Azzolini Riccardo 2020-03-19

Funzionalità OLAP di SQL:2003

1 On-Line Analytical Processing

Il termine **OLAP** (**On-Line Analytical Processing**) indica le interrogazioni effettuate su un database al fine dell'analisi dei dati, per distinguerle dalle "normali" operazioni di lettura e aggiornamento necessarie per la gestione delle transazioni del dominio applicativo, che sono invece indicate con il termine *OLTP* (*On-Line Transaction Processing*).

Mentre le interrogazioni OLTP riguardano solitamente poche tuple alla volta, quelle OLAP considerano in genere un gran numero di tuple, ad esempio al fine di calcolare dei valori aggregati (somme, medie, conti, ecc.).

Tradizionalmente, le analisi di questo tipo dovevano essere effettuate mediante appositi strumenti, esterni al DBMS, ma gli standard SQL:1999 e SQL:2003 hanno introdotto delle funzionalità OLAP direttamente nelle interrogazioni SQL.

Nota: Le funzionalità OLAP sono riferite al modello relazionale "puro", cioè non sono un'estensione legata al modello object-relational (ma sono piuttosto utilizzate, quindi è utile conoscerle).

2 `GROUP BY`, `HAVING` e operatori di aggregazione

Le funzionalità OLAP di SQL "nascono" dalle clausole `GROUP BY` e `HAVING`, e dagli operatori di aggregazione "classici" (`COUNT`, `SUM`, `AVG`, `MAX` e `MIN`), presenti fin dalla prima versione di SQL. Prima di presentare le estensioni introdotte negli standard successivi, è allora utile riassumere queste funzionalità di base, mediante una query d'esempio.

Data la tabella

```
CREATE TABLE movie_titles (
  title VARCHAR(30),
  year_released DATE,
  movie_type VARCHAR(10),
  dvds_in_stock INTEGER,
  total_dvd_units_sold INTEGER
  -- ...
)
```

calcolare le aggregazioni "classiche" per ciascuna delle categorie di film "horror", "romance" e "action", ma solo se tale categoria comprende almeno 10 titoli.

```
SELECT
  movie_type,
  COUNT(*),
  MAX(total_dvd_units_sold),
  MIN(total_dvd_units_sold),
  SUM(total_dvd_units_sold),
  AVG(total_dvd_units_sold)
FROM movie_titles
GROUP BY movie_type
HAVING movie_type IN ('horror', 'romance', 'action')
  AND COUNT(DISTINCT title) >= 10;
```

L'esecuzione di quest'interrogazione comprende i seguenti passaggi:

1. la relazione viene partizionata secondo il `GROUP BY`;
2. vengono calcolate le funzioni di aggregazione per ogni gruppo;
3. sulle righe così ottenute, viene testato il predicato specificato nella clausola `HAVING`, e solo le tuple che lo soddisfano vengono incluse nel risultato della query.

3 EVERY e ANY/SOME

Con SQL:1999 sono stati aggiunti alcuni nuovi operatori di aggregazione:

- `EVERY`: restituisce `TRUE` se e solo se il valore del suo argomento è `TRUE` per *ogni* tupla della tabella (o di un gruppo, se usato con `GROUP BY`).
- `ANY` (chiamato anche `SOME`): restituisce `TRUE` se e solo se il valore del suo argomento è `TRUE` per *almeno una* tupla della tabella/gruppo.

3.1 Esempio

Data la tabella `movie_titles` dell'esempio precedente, supponendo che essa abbia anche un attributo `store` che specifica il negozio nel quale sono state effettuate le vendite, si vogliono calcolare le aggregazioni classiche per ogni negozio che ha venduto almeno un DVD di ogni film disponibile in tale negozio.

```
SELECT
  store,
  COUNT(*),
  MAX(total_dvd_units_sold),
  MIN(total_dvd_units_sold),
  SUM(total_dvd_units_sold),
  AVG(total_dvd_units_sold)
FROM movie_titles
GROUP BY store
HAVING EVERY(total_dvd_units_sold >= 1);
```

4 ROLLUP e CUBE

Gli operatori `ROLLUP` e `CUBE`, introdotti da SQL:1999, estendono le funzionalità della clausola `GROUP BY`, permettendo il calcolo di aggregazioni su gruppi diversi di righe all'interno di una singola query.

