Con le ultime versioni del suo database, Oracle ha cercato di migliorare le funzionalità per il data warehousing. Grandi implementazioni sono state apportate al linguaggio SQL

Analytic

Statistiche sui dati con SQL in Oracle9i

mruocchio@programmers.net di Massimo Ruocchio

È laureato in matematica ed è certificato Oracle Application Developer. Si occupa di analisi, progettazione e sviluppo di applicazioni software

Un problemino semplice semplice: in una tabella avete dieci record, ogni riga riporta il nome di uno studente e la media dei suoi voti. Dovete redigere la classifica degli studenti per media dei voti con un'istruzione SQL. Come fate a calcolare la posizione di ogni studente? Se state pensando alla pseudocolonna ROWNUM riflettete sul caso di due studenti con la stessa media, che devono avere la stessa posizione in classifica.

In quest'articolo vogliamo parlare un po' di Data Warehouse e vedere come Oracle ha migliorato il linguaggio SQL per consentire agli sviluppatori di risolvere problemi come quello cui abbiamo appena accennato.

Ma cos'è un Data Warehouse?

Un data warehouse è un database relazionale che non viene progettato per conservare i dati necessari ad una classica applicazione transazionale, bensì per effettuare analisi di tipo statistico sulle informazioni che contiene. La differenza tra il database utilizzato in un classico sistema OLTP (OnLine Transaction Processing) ed un database utilizzato come data warehouse non è nei dati, ma in come questi sono strutturati. In genere un database classico è progettato in terza forma normale (va bene, va bene correggo: dovrebbe essere in terza forma normale ma…) mentre un data warehouse è denormalizzato (ma senza esagerare!) perché quello che più conta sono le prestazioni ed i dati sono abbastanza statici.

Per analizzare i dati presenti in un data warehouse si possono utilizzare dei tool già pronti disponibili in commercio, oppure è possibile scrivere da sé un software utilizzando, ad esempio, SQL e PL/SQL. Per invogliare gli utenti ad utilizzare i propri database come data warehouse, Oracle ha pensato bene di potenziare il linguaggio SQL con funzionalità molto avanzate per l'analisi dei dati. Nei paragrafi che seguono incontreremo alcune di queste nuove funzioni, dette *funzioni analitiche*. Tutti gli esempi sono basati su un semplice schema di prova. Le istruzioni SQL per creare le tabelle e popolarle di record di prova e tutte le query di questo articolo si trovano sul sito ftp di Infomedia. Ci sono quattro tabelle, le cui strutture sono evidenziate nel **Listato 1**. Nelle tabelle ci sono: dieci studenti, dieci materie d'esame, sette docenti e cento esami superati. Ogni studente ha superato tutti gli esami. Vogliamo analizzare questi dati facendo ricorso a funzioni SQL introdotte da Oracle nella release 2 di Oracle8i oppure in Oracle9i. Tutte le query eseguite ed i risultati ottenuti sono visualizzati nel **Listato 2**.

Facciamo una classifica

Cominciamo subito con il problema esposto ad inizio articolo. Vogliamo tirare fuori la classifica degli studenti per media dei voti, gestendo correttamente le eventuali parità. Eseguiamo l'istruzione che segue:

```
select studente, avg(voto) media,
       rank () over(order by avg(voto) desc) posizione
  from esami
group by studente;
```

La posizione è determinata utilizzando la funzione RANK. Una volta specificato di volere la posizione (il rank) bisogna dire rispetto a quale insieme di dati va calcolato. L'insieme di dati si determina con la parola chiave OVER. In realtà non abbiamo bisogno di specificare alcun insieme particolare di dati, vogliamo una classifica assoluta su tutti gli studenti, quindi nella clausola OVER ci limitiamo ad indicare che i dati

