SQL-2003/PSM

Procedure memorizzate nel database
Programmazione general-purpose

Lucidi derivati da quelli di Jeffrey D. Ullman

Procedure memorizzate

- Una estensione di SQL, chiamata SQL/PSM, o "persistent, stored modules," ci consente di memorizzare procedure come elementi di uno schema di database.
- Lo stile di programmazione e' un misto di istruzioni convenzionali (if, while, ecc.) ed SQL.
- Ci consente di fare cose che non si potrebbero fare con SQL da solo.

Forma base di PSM

CREATE PROCEDURE <nome> (
    <lista di parametri>)
  <dichiarazioni locali opzionali>
  <corpo>;

- Alternativa funzione:

CREATE FUNCTION <nome> (
    <lista di parametri>) RETURNS
  <tipo>

Parametri in PSM

- A differenza delle usuali coppie nome-tipo in linguaggi come il C, i PSM usano triple modo-nome-tipo, dove il *modo* puo' essere:
  - IN = la procedure usa il valore, non lo cambia.
  - OUT = la procedure lo cambia, non lo usa.
  - INOUT = entrambe.

Esempio: procedure memorizzate

- Tabella Sells(bar, beer, price)
- Scriviamo una procedura che prende due argomenti *b* e *p* e aggiunge una tupla a Sells che ha bar=’Joe”s Bar’, beer = *b*, e price = *p*.
  - Usata da Joe per aggiungere piu’ facilmente una voce al suo menu.

La procedura

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

I parametri sono entrambi di sola lettura, non vengono cambiati

Il body---
un singolo inserimento

Invocare le procedure

- Usa l'istruzione CALL di SQL/PSM, con il nome della procedura e gli argomenti.
- Esempio:

  CALL JoeMenu('Moosedrool', 5.00);

- Le funzioni sono usate in espressioni SQL dove un valore del loro tipo di ritorno e' appropriato.

Tipi di istruzioni PSM -- 1

- RETURN <espressione>; imposta il valore di ritorno di una funzione.
  - A differenza del C, ecc., RETURN *non* termina l'esecuzione della funzione.
- DECLARE <nome> <tipo>; e' usato per dichiarare variabili locali.
- Usa BEGIN . . . END; per gruppi di istruzioni.
  - Separate da punto e virgola.

Tipi di istruzioni PSM -- 2

- Istruzioni di assegnamento:

  SET <variabile> = <espressione>;

  - Esempio: SET b = 'Bud';
- Etichette di istruzioni: si puo' dare a una istruzione una etichetta facendola precedere da un nome e due punti.

Istruzioni IF

- Forma base:

  IF <condizione> THEN

  <istruzione(i) separate da ;>

  END IF;

- Si puo' aggiungere l'istruzione ELSE <istruzione(i)> se richiesto, come in

  IF . . . THEN . . . ELSE . . . END IF;

- Si aggiungono i casi addizionali con ELSEIF <istruzione(i) separate da ;>:

IF … THEN … ELSEIF … ELSEIF … ELSE … END IF;

Esempio: IF

- Classfichiamo i bar a seconda di quanti clienti hanno, sulla base di Frequents(drinker, bar).
  - <100 clienti: ‘unpopular’.
  - 100-199 clienti: ‘average’.
  - >= 200 clienti: ‘popular’.
- La funzione Rate(b) classifica il bar b.

Esempoi: IF (continua)

```
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;
```

Numero di clienti del bar b

Istruzione IF innestata

Il ritorno avviene qui, non a una delle istruzioni RETURN

Cicli

- Forma base:

  LOOP <istruzioni> END LOOP;

- Esci da un ciclo con:

  LEAVE <nome del ciclo>

- Il <nome del ciclo> e' associato a un ciclo facendo precedere la parola chiave LOOP da un nome e due punti.

Esempio: uscire da un ciclo

```
loop1: LOOP
        . . .
        LEAVE loop1;
        . . .
END LOOP;
```

Se questa istruzione e' eseguita. . .

