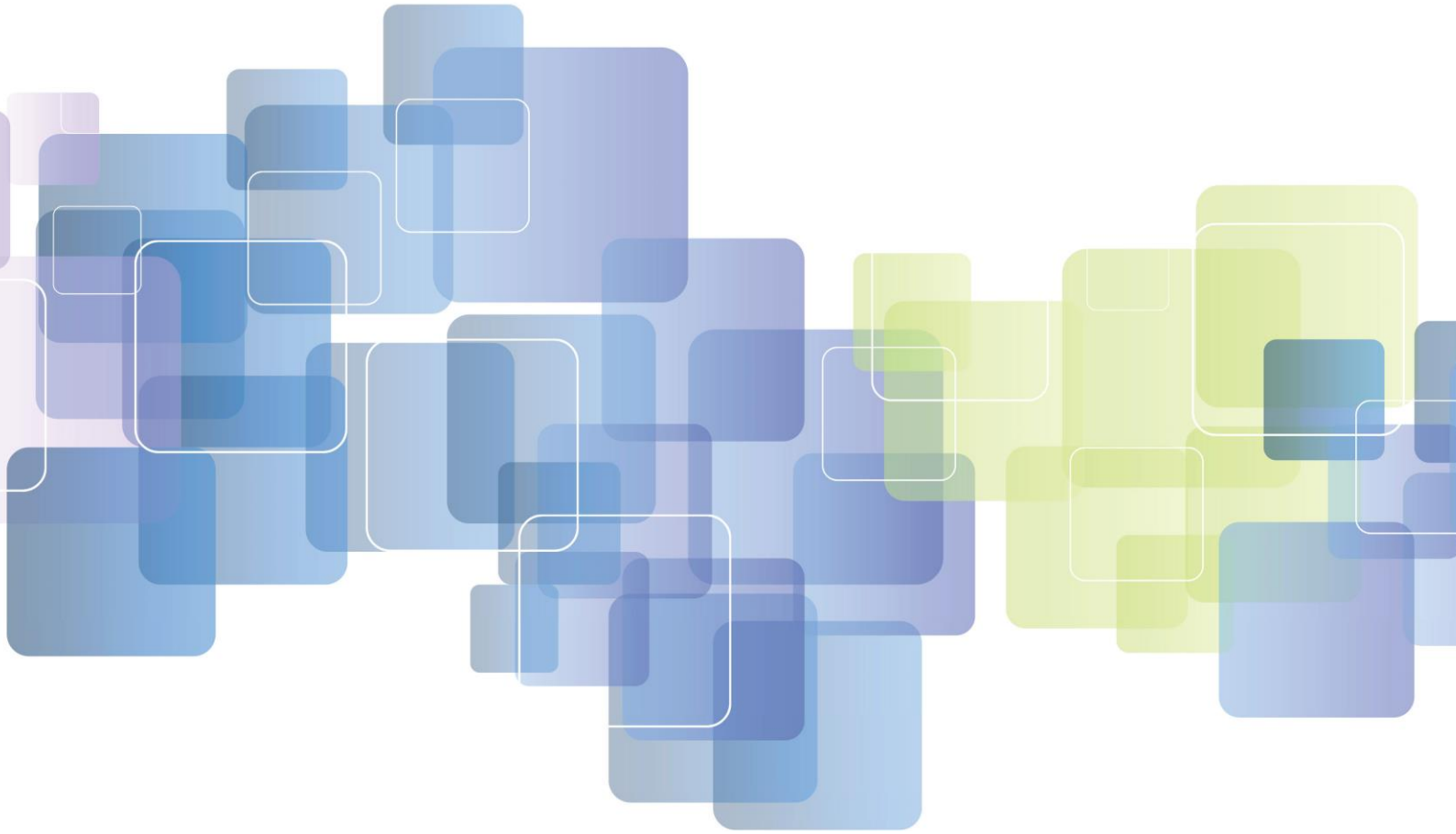
TecnetDati

Suggerimenti per lo Sviluppo delle Applicazioni con PL/SQL

Simona Rotolo



Questo documento, rivolto a chi sviluppa codice in PL/Sql, è stato redatto al fine di fornire degli standard di sviluppo che aiuteranno a rendere più omogenee tra loro e più potenti le stored procedures.
Gli standard forniti qui di seguito sono raggruppati in due categorie che vengono così definite:

Coding standard

Al fine di rendere più leggibile e chiaro il contenuto del codice Pl/Sql a tutti coloro che dovranno effettuarne la manutenzione viene fornito l'elenco dei principali standard:

Forma

Il codice di una procedura diventa più trasparente a chi lo deve modificare se vengono inseriti dei commenti ogni volta che si scrive una serie di istruzioni. Ogni commento dovrebbe fornire delle informazioni sulla logica applicativa che ha dato origine a ciascuna parte di codice.
Es:

```
...
begin

select count(*)
into var_righe_ordine
from tab_righe_ordine
where cod_ordine = 123;

if var_righe_ordine = 0 then
/*
    Se il numero delle righe ordine è 0, l'ordine non può essere accettato.
    Visualizzare l'errore
*/
     dbms_output.put_line(' Inserire almeno un articolo nell'ordine ');
else
 -- registro l'ordine -
     begin
     insert ...
     end;
end if;
...
```

Cercare di scrivere il codice in maniera ordinata ed evitare di creare procedure troppo lunghe.
Fare delle procedure a parte per il codice ricorrente (code modularity).
La modularità è tra i principali concetti dell'object oriented, consiste nello strutturare un'applicazione in moduli indipendenti, che permettono di trasformare in parti più

semplici, sistemi molto complessi. Una volta che l'applicazione è divisa in più componenti, ciascuno di essi può essere riutilizzato in futuro da altre applicazioni e può essere facilmente mantenuto.
Dividere la procedura in vari blocchi distribuendo così la logica che è utilizzata al suo interno.

Dichiarazione variabili

Cercare di non dichiarare le variabili con nomi uguali a identificatori di oggetti utilizzati all'interno della stessa procedura(ad es con nomi di tabelle). Pl/sql compila correttamente ma il risultato sperato non è assicurato.

Es:

```
Procedure Errori_Sistema_Pr(msg_no in number) is
   Cursor c_msg is
   Select msg_text
   From system_message
   Where msg_no = msg_no;

   Begin
      ...
   End;
```

In questo caso la compilazione non dà errori, ma a runtime potrebbero verificarsi dei risultati indesiderati, visto che il parametro msg_no potrebbe essere visto come la colonna msg_no.
Assegnare sempre dei nomi significativi.

	Prefisso	***Esempio***
Variabili locali	Var_	Var_TipoTitolo
Variabili globali	Var_G_	Var_G_Salvato
Parametri	P_	P_codCliente
Cursori	Cur_	Cur_Impiegati
Record	Rec	Rec_saldi

Le variabili "indice" in un ciclo possono essere descritte da una sola lettera.
Es.

```
for j in 1..100 loop
```

Per maggiore leggibilità del codice, conviene inserire una sola dichiarazione per riga. Le variabili locali è meglio inizializzarle al momento della dichiarazione, fatta eccezione per i casi in cui i valori iniziali sono diversi.

Dichiarazione Costanti

Tutte le costanti devono essere scritte in maiuscolo. Se il nome della costante è formato da più parole, utilizzare come separatore il carattere underscore "_".
Es. ANNO_ATTUALE CONSTANT number := 2000;

Tipo delle variabili

Il nome di una variabile dovrebbe riflettere lo scopo al quale è destinata. Per es:
Declare
Var_ Cod_Cli Cliente.Codice_Cliente**%type;**
Var_ Nome_Cli Cliente.Nome_Cliente**%type;**
Ind Number(2);
Begin
Utilizzare per quanto possibile la notazione % Type, in questo modo si evita anche di incorrere in errori di ***value error*** durante l'esecuzione della procedura nel momento in cui viene cambiato il tipo o aumentata la lunghezza del campo della tabella a cui fa riferimento la variabile.
Questa notazione è da seguire anche per la definizione dei parametri.

Inconsistenza tra versioni di PL/SQL

Fare attenzione alla differenza di comportamento del Pl/Sql e in generale delle funzioni e utility di Oracle, in ambienti diversi o con versioni diverse.
Ad esempio il sql*Loader in ambiente dos per la versione 8.0 di Oracle si esegue col comando
Sqlldr80 ....
Mentre in ambiente Unix si esegue con il comando
Sqlload ....

Nomenclatura Stored Procedures

Cercare di utilizzare una nomenclatura standard nella definizione delle procedure, aggiungendo prefissi e/o suffissi nel nome in modo tale da facilitarne il riconoscimento del tipo (funzione, procedura o package) e della posizione (all'interno di un package, di una libreria di Forms 4.5, ecc.).

Tipo di Procedura	***Prefisso***	***Suffisso***
Funzione		Fn
Procedura		Pr
Package		PcK
Pubblica		
Privata	Prv	

Esempi
RegistroOperazioni_Pr Procedura pubblica
TipoTitoli_Pck Package pubblico
Prv_Gettitolo_Fn Funzione contenuta all'interno di un package,non richiamabile dall'esterno

Statement di inserimento

Nello statement di inserimento specificare SEMPRE le colonne di cui si sta passando il valore, così si evita la generazione di errori nel momento in cui cambia l'ordine o il numero di colonne di una tabella.

Tuninig standard

Consigli al fine di rendere più efficiente e potente il codice Pl/sql.

Cursori

I cursori eslpiciti sono più efficienti di quelli impliciti, anche se quest'ultimi sono più leggibili.
Es:
cursore esplicito:

```
Declare
Cursor Cur_Cognomi is
   Select Citta
   From Clienti
   Where cognome_nome = 'Rossi Gino';

begin
   ....
```

cursore implicito:

```
..
begin
   Select citta
   Into var_Citta
   From clienti
   Where cognome_nome = 'Rossi Gino';

   if var_citta = 'TO' then
   ..
end ;
```

Potenzialità del PL/SQL

Sfruttare le potenzialità del Pl/Sql utilizzando i package e le funzioni che Oracle mette a disposizione, come ad esempio il package UTL_FILE per gestire i file, il Native Dynamic Sql per scrivere Sql dinamico e molti altri.

Group By

La group by esegue implicitamente un sort dei dati, evitare di accompagnarla con un order by quando possibile.
Es:

```
select depno, max(num_emp) as tot_impiegati
from department
group by depno;
```

result:

```
depno tot_impiegati
-----------------------
1               15
2               30
3               10
```

Trigger

Identificare i trigger che saranno necessari, durante il disegno del database. Utilizzarare gli identificatori delle operazioni DML forniti da Oracle piuttosto che creare troppi trigger per una stessa tabella.

Es:

```
Create or Replace Trigger Trg_TipoTitolo
After Insert Or Update Or Delete
On Titoli for each Row
Declare
..
Begin
 If INSERTING
     :new.Scost_Num:= :Old.Scost_num + 1;
 elsif UPDATING
     ..

 else
     ..
 end if;
```

Il trigger viene compilato tutte le volte che viene eseguito quindi conviene scrivere il codice in una procedura in quanto è precompilata e il codice del trigger si riduce ad un'unica riga di comando.
Non utilizzare i trigger per fare dei controlli che possono essere fatti utilizzando le DDL di Oracle, come ad esempio controlli sull'integrità referenziale dei dati.

Colonne con Default

Fare particolare attenzione alle colonne che hanno dei default in quanto questo valore è assegnato solo nel caso in cui la colonna NON è specificata nello statement di inserimento e non quando viene passato come valore NULL.

Procedure Overlaying

Se si utilizzano delle procedure overlaying con stesso identificativo ma con parametri diversi e/o codice al loro interno diverso, metterle in un unico package per facilitarne il match dei parametri quando vengono richiamate.

Keep Shared Pool

Quando viene richiamata una stored procedure, questa risiede nella shared buffer pool dove viene controllato il codice e solo alla fine dell'esecuzione viene liberata la memoria.
Tenere in memoria per quanto possibile le procedure che vengono eseguite spesso, utilizzando il package:
execute *dbms_shared_pool.keep*(Pg_Controllo.Tipo_Titolo_Prc)

Package

Non creare package troppo grossi altrimenti si possono avere problemi in compilazione. Si possono scatenare errori del tipo: UNABLE TO EXTEND ROLLBACK SEGMENT.
Dalla versione 8 di Oracle non dovrebbero più esserci problemi, in quanto si è passati da circa 3000 linee a circa 6,000,000 di linee di codice.
La grandezza dei package riguarda anche la complessità delle query in essi contenute, un package complesso richiederà più memoria di uno, magari con più righe di codice, ma che contiene query più semplici. In compilazione si potrebbe scatenare l'errore "program too large", per evitarlo basta dividerlo in package più piccoli.

Tecnet Dati s.r.l.
C.so Svizzera 185 –
10149 - Torino (TO), Italia
Tel.: +39 011 7718090 Fax.: +39 011 7718092
P.I. 05793500017 C.F. 09205650154
www.tecnetdati.com