LISTATO 1 Struttura dello schema utilizzato per gli esempi

```
SQL> desc studenti
   Nome                      Nullo?       Tipe
   ------------------------- ------------ -------------
   NOME                                   VARCHAR2(10)
   COGNOME                                VARCHAR2(10)
   MATRICOLA                              NUMBER(5)

SQL> desc materie
   Nome                      Nullo?       Tipe
   ------------------------- ------------ -------------
   CODICE                                 NUMBER(5)
   DESCRIZIONE                            VARCHAR2(30)

SQL> desc esami
   Nome                      Nullo?       Tipe
   ------------------------- ------------ -------------
   STUDENTE                               NUMBER(5)
   MATERIA                                NUMBER(5)
   DOCENTE                                NUMBER(5)
   VOTO                                   NUMBER(2)
   LODE                                   CHAR(1)

SQL> desc docenti
   Nome                      Nullo?       Tipe
   ------------------------- ------------ -------------
   MATRICOLA                              NUMBER(5)
   COGNOME                                VARCHAR2(10)
   DIPARTIMENTO                           VARCHAR2(11)
```

LISTATO 2 Tutti i risultati delle query d'esempio

```
SQL> desc studenti
SQL> /* Classifica degli studenti per media di voti */
SQL> select studente, avg(voto) media,
  2  rank () over(order by avg(voto) desc) posizione
  3  from esami group by studente;

  STUDENTE      MEDIA  POSIZIONE
 --------- ---------- ----------
     14856       29,1          1
     15656       27,8          2
     14877       27,3          3
     13268       27,2          4
     13769       26,7          5
     12248       26,4          6
     12329       26,1          7
     14670       26,1          7
     13557       25,6          9
     12198       23,6         10

Selezionate 10 righe.

SQL> /* Classifica densa degli studenti per media di voti */
SQL> select studente, avg(voto) media,
  2  dense_rank () over(order by avg(voto) desc) posizione
  3  from esami group by studente;

  STUDENTE      MEDIA  POSIZIONE
 --------- ---------- ----------
     14856       29,1          1
     15656       27,8          2
     14877       27,3          3
     13268       27,2          4
     13769       26,7          5
     12248       26,4          6
     12329       26,1          7
     14670       26,1          7
     13557       25,6          8
     12198       23,6          9

Selezionate 10 righe.

SQL>
SQL> /* In caso di pari media ha la precedenza
                               chi ha avuto piu' lodi */
SQL> select studente, avg(voto) media,
                          sum(decode(lode,'S',1,0)) Lodi,
  2  rank () over(order by avg(voto) desc,
                 sum(decode(lode,'S',1,0)) desc) posizione
  3  from esami group by studente;

  STUDENTE       MEDIA       LODI   POSIZIONE
 --------- ---------- ---------- ----------
     14856        29,1          2          1
     15656        27,8          1          2
     14877        27,3          0          3
     13268        27,2          1          4
     13769        26,7          0          5
     12248        26,4          1          6
     12329        26,1          1          7
     14670        26,1          0          8
     13557        25,6          0          9
     12198        23,6          0         10

Selezionate 10 righe.

