Sviluppo di Applicazioni per Basi di Dati

Lorenzo Vaccarecci

23 Maggio 2024

1 Problema

SQL non permette di eseguire elaborazioni complesse sui dati e interagire con il sistema operativo.
SQL quindi non è completo:

- Computazionalmente: banalmente mancano i costrutti di scelta o iterazione
- Operazionalmente: mancano i costrutti per interagire con il sistema operativo

2 Soluzione

Si combina SQL con un linguaggio di programmazione generico.
Approcci:

- interno(al DBMS): lato server, il codice eseguito dal DBMS
- esterno(al DBMS): lato client o server, il codice eseguito da un'applicazione che invia la query al DBMS (*es. un programma Java*).

3 Routine

I programmi sono organizzati in routine (procedure o funzioni) e possono essere eseguite direttamente dal DBMS o chiamate da applicazioni basate su accoppiamento esterno.

3.1 Routine Anonime

Sono dei blocchi di codice a cui non viene dato un nome:

```
1    DO
2    $$
3        <Corpo Routine>
4    $$
5    LANGUAGE <Linguaggio>;
```

Il linguaggio usato deve essere supportato dal DBMS.

4 Attività

Questa parte di codice crea una routine anonima che stampa un messaggio:

```
DO
$$
    BEGIN
        RAISE NOTICE 'Hello, world!';
    END
$$;
```

Per dichiarare delle variabili

```
DO
$$
    DECLARE
        <Nome Variabile> <Tipo Variabile> [:= <Valore> | DEFAULT <Valore>];
    BEGIN
        <Corpo Routine>
    END
$$
```

La dichiarazione è molto simile a quella usata per creare le colonne di una tabella.
Per poter stampare le variabili:

```
RAISE NOTICE 'Messaggio %', <Variabile>;
```

Se la variabile non è stata inizializzata, stamperà `NULL`. Ovviamente se una variabile ha l'opzione `NOT NULL` non possiamo assegnarli `NULL`.
A una variabile possiamo assegnare il risultato di una query:

```
denominazioneCdL := (SELECT denominazione FROM corsidilaurea WHERE id = 9)
```

While loop:

```
DO
$$
    BEGIN
        WHILE <Condizione>
        LOOP
            <Corpo Loop>
        END LOOP;
    END
$$
```

4.1 Cursore

Serve per manipolare il risultato di una query.

```
DECLARE
    <Nome Cursore> CURSOR FOR <Query>;
BEGIN
    OPEN <Nome Cursore>; -- Apre il cursore ed esegue la query
    FETCH <Nome Cursore> INTO <Variabile>; -- Prende il risultato della query e lo mette nella variabile
END
```