Applicazioni per DB / 23-05

Si risponde alla richiesta di poter visualizzare, inserire ed elaborare dati comodamente con un DB, che SQL non necessariamente garantisce.

SQL non è completo:
- computazionalmente (no scelte e iterazioni)
- operazionalmente (no sviluppo di interfacce, no I/O da file)

Si combina allora SQL con un linguaggio di programmazione generico

Si amplia SQL, eseguito internamente dal DBMS (lato server, vicino ai dati)
(in PostGre → pgSQL)

Esternamente un linguaggio di programmazione viene eseguito lato client (a server esterno), si collega al DBMS e manda in esecuzione comandi SQL.

Nel linguaggio posso avere:
- librerie di funzioni standard e proprietarie
  - JDBC
- SQL ospitato nel linguaggio stesso
  - # sql <comando>
  - EXEC SQL ...

PgSQL

```
CREATE [OR REPLACE] FUNCTION <nome routine>
    ([<lista parametri>])
[RETURNS <tipo ritorno>]
AS
$$ <corpo routine> $$
LANGUAGE plpgsql
```

ogni <parametro> = [<modo>] [<nome param>] <tipo param>
                    ↑ se I/O

La funzione può essere chiamata in un comando SQL o istruzione pgSQL dove è possibile lavorare con una singola tupla.

SELECT AggiornaVel1('dinamico') -> se avessi parametri out non devo specificarli
Restituisce una tabella con una tupla vuota
-> solo se la significa che funziona la procedura (aggiorna le tuple con quel genere)

Se non specifico RETURN nella creazione della funzione, il valore di output viene allora messo nel parametro di output

Nota: anche se ho più parametri di output, ho solo una colonna comunque (di tipo record, un tipo strutturato)

SELECT * FROM <nome funzione> restituisce più colonne quante i parametri di output

Il corpo di una procedura ha una sezione di dichiarazione e una di esecuzione (es. BEGIN & END)

DECLARE (posso imporre vincoli e inizializzare)
[nome parav] [tipo] [vincoli/inizializzazioni]

Assegnazione: <nome var> := <espressione>

Come scrivere in output: RAISE NOTICE <stringa> [<espressione>]
→ se contiene % sostituisce con l'espressione
(simil C)

→ lo si vede solo lato server, no utente (quello eseguito dal DBMS), utile x debugging

Tramite la clausola SELECT valutare INTO la Valutare
FROM ...
WHERE ...
↑
variabile

immette nella variabile il valore dato dalla query SCALARE
(un solo risultato, ad es. tramite selezione su chiave)

→ se la query restituisce più di una tupla viene preso il valore della prima tupla

→ se restituisce un insieme vuoto mette NULL
(se dichiarato NOT NULL dà errore)

se specifico STRICT dà disambiguazione non vengono applicate e segnala errore

disambiguazione

Costrutti: IF ... END IF

[WHILE] LOOP ... END LOOP
↑ condiz

se non c'è condizione / è WHILE TRUE

FOR IN [REVERSE] LOOP ... END LOOP

Posso utilizzare cursori per manipolare risultati tupla per tupla senza dover mettere singole tuple nelle variabili
→ il risultato ha dim. non nota a tempo di compilazione
→ può essere troppo grande x caricarlo in memoria

Il cursore permette di "portare in memoria" tupla per tupla, è un puntatore a una tupla risultato di una query

dichiaro: <nome cursore> CURSOR FOR <select statement>
(risultato dell'interrogazione ancora non calcolato)

apro: OPEN <nome cursore> si elabora la query

prelevamento: FETCH <nome cursore> [INTO <lista variabili>]

chiudo: CLOSE <nome cursore>
→ rompo il legame fra cursore e query
→ libero la memoria riservata al cursore

WHILE FOUND
↳ permette di "esplorare" tutto il cursore

Solo nel caso necessito di operazioni tupla per tupla uso i cursori,

altrimenti è sconsigliato

Questa vista è detta SQL statico (noto a tempo di compilazione)

Vi è anche SQL dinamico:

```
EXECUTE <espressione stringa>
[INTO [STRICT] <lista variabili>]
```

ovvero valuta a runtime dell'applicazione il comando

(o concatenazione fra stringhe ovviene con ||

Si possono anche gestire eccezioni (dentro begin end)

```
EXCEPTION
    WHEN <condizione>
```

il valore è salvato dentro una var globale SQLSTATE
(dovrei conoscere i codici di errori), posso usare alcune costanti
di ritorno:

es. no_data è quando accedo a dati mancanti (es. FETCH)
se ho esempio la query specificata nel cursore da risultato
vuoto e vi accedo
→ il sistema lancia eccezione → posso catturarla e
mostrare un messaggio di errore.