SQL>
SQL> /* Migliori 3 studenti (mostrare nome e cognome)
                                                    per media */
SQL> select posizione, s.nome, s.cognome, x.media
  2  from studenti s,
  3  (select studente, avg(voto) media,
  4  rank () over(order by avg(voto) desc) posizione
  5  from esami group by studente) x
  6  where s.matricola = x.studente
  7  and posizione <= 3
  8  order by posizione;

 POSIZIONE NOME       COGNOME         MEDIA
---------- ---------- ---------- ----------
         1 Massimo    Esposito         29,1
         2 Marta      Pieri            27,8
         3 Anna       Cassini          27,3

SQL>
SQL> /* Posizione ipotetica di uno studente con media 27.5 */
SQL> select  rank (27.5) within group
                          (order by media desc) posizione
  2  from (select studente, avg(voto) media
                           from esami group by studente);

 POSIZIONE
----------
         3

SQL>
SQL> /* Classifica delle materie per voto,
                         evidenziando il dipartimento */
SQL> select d.dipartimento, m.descrizione, avg(voto) media,
  2  rank () over(order by avg(voto) desc) posizione
  3  from esami, docenti d, materie m
  4  where d.matricola = docente
  5  and m.codice = materia
  6  group by dipartimento, descrizione;

DIPARTIMENT   DESCRIZIONE                MEDIA   POSIZIONE
-----------   ----------------------  --------  ----------
Fisica        Fisica I                   28,5           1
Fisica        Fisica II                  27,4           2
Informatica   Calcolo numerico           27,4           2
Matematica    Geometria I                26,8           4
Informatica   Informatica                26,7           5
Matematica    Meccanica Razionale        26,5           6
Matematica    Geometria II               26,2           7
Matematica    Algebra                    25,7           8
Matematica    Analisi Matematica I       25,5           9
Matematica    Analisi Matematica II      25,2          10

Selezionate 10 righe.

SQL>
SQL> /* Per ogni dipartimento di appartenenza
                                 dei docenti, la classifica
DOC>    delle materie per media di voti */
SQL> select d.dipartimento,
  2       rank () over(partition by dipartimento
  3                    order by avg(voto) desc) posizione,
  4       m.descrizione, avg(voto) media
  5    from esami, docenti d, materie m
  6  where d.matricola = docente
  7    and m.codice = materia
  8  group by dipartimento, descrizione;

DIPARTIMENT  POSIZIONE  DESCRIZIONE                 MEDIA
-----------  ---------  -------------------------- ------
Fisica               1  Fisica I                    28,5
Fisica               2  Fisica II                   27,4
Informatica          1  Calcolo numerico            27,4
Informatica          2  Informatica                 26,7
Matematica           1  Geometria I                 26,8
Matematica           2  Meccanica Razionale         26,5
Matematica           3  Geometria II                26,2
Matematica           4  Algebra                     25,7
Matematica           5  Analisi Matematica I        25,5
Matematica           6  Analisi Matematica II       25,2

Selezionate 10 righe.

SQL> select studente, avg(voto) media,
  2  rank () over(order by avg(voto) desc) posizione,
  3  round(avg(avg(voto)) over(order by avg(voto) desc
  4  rows between 2 preceding and 2 following),2) media5
  5  from esami group by studente;
```