I risultati di questo tipo di aggregazioni sono chiamati *riassunti multidimensionali* (perché gli attributi vengono associati a delle "dimensioni", come in un grafico, e le aggregazioni vengono calcolate "lungo dimensioni diverse").

4.1 Esempi di ROLLUP

Un esempio di interrogazione OLAP non esprimibile con una singola query SQL "classica" è la seguente: "trovare il numero di DVD in magazzino per ogni tipo di film, prima raggruppati per anno di release, e poi in totale su tutti gli anni". L'operatore di `ROLLUP` permette invece di realizzare quest'aggregazione "a due livelli" in una sola query:

```
SELECT movie_type, year_released, SUM(dvds_in_stock) AS sum_of_dvds
FROM movie_titles
GROUP BY ROLLUP(movie_type, year_released);
```

Un esempio di risultato di quest'interrogazione potrebbe essere:

movie_type	year_released	sum_of_dvds
action	1990	250
action	1991	374
action	1992	288
...	...	...
action	NULL	1049
romance	1990	195
romance	1991	439
romance	1992	221
...	...	...
romance	NULL	1194
...	...	...
...	...	...
NULL	NULL	2994

- Le righe senza valori nulli sono le aggregazioni relative ai singoli anni, per ciascun tipo di film (che si sarebbero ottenute anche con un `GROUP BY` "normale").
- Le righe con valori nulli nella colonna `year_released` sono le aggregazioni effettuate su tutti gli anni per un determinato tipo di film.
- L'ultima riga, nella quale sia `movie_type` che `year_released` sono `NULL`, rappresenta il totale calcolato su tutti i tipi di film e tutti gli anni.

Osservazione: L'ordine in cui gli attributi vengono specificati nel `ROLLUP` è importante: in questo esempio, specificando invece `ROLLUP(year_released, movie_type)` sarebbero state calcolate le aggregazioni per ogni anno su tutti i tipi di film, invece che per ogni tipo su tutti gli anni.

Come in una normale interrogazione con `GROUP BY`, dopo il `ROLLUP` può essere aggiunta una clausola `HAVING`, per selezionare i valori aggregati in base a un predicato. Ad esempio:

```
SELECT movie_type, year_released, SUM(dvds_in_stock) AS sum_of_dvds
FROM movie_titles
GROUP BY ROLLUP(movie_type, year_released)
HAVING SUM(dvds_in_stock) < 300;
```

4.2 Esempio di `CUBE`

Partendo da una delle query precedenti,

```
SELECT movie_type, year_released, SUM(dvds_in_stock) AS sum_of_dvds
FROM movie_titles
GROUP BY ROLLUP(movie_type, year_released);
```

se si vuole calcolare l'aggregazione anche per ciascun anno, sommando tutti i tipi di film, è sufficiente sostituire l'operatore `ROLLUP` con `CUBE`:

```
SELECT movie_type, year_released, SUM(dvds_in_stock) AS sum_of_dvds
FROM movie_titles
GROUP BY CUBE(movie_type, year_released);
```

Si ottiene così il seguente risultato:

movie_type	year_released	sum_of_dvds
action	1990	250
action	1991	374
action	1992	288
...	...	...
action	NULL	1049
romance	1990	195
romance	1991	439
romance	1992	221
...	...	...
romance	NULL	1194
...	...	...
...	...	...
NULL	1990	546
NULL	1991	1006
NULL	1992	724
...	...	...
NULL	NULL	2994

Rispetto al risultato di `ROLLUP`, qui si sono aggiunte delle righe corrispondenti ai totali per ciascun anno, contrassegnate da valori nulli nella colonna `movie_type`.

