Tipo Boolean in Oracle – Una piccola digressione

Due parole sul tipo Boolean

In questo post andremo ad affrontare un argomento particolare: parleremo di questo tipo Boolean sotto Oracle.

Esiste il tipo Boolean?

Oracle mette a disposizione il tipo Boolean, ma occorre tenere conto di queste precisazioni:

- NON E' usabile nella definizione dei campi della tabella.
- PUO' essere usato nel PLSQL.

Se vogliamo utilizzarlo nella definizione di una tabella, come suggerito da Tom Kyle, conviene sostituirlo con il tipo ***CHAR(1)***, che presenta le stesse caratteristiche, come mostrato di seguito:

```
flag char(1) check (flag in ( 'Y', 'N' )),
```

Il risultato è il medesimo.

Una precisazione

Anche se esiste come ***datatype***, non è possibile usarlo ovunque. ***PL-SQL ha le sue regole*** e chi lo usa deve mettersi in testa che si devono seguire. Non si può pretendere che, se altri linguaggi (quali ad es. JAVA) lo consentono, PL-SQL si deve adeguare.

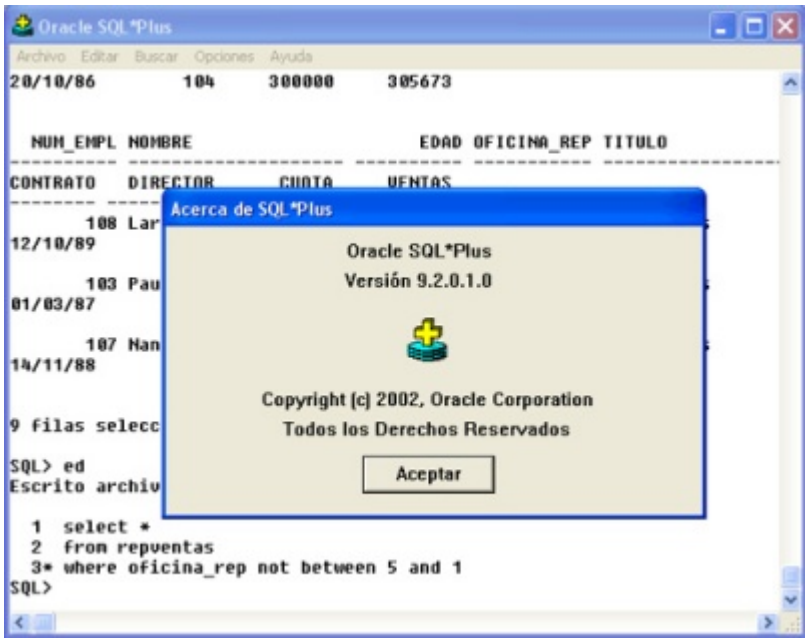
Conclusione

Abbiamo visto un piccolo tassello, ma di grande importanza. Questo ci ricorda che ogni linguaggio ha le sue regole e i suoi ambiti di applicazione. Dobbiamo tenerne presente sempre.