(segue a pagina seguente)

segue LISTATO 2

```
   STUDENTE      MEDIA   POSIZIONE     MEDIA5
---------- ---------- ---------- ----------
      14856       29,1          1      28,07
      15656       27,8          2      27,85
      14877       27,3          3      27,62
      13268       27,2          4      27,08
      13769       26,7          5      26,74
      12248       26,4          6       26,5
      12329       26,1          7      26,18
      14670       26,1          7      25,56
      13557       25,6          9      25,35
      12198       23,6         10       25,1

Selezionate 10 righe.

SQL>
SQL> /* Percentili delle medie degli studenti */
SQL> select studente, avg(voto) media,
  2  cume_dist() over(order by avg(voto) ) cume_dist,
  3  cume_dist() over(order by avg(voto))*100 percentile
  4  from esami group by studente
  5  order by 4 desc;

 \STUDENTE      MEDIA  CUME_DIST PERCENTILE
---------- ---------- ---------- ----------
      14856       29,1          1        100
      15656       27,8         ,9         90
      14877       27,3         ,8         80
      13268       27,2         ,7         70
      13769       26,7         ,6         60
      12248       26,4         ,5         50
      12329       26,1         ,4         40
      14670       26,1         ,4         40
      13557       25,6         ,2         20
      12198       23,6         ,1         10

Selezionate 10 righe.

SQL>
SQL> /* Calcolo dei terzili degli studenti per media voto */
SQL> select studente, avg(voto) media,
  2  ntile(3) over(order by avg(voto) desc) terzile
  3  from esami group by studente ;

   STUDENTE      MEDIA    TERZILE
---------- ---------- ----------
      14856       29,1          1
      15656       27,8          1
      14877       27,3          1
      13268       27,2          1
      13769       26,7          2
      12248       26,4          2
      12329       26,1          2
      14670       26,1          3
      13557       25,6          3
      12198       23,6          3

Selezionate 10 righe.

SQL>
SQL> /* per ogni studente evidenzia la differenza
                  con il piu' bravo e con il meno bravo*/
SQL> select studente, avg(voto) media,
  2  rank () over(order by avg(voto) desc) posizione,
  3  avg(voto) - first_value(avg(voto))
          over(order by avg(voto) desc) "Risp. al primo",
  4  avg(voto) - first_value(avg(voto)) over(order
                    by avg(voto) asc) "Risp. all'ultimo"
  5  from esami group by studente
  6  order by avg(voto) desc;

 STUDENTE   MEDIA  POSIZIONE  Risp. al primo  Risp. all'ultimo
--------  -----  --------  -------------  ---------------
   14856   29,1         1              0              5,5
   15656   27,8         2           -1,3              4,2
   14877   27,3         3           -1,8              3,7
   13268   27,2         4           -1,9              3,6
   13769   26,7         5           -2,4              3,1
   12248   26,4         6           -2,7              2,8
   12329   26,1         7             -3              2,5
   14670   26,1         7             -3              2,5
   13557   25,6         9           -3,5                2
   12198   23,6        10           -5,5                0

Selezionate 10 righe.

SQL>
SQL> /* Per ogni studente evidenzia la differenza con quelli
             che lo precedono e lo seguono di due posti */
SQL> select studente, avg(voto) media,
  2  rank () over(order by avg(voto) desc) posizione,
  3  avg(voto) - LAG(avg(voto),2)
         over(order by avg(voto) desc) "Risp. a 2 prima",
  4  avg(voto) - LEAD(avg(voto),2) over(order by avg(voto)
                                    desc) "Risp. a 2 dopo"
  5  from esami group by studente;

 STUDENTE   MEDIA  POSIZIONE  Risp. a 2 prima  Risp. a 2 dopo
--------  -----  --------  --------------  --------------
   14856   29,1         1                             1,8
   15656   27,8         2                              ,6
   14877   27,3         3            -1,8              ,6
   13268   27,2         4             -,6              ,8
   13769   26,7         5             -,6              ,6
   12248   26,4         6             -,8              ,3
   12329   26,1         7             -,6              ,5
   14670   26,1         7             -,3             2,5
   13557   25,6         9             -,5
   12198   23,6        10            -2,5

Selezionate 10 righe.
```

vanno presi in ordine discendente di media voto. Come è possibile vedere nel **Listato 2**, ci sono due studenti con la stessa media al settimo posto. Lo studente che segue occupa giustamente la nona posizione, mentre l'ottava posizione non viene assegnata. Nel caso in cui si voglia attribuire l'ottava posizione allo studente che segue i due classificati al settimo posto, bisogna utilizzare la funzione DENSE_RANK come segue:

```
select studente, avg(voto) media,
dense_rank () over(order by avg(voto) desc) posizione
   from esami
group by studente;
```

Se si desidera inserire una condizione da valutare a parità di media basta aggiungerla nella clausola OVER. Ad esempio nella nostra classifica vogliamo attribuire, a parità di media, una posizione migliore allo studente che ha avuto più volte la lode:

```
select studente, avg(voto) media,
      sum(decode(lode,'S',1,0)) Lodi,
      rank () over(order by avg(voto) desc,
      sum(decode(lode,'S',1,0)) desc) posizione
   from esami
group by studente;
```

In questa maniera nella nostra classifica non ci sono più situazioni di parità. Per visualizzare solo i migliori tre studenti bisogna includere la classifica completa in una query esterna che effettui una limitazione sulla posizione. Questo perché non è possibile utilizzare la funzione RANK nella clausola WHERE di una query, in quanto la clausola WHERE viene

applicata dal DBMS prima di passare i dati alle funzioni analitiche. Visualizziamo pure nome e cognome degli studenti:

```
select posizione, s.nome, s.cognome, x.media
   from studenti s,
        (select studente, avg(voto) media,
                rank () over(order by avg(voto) desc) posizione
           from esami group by studente) x
   where s.matricola = x.studente
     and posizione <= 3
order by posizione;
```