5 Windowing

La funzionalità di **windowing** permette di specificare, per ciascuna tupla del risultato di una query, una selezione di altre tuple, chiamata appunto **window** (finestra), sulla quale effettuare delle aggregazioni.

La caratteristica fondamentale di una window è di essere *mobile*, cioè di variare ("muoversi") in base alla tupla corrente, secondo determinati criteri, come ad esempio:

- includere nella finestra le 4 tuple precedenti e le 4 successive a quella corrente;
- includere nella finestra tutte le tuple nelle quali un certo attributo assume un valore vicino (entro un certo margine) a quello che assume nella tupla corrente.

Non è detto che le tuple della finestra siano contigue (ad esempio, con quest'ultimo criterio potrebbero facilmente non esserlo).

5.1 Esempio: media mobile

Uno degli esempi più tipici di uso del windowing è il calcolo di una "media mobile".

Data una tabella `sales_history`, contenente dati relativi alle vendite di DVD suddivise per negozi e mesi, restituire, per ogni negozio e per ciascun mese dell'ultimo terzo dell'anno 2001 (cioè da settembre a dicembre):

- le vendite di DVD;
- le medie mobili di vendite di DVD, considerando al più due mesi precedenti (sempre limitandosi ai mesi dell'ultimo terzo del 2001).

```
SELECT
  h.store_id,
  h.month,
  h.total_dvd_sales,
  AVG(h.total_dvd_sales) OVER w AS moving_avg
FROM sales_history AS h
WHERE h.month BETWEEN 200109 AND 200112
WINDOW w AS (
  PARTITION BY h.store_id
  ORDER BY h.month
  ROWS 2 PRECEDING
);
```

Un ipotetico risultato di questa interrogazione potrebbe essere:

store_id	month	total_dvd_sales	moving_avg
1	200109	46936.75	46936.75
1	200910	22842.41	34889.58
1	200911	30927.90	33569.02
1	200912	51751.03	35173.78
2	200109	20554.16	20554.16
2	200910	9418.14	14986.15
2	200911	13600.54	14524.28
2	200912	24977.81	15998.83
…	…	…	…

Per effetto del windowing:

- i valori di `moving_avg` nelle tuple di settembre considerano solo le vendite di settembre del negozio corrente;
- quelli relativi a ottobre corrispondono alla media delle vendite di settembre e ottobre;
- quelli relativi a novembre sono la media delle vendite di settembre, ottobre e novembre;
- quelli nelle righe di dicembre sono la media delle vendite di ottobre, novembre e dicembre;

Nota: La sintassi usata per specificare la finestra in questo esempio è chiamata *explicit*. Esiste anche una sintassi *inline*, che permette la specifica di una window direttamente nella clausola `SELECT`, ma essa ha alcune funzionalità in meno.

5.2 Partitioning, ordering e framing

La specifica di una finestra, come quella nell'esempio precedente, è caratterizzata da tre parti fondamentali:

```
WINDOW w AS (
  PARTITION BY h.store_id -- Partitioning
  ORDER BY h.month        -- Ordering
  ROWS 2 PRECEEDING       -- Framing
);
```

Partitioning: raggruppa tutte le tuple che hanno lo stesso valore per gli (uno o più) attributi specificati, analogamente a `GROUP BY`.

Ordering: ordina le tuple all'interno di ciascuna partizione, in funzione di un criterio di ordinamento specificato su uno o più attributi (come la clausola `ORDER BY` usata normalmente nelle interrogazioni).

Framing: data la tupla corrente, stabilisce quali tuple all'interno della partizione (a cui essa appartiene) prendere in considerazione per calcolare l'aggregazione. Ad esempio, `ROWS 2 PRECEEDING` indica che la finestra comprende, oltre alla tupla corrente, anche le due precedenti (purché ci siano almeno due righe precedenti alla corrente nella partizione, altrimenti viene incluso solo il numero di righe effettivamente disponibili).