Esportazione dati via SQLPLUS

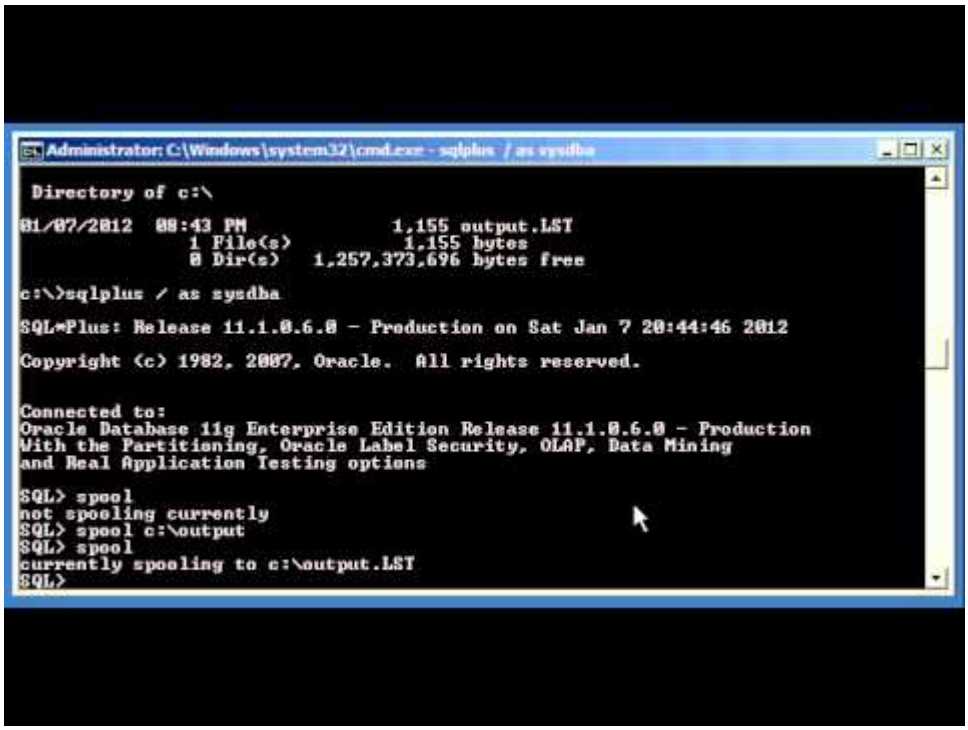
Un suggerimento

In questo post, andremo a vedere un piccolo ***trucchetto*** che ho usato per poter esportare dei dati, sopratutto quando si tratta di esportare dati di una certa mole.



Andiamo nel dettaglio

L’idea è di fare uso di ***SQLPLUS***, che si presta bene a questo genere di lavori. sfruttando le sue capacità. In particolare ci appoggiamo alla possibilità di poter inviare su file il risultato della esecuzione della query. Stiamo parlando del comando: ***SPOOL***;



Andiamo nel dettaglio e vediamo come fare.

Una volta definita la query, con tutti i dati che sono necessari. Quindi componiamo la query come segue:

```
Select <campo1> || <campo2> || ..... || <campon> from .....
```

oppure come

```
Select <campo1> || <delimitatore> || <campo2> ||
<delimitatore> || ..... || <campon> from .....
```

a seconda che si voglia avere una estrazione dove i campi sono a lunghezza fissa (primo caso) o con delimitatore (secondo caso).

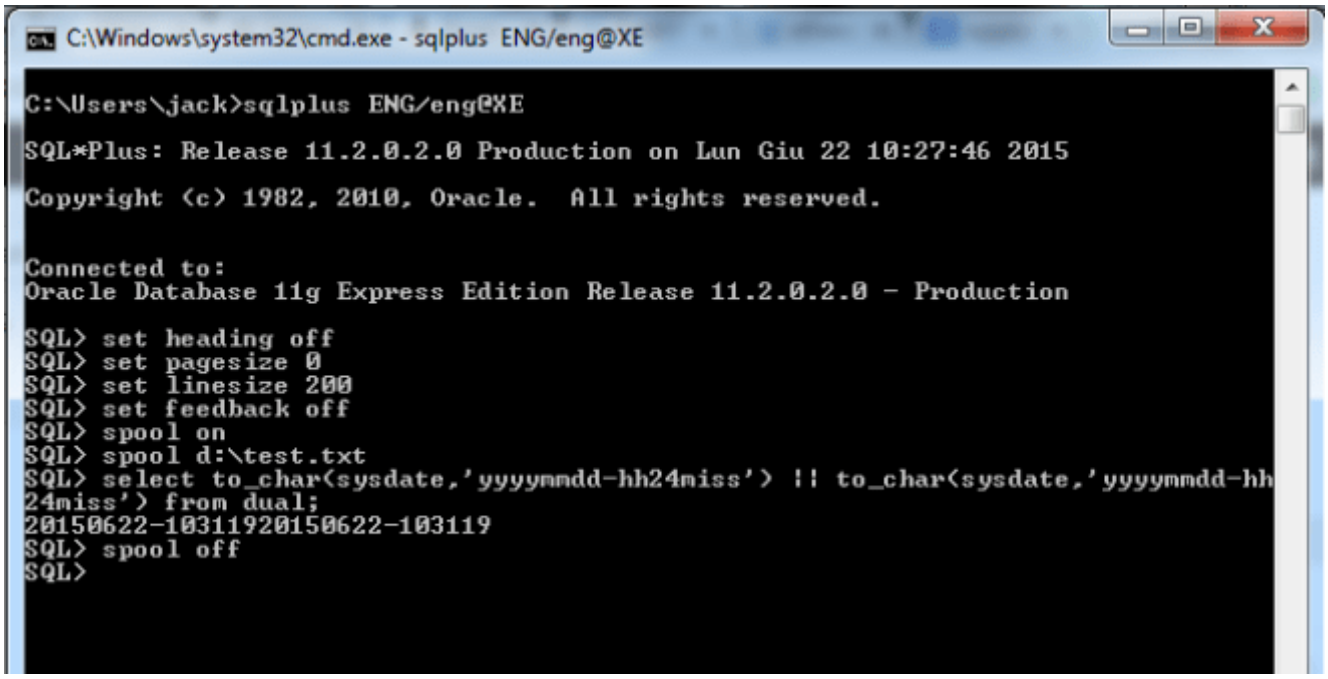
A questo punto, vediamo come procedere per avere l’estrazione vera e propria.