La funzione RANK può essere utilizzata anche per determinare una posizione ipotetica dato un certo valore. Ad esempio vorremmo sapere che posizione occuperebbe nella nostra classifica uno studente che avesse la media del 27,5. Facciamo come segue:

```
select rank (27.5) within group
      (order by media desc) posizione
from (select studente, avg(voto) media
        from esami
   group by studente);
```

Abbiamo introdotto anche la clausola WITHIN GROUP che permette di specificare con quale gruppo di record va confrontato il valore 27,5. Nel nostro caso il gruppo è unico, costituito da tutti i record presi in ordine decrescente di media.

Modifichiamo un po' la classifica analizzando le materie anziché gli studenti:

```
select d.dipartimento, m.descrizione, avg(voto) media,
           rank () over(order by avg(voto) desc) posizione
      from esami, docenti d, materie m
   where d.matricola = docente
         and m.codice = materia
group by dipartimento, descrizione;
```

Nel prossimo paragrafo vedremo come suddividere i dati in partizioni e finestre.

Partizioni e finestre

Dopo il primo approccio con le funzioni analitiche riflettiamo un momento sulla vera innovazione introdotta da Oracle con questo strumento.

Nel linguaggio SQL classico ogni riga restituita da una query non conosce, e non può utilizzare, i dati presenti in altre righe. Le funzioni analitiche vengono applicate dopo avere valutato le clausole WHERE, GROUP BY ed HAVING, quindi hanno a disposizione tutti i dati pronti per essere messi in output. In quella fase è possibile anche tenere conto dei dati di altre righe. Le righe date in pasto alle funzioni analitiche possono essere, su richiesta dell'utente, suddivise in partizioni. Ripartiamo dall'ultima query eseguita: era una classifica delle materie per media dei voti. Per ogni materia avevamo evidenziato il dipartimento del docente. Adesso vogliamo una classifica delle materie all'interno del dipartimento; facciamo come segue:

```
select d.dipartimento,
           rank () over(partition by dipartimento
              order by avg(voto) desc) posizione,
           m.descrizione, avg(voto) media
      from esami, docenti d, materie m
   where d.matricola = docente
         and m.codice = materia
group by dipartimento, descrizione;
```

Rispetto a prima c'è una sola differenza nella sintassi: all'interno della clausola OVER abbiamo utilizzato la parola chiave PARTITION per dividere i dati in partizioni, una partizione per ogni dipartimento. La classifica viene calcolata, ed azzerata, per ogni partizione/dipartimento. I risultati sono nel **Listato 2**.

Un altro concetto che bisogna spiegare quando si parla di funzioni analitiche è quello di finestra. Per ogni riga in una partizione, una finestra è un intervallo di righe che includano la riga corrente. Ovviamente una finestra può essere grande quanto l'intera partizione. Per definire una finestra se ne può dichiarare l'ampiezza in righe. Ad esempio per ogni studente si vuole calcolare la media delle medie in una finestra, ampia cinque record, centrata nella riga corrente:

```
select studente, avg(voto) media,
rank () over(order by avg(voto) desc) posizione,
round(avg(avg(voto)) over(order by avg(voto) desc
rows between 2 preceding and 2 following),2) media5
from esami group by studente;
```

Abbiamo specificato che la finestra deve cominciare due righe prima e finire due righe dopo la riga corrente. Ovviamente le finestre della prima e dell'ultima riga avranno ampiezza tre, mentre le finestre della seconda e della penultima riga avranno ampiezza quattro. Le finestre possono avere ampiezze variabili in funzione dei dati presenti sulla riga corrente; per maggiori dettagli basta consultare il capitolo 19 di **[1]**.

Partizioni e finestre sono i concetti fondamentali con cui familiarizzare per utilizzare correttamente le funzioni analitiche. Nei prossimi paragrafi incontreremo altre funzioni.

