;

- "Prepare" = optimizare interogare.

- Se pregătește o dată, se execută de mai multe ori.

Exemplu: O Interfață Generică

```
EXEC SQL BEGIN DECLARE SECTION;
```

```
    char interogare[MAX_LENGTH];
```

```
EXEC SQL END DECLARE SECTION;
```

```
while(1) {
```

```
    /* se afișează promptul SQL> */
```

```
    /* se citește interogarea utilizator în tabloul  
    interogare */
```

```
    EXEC SQL PREPARE q FROM :interogare;
```

```
    EXEC SQL EXECUTE q;
```

```
}
```

q este o variabilă SQL
ce reprezintă forma optimizată
a unei instrucțiuni ce este
ținută în :*interogare*

Execute-Immediate

- Dacă urmează să executăm doar o singură dată interogarea, se pot combina pașii PREPARE și EXECUTE în unul singur:

```
EXEC SQL EXECUTE IMMEDIATE <text>;
```

Exemplu: Interfața Generică

```
EXEC SQL BEGIN DECLARE SECTION;  
    char interogare[MAX_LENGTH];  
EXEC SQL END DECLARE SECTION;  
while(1) {  
    /* se afișează prompter SQL> */  
    /* se citește interogarea utilizator în tabloul  
    interogare */  
    EXEC SQL EXECUTE IMMEDIATE :interogare;  
}
```