Attiviamo SQLPLUS, da riga di comando:

```
sqlplus <utente>/<password>@<dbsid>
```

Quindi forniamo la sequenza di comandi:

```
set heading off
set pagesize 0
set linesize 200
set feedback off
Spool on
Spool <file_destinazione>.txt
<scriviamo la Query, mettendo il ';' finale >
Spool off <da lanciare al termine della esecuzione della query>
```



La precedente figura da una idea per capire come procedere. Ovviamente mi sono limitato a eseguire una query di esempio per mostrare il funzionamento.

Conclusioni

Abbiamo un sistema molto semplice per eseguire delle estrazioni dati molto semplici. Il metodo è automatizzabile, creando degli script ad hoc, per semplificarci ulteriormente la vita :-).

Creare una funzione split con plsql

Funzione SPLIT in PL-SQL

Riporto un sistema molto semplice per realizzare una funzione SPLIT in PL-SQL. Si tratta di una procedura che ho trovato in questo blog.

Si tratta di una funzione che, data una stringa con separatori, restituisce l'elemento i-esimo.

```
create or replace function get_token(
   the_list  varchar2,
   the_index number,
   delim     varchar2:= ','
)
   return varchar2 is
   start_pos number;
   end_pos   number;
begin
```

```
    if the_index = 1 then
        start_pos := 1;
    else
        start_pos := instr(the_list, delim, 1, the_index - 1);
        if start_pos = 0 then
           return null;
        else
            start_pos := start_pos + length(delim);
        end if;
    end if;

    end_pos := instr(the_list, delim, start_pos, 1);

    if end_pos = 0 then
        return substr(the_list, start_pos);
    else
          return substr(the_list, start_pos, end_pos -
 start_pos);
    end if;

 end get_token;
 /
```

Si tratta di una funzione molto semplice. Il risultato è il seguente:

```
 select
    get_token('foo,bar,baz',1), -- 'foo'
    get_token('foo,bar,baz',3), -- 'baz'
    --
    get_token('a,,b',2),        -- '' (null)
    get_token('a,,b',3),        -- 'b'
    --
    get_token('a|b|c',2,'|'),   -- 'b'
    get_token('a|b|c',4,'|')    -- '' (null)
 from
    dual
```

Punti di attenzione

- L’indice parte da 1 e non da 0
- Se per l’indice specificato, non è presente alcun valore, allora non viene restituito alcun valore
- Se si fornisce un indice che non esiste (es. la stringa è formata da 3 elementi e si richiede il 4 elemento), allora anche in questo caso, la funzione non restituisce nulla.

Suggerimenti

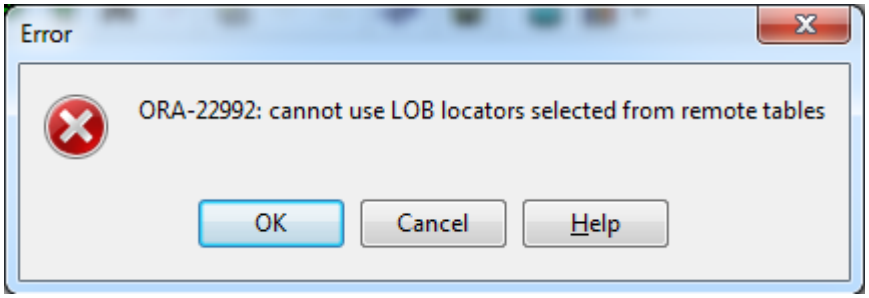
Potrebbe tornare utile far restituire una stringa ad hoc, qualora non venga fornito un indice non esistente nella stringa, sostituendo il ***return null;*** con ***return ‘-1’;*** in modo da distinguere la situazione in cui non venga restituito un valore da l’aver richiesto un indice non esistente.