Altre funzioni utili

Per calcolare i percentili Oracle mette a disposizione la funzione CUME_DIST che si usa come nell'esempio seguente:

```
select studente, avg(voto) media,
cume_dist() over(order by avg(voto) ) cume_dist,
cume_dist() over(order by avg(voto))*100 percentile
from esami group by studente
order by 4 desc;
```

in pratica la CUME_DIST restituisce un valore compreso tra zero ed uno che rappresenta la percentuale di righe che presentano un valore minore o uguale del valore corrente. Per ottenere il percentile classico, compreso tra zero e cento, abbiamo solo moltiplicato per cento. Nell'esempio i percentili sono multipli di dieci perché gli studenti sono giusto dieci.

La funzione NTILE consente di suddividere i dati in fasce di ampiezza fissa. Ad esempio volendo suddividere i dieci studenti in tre fasce possiamo eseguire:

```
select studente, avg(voto) media,
ntile(3) over(order by avg(voto) desc) terzile
from esami group by studente ;
```

Ovviamente poiché 10 non è multiplo di tre, non è possibile ottenere tre fasce tutte esattamente della stessa ampiezza. In questi casi i record in più vengono aggiunti a partire dalla prima fascia. La massima differenza di ampiezza tra fasce è, quindi, di un record. Nel nostro esempio la prima fascia è composta da quattro record mentre le successive due sono composte da tre record.

Le funzioni FIRST_VALUE e LAST_VALUE consentono di utilizzare, su qualunque riga, i valori della prima e dell'ultima riga di una data finestra. Ad esempio volendo calcolare,

per ogni studente, la differenza tra la sua media e quella del più bravo e la differenza con il meno bravo possiamo fare:

```
select studente, avg(voto) media,
        rank () over(order by avg(voto) desc) posizione,
      avg(voto) - first_value(avg(voto))
        over(order by avg(voto) desc) "Risp. al primo",
      avg(voto) - first_value(avg(voto))
        over(order by avg(voto) asc) "Risp. all'ultimo"
   from esami
group by studente
order by avg(voto) desc;
```

Ovviamente in questo caso abbiamo utilizzato una sola finestra, ampia quanto tutto l'insieme dei dati.

Le ultime funzioni che incontriamo sono LAG e LEAD che servono a referenziare i valori delle righe che si trovano n record prima (con LAG) oppure n record dopo (con LEAD) rispetto alla riga corrente. Ad esempio con la query:

```
select studente, avg(voto) media,
rank () over(order by avg(voto) desc) posizione,
avg(voto) - LAG(avg(voto),2)
   over(order by avg(voto) desc) "Risp. a 2 prima",
avg(voto) - LEAD(avg(voto),2)
   over(order by avg(voto) desc) "Risp. a 2 dopo"
from esami group by studente;
```

calcoliamo con la LAG la differenza tra la media di uno studente e quello che lo precede di due posizioni, e con la LEAD la differenza rispetto allo studente che lo segue di due posizioni.

Esistono ancora altre funzioni analitiche che seguono all'incirca la stessa logica di quelle che abbiamo visto.

Conclusioni

Oracle sta cercando di rendere il proprio database funzionale al massimo per quanti abbiamo bisogno di un data warehouse. In quest'ottica sono state introdotte innovazioni nell'indicizzazione delle tabelle, nei tool di estrazione e caricamento dei dati, nelle funzioni SQL per l'accorpamento dei dati e nelle funzioni SQL per l'analisi dei dati. In quest'articolo ci siamo occupati di queste ultime innovazioni. Abbiamo visto che le funzioni analitiche hanno come principale caratteristica la capacità di utilizzare, per ogni riga ottenuta da una query, i dati presenti sulle altre righe.

Il documento **[1]** in bibliografia è un riferimento eccellente per conoscere la proposta di Oracle per il data warehousing. Nel sito presente in **Riferimenti** possono essere trovate molte utili informazioni.

BIBLIOGRAFIA

[1] Oracle – "*Oracle9i Data Warehousing Guide*", Oracle Corp., 2001

RIFERIMENTI

[2] http://technet.oracle.com