Azzolini Riccardo 2020-04-30

Integrazione di XML in SQL

1 XMLFOREST

Per riassumere le funzioni predefinite di XML publishing viste finora:

- `XMLELEMENT` crea elementi XML (uno per riga) contenenti i valori di una singola colonna di una tabella (o di una singola espressione);
- `XMLAGG` raggruppa questi elementi in un elemento padre;
- `XMLATTRIBUTES` permette di definire gli attributi di un elemento.

Spesso, però, è necessario creare più elementi per ciascuna riga di una tabella, nei quali inserire, tipicamente, i valori di colonne diverse. Ad esempio, facendo riferimento al database dei film, si potrebbe voler costruire, per ciascun film, un albero XML contenente varie informazioni su di esso:

```
Movie Details
<movie-details> <title>...</title> <year>...</year> ... </movie-details>
<movie-details> <title>...</title> <year>...</year> ... </movie-details>
...
```

Per ottenere questo risultato, esistono sostanzialmente due possibilità:

- Usare più invocazioni di `XMLELEMENT`:

```
SELECT XMLELEMENT(NAME "movie-details",
  XMLELEMENT(NAME "title", title),
  XMLELEMENT(NAME "year", year_released)
  -- altri...
) AS "Movie Details"
FROM movie;
```

- Usare la funzione `XMLFOREST`, che presenta una sintassi simile a `XMLATTRIBUTES`:

```