Campo CLOB via DB-LINK

Vi sarà capitato di ….

…. dover gestire delle tabelle via DB-LINK. Ma a quanti sarà capitato di dover lavorare via db-link con tabelle che presentano dei campi LOB? Magari con una versione un pò più datata di Oracle (Oracle 9, per intenderci)?

Di sicuro ci si sarà scontrati con l'errore:



semplicemente lanciando una query del tipo:

Select * from tabella@dblink.

Oracle non consente di poter eseguire la select diretta, come riportato nel seguente articolo di AskTom, ma sono possibili altre operazioni molto interessanti. Non potendo leggere da remoto, possiamo portare le informazioni in locale attraverso due possibili strade:

1. ***INSERT INTO….. SELECT * FROM TABELLA@dblink***
2. ***CREATE TABLE ….. AS SELECT * FROM TABELLA@dblink***

Il primo sistema presuppone la presenza di una tabella destinazione, mentre con il secondo sistema, andiamo fisicamente a generare la tabella in locale. Una volta portati i dati in locale, possiamo fare qualsiasi operazione.

Controindicazioni?

Ovviamente se la tabella remota presenta un numero di record molto alto, conviene impostare delle condizioni di filtro alla query su tabella remota, in modo da non portare troppi record.

Esiste una funzione isNumeric in PLSQL??

Potrebbe sorgere la necessità di dover leggere dei valori solo numerici, memorizzati su di un campo alfanumerico. I motivi per cui si può avere questa necessità possono essere svariati.

Al momento in cui scrivo, una funzione ***isNumeric*** o similare non esiste in PLSQL. Come possiamo realizzare questo? Ci viene in aiuto il sito di AskTom, che ci fornisce alcuni spunti.

Abbiamo due strade:

Funzione Translate….

La funzione Translate può risultare molto utile. Come indicato in questa pagina, la funzione esegue una sostituzione di caratteri basandosi su delle corrispondenze. Di conseguenza, un sistema per ottenere il risultato desiderato, si può usare la seguente istruzione:

replace(translate(<valore>, ‘0123456789’, ‘0000000000’), ‘0’, ”)

Spieghiamo il funzionamento: La translate, fondamentalmente, esegue una sostituzione delle cifre numeriche con ‘0’ (zero). La seconda funzione replace, banalmente, elimina lo ‘0’ (zero).

Conseguenza? Se ***<valore>*** contiene delle cifre numeriche, queste sono eliminate e viene restituita una stringa vuota. In alternativa, se su ***<valore>*** è presente un carattere alfanumerico, questo rimane e viene restituito nella stringa.

.

REGEX – Espressioni Regolari

E’ possibile anche utilizzare le espressioni regolari, per riuscire ad identificare i valori solo numerici. Se la versione di Oracle è la 10g o successive, la funzione ***REGEXP_LIKE*** può essere utilizzata. Di seguito un esempio:

SELECT col1 FROM t1 WHERE REGEXP_LIKE(col1, ‘[[:digit:]]’);

Basta semplicemente impostare le regole, ed il gioco è fatto.

Conclusioni

Si tratta di due soluzioni più che valide, ma la prima risulta essere molto semplice, veloce, pratica ed usabile in vari contesti. La seconda risulta più pesante.

Scrivere file con Oracle

Oggi tratteremo un argomento molto semplice. Si tratta della gestione dei file da parte di Oracle.

Oracle consente di poter leggere/scrivere dei file attraverso un package standard: **UTL_FILE;** che mette a disposizione tutti i metodi necessari per poter eseguire qualsiasi operazione sui file.

> ***Una piccola raccomandazione: Prima di poter utilizzare il package, assicurarsi che l'utenza funzionale, cui impostare la stored procedure/package di utilizzo dei file, disponga delle grant di execute per poter usare il package. Se non dispone di tali grant, chiedere all'amministratore del Database di fornirle.***

Come prima cosa, occorre indicare ad Oracle dove si intende andare ad eseguire le operazioni di lettura/scrittura dei file. Per far ciò, occorre definire una DIRECTORY, ovvero indicare ad Oracle dove eseguire le varie operazioni. Per far ciò, si utilizza il seguente comando:

> ***CREATE OR REPLACE DIRECTORY TMP_DIR AS '<path completo della directory>';***

Attraverso questa istruzione si indica la directory (locale al server dove è installato il DB Oracle), dove sarà possibile eseguire le varie operazioni di lettura/scrittura dei file. E' bene ribadire che le operazioni di lettura/scrittura sono eseguite sulla macchina locale e non su dei server remoti. Se si ha la necessità di poter scrivere su dei server remoti, occorre disporre di altre procedure, che magari sfruttando il

protocollo FTP, possano scrivere i file su server remoti. Package e procedure sono disponibili e saranno trattati in un altro post.