...il controllo passa qui

Altre forme di ciclo

- WHILE <condizione>
  DO
  <istruzioni>
  END WHILE;
- REPEAT
  <istruzioni>
  UNTIL <condizione>
  END REPEAT;

Interrogazione

- Interrogazioni generali del tipo SELECT-FROM-WHERE *non* sono permesse in PSM.
- Ci sono tre modi per ottenere l'effetto di una interrogazione:
  1. Le interrogazioni che producono un valore singolo possono essere l'espressione in un assegnamento.
  2. SELECT . . . INTO che restituiscono una sola riga.
  3. Cursori.

Esempio: assegnamento/interrogazione

- Se *p* e' una variabile locale e Sells(bar, beer, price) la relazione usuale, possiamo ottenere il prezzo che Joe chiede per la Bud con:

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

SELECT . . . INTO

- Un modo equivalente per ottenere il valore di una query che sicuramente restituisce una singola tupla e' di inserire INTO <variabile> dopo la clausola SELECT.
- Esempio:

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

Cursori

- Un *cursore* e' essenzialmente una variabile-tupla che varia su tutte le tuple nel risultato di una query.
- Si dichiara un cursore *c* con:

DECLARE c CURSOR FOR <query>;

Aprire e chiudere cursori

- Per usare il cursore *c*, dobbiamo dare il comando:

  OPEN c;

  - La query di *c* e' valutata e *c* punta alla prima tupla del risultato.
- Quando abbiamo finito con *c*, diamo il comando:

  CLOSE c;

Recuperare tuple da un cursore

- Per ottenere la prossima tupla da un cursore *c* si usa il comando:

  FETCH FROM c INTO x1, x2,…,x*n* ;

- Gli xi sono variabili, una per ogni componente delle tuple riferite da *c*.
- *c* e' spostato automaticamente alla prossima tupla.

Uscire dai cicli con i cursori - 1

- Il modo usuale di usare un cursore e' di creare un ciclo con una istruzione FETCH e fare qualcosa con ogni tupla recuperata.
- Un punto delicato e' come uscire dal ciclo quando il cursore non ha piu' tuple da recuperare.

Uscire dai cicli con i cursori - 2

- Ogni operazione SQL restituisce uno *stato*, che e' un sequnza di cifre in una stringa di 5 caratteri.
  - Per esempio, `00000' = "Tutto OK," e `02000' = "Nessuna tupla trovata."
- In PSM, possiamo ottenere il valore dello stato in una variabile chiamata SQLSTATE.

Uscire dai cicli con i cursori - 3

- Possiamo dichiarare una condizione, che e' una variabile booleana che e' vera se e solo se SQLSTATE ha un valore particolare.
- Esempio: possiamo dichiarare che la condizione NotFound rappresenta 02000 by:

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

Uscire dai cicli con i cursori - 4

- La struttura di un ciclo con cursore e' cosi' :

```
cursorLoop: LOOP
   …
   FETCH c INTO … ;
   IF NotFound THEN LEAVE cursorLoop;
   END IF;
   …
END LOOP;
```

Esempio: cursore

- Scriviamo una procedura che esamina Sells(bar, beer, price), e incrementa di $1 il prezzo di tutte le birre al bar di Joe che costano meno di $3.
  - Si', si potrebbe fare cio' con una semplice UPDATE, ma i dettagli sono istruttivi.

Le dichiarazioni richieste

```
CREATE PROCEDURE JoeIncrease ( )
  DECLARE theBeer CHAR(20);
  DECLARE thePrice REAL;
  DECLARE NotFound CONDITION FOR
      SQLSTATE '02000';
  DECLARE c CURSOR FOR
      (SELECT beer, price FROM Sells
       WHERE bar = 'Joe"s Bar');
```

Usate per contenere coppie beer-price quando le si recupera attraverso il cursore c

Restituisce il menu del bar di Joe

Il corpo della procedura

```
BEGIN
   OPEN c;
   menuLoop: LOOP
      FETCH c INTO theBeer, thePrice;
      IF NotFound THEN LEAVE menuLoop END IF;
      IF thePrice < 3.00 THEN
         UPDATE Sells SET price = thePrice+1.00
         WHERE bar = 'Joe"s Bar' AND beer = theBeer;
      END IF;
   END LOOP;
   CLOSE c;
END;
```

Verifica se la recente FETCH non ha restituito una tupla

Se Joe chiede meno di $3 per la birra, alza il prezzo al bar di Joe di $1.