> ***Si segnala altresì che la directory non appartiene ad uno schema specifico. Occorre quindi prestare attenzione al nome della directory da usare, in quanto è univoco per l'intero database.***

Una volta che a directory è stata creata, adesso usiamo i metodi del package per scrivere un semplice file:

```
/*
* Codice di esempio di scrittura di un file
*/

DECLARE

lv_nome_file VARCHAR2(2000);
lv_percorso_file VARCHAR2(2000);
lv_file_id UTL_FILE.file_type;

BEGIN

lv_percorso_file := '<directory_definita>';
lv_nome_file := '<nome_file_da_scrivere>;

-- Apertura del file
lv_file_id := UTL_FILE.fopen( lv_percorso_file, lv_nome_file,
'w' );

-- Scrittura della riga nel file
UTL_FILE.put_line(file => lv_file_id , buffer => 'Questo è un
esempio di testo scritto su di un file' );

-- Chiusura del file.
UTL_FILE.fclose(file => lv_file_id );

EXCEPTION
WHEN OTHERS THEN
```

```
DBMS_OUTPUT.PUT_LINE('Errore – ' || TO_CHAR(SQLCODE) || ' – '
|| SUBSTR(SQLERRM,1,150));
END;
```

Analogamente, la lettura di un file viene eseguita con un codice similare e sfruttando gli stessi metodi.

```
/*
* Codice di esempio di lettura di un file
*/

DECLARE
lv_nome_file VARCHAR2(2000);
lv_percorso_file VARCHAR2(2000);
lv_file_id UTL_FILE.file_type;
lv_text VARCHAR2(32767);
BEGIN

lv_percorso_file := '<directory_definita>';
lv_nome_file := '<nome_file_da_scrivere>;;

lv_file_id  :=  UTL_FILE.fopen(  lv_percorso_file,
 lv_nome_file, 'r', 32767);

BEGIN

LOOP

UTL_FILE.get_line(lv_file_id, lv_text, 32767);
DBMS_OUTPUT.put_line('Line: |' || lv_text || '|');
END LOOP;
EXCEPTION
WHEN NO_DATA_FOUND THEN
NULL;
END;

DBMS_OUTPUT.put_line('Line : |' || lv_text || '|');

UTL_FILE.fclose(lv_file_id);
```

END;

Come si vede il package è molto semplice e consente di poter eseguire tutte le operazioni di cui si abbisogna senza grandi difficoltà.

URL di riferimento

- Semplice articolo in inglese su UTL_FILE
- Indicazioni sulle Directory di Oracle sono presenti su questo link e su questa URL.

Numero massimo di JOB che possono essere eseguiti sotto Oracle

Sarà comunicato a molti che, lanciando dei Jobs, questi non venissero eseguiti. Nessuna indicazione, nessun failure, niente di niente. Quindi? che cosa causa il problema?

Semplice. Oracle imposta un numero massimo di JOB che possono essere eseguiti. Se si vuole eseguire un numero maggiore, occorre controllare un parametro:

JOB_QUEUE_PROCESSES

Eseguendo la query:

Select * from v$parameter

where name like '%job%'

è possibile visionare il valore impostato.

La modifica del valore viene eseguita attraverso una ***ALTER SYSTEM***, come mostrato di seguito.

alter system set job_queue_processes=10

Potete trovare un pò di informazioni aggiuntive nel seguente link.

Una piccola annotazione. Per poter eseguire l'istruzione di Alter System, occorre che l'utenza disponga delle corrispondenti grant. Fare riferimento al seguente link per ulteriori delucidazioni

---

Come realizzare la funzione di SLEEP in PLSQL

In alcune circostanze, occorre impostare dei ritardi nella procedura che deve essere eseguita. Fino a qualche tempo fa avevo risolto attraverso un ciclo, eseguito un tot di volte, in modo da simulare un ritardo.

Consultando il prolifico sito di Ask Tom, ho scoperto che esiste una funzione che realizza il tutto. Si tratta di:

DBMS_LOCK.SLEEP(<numero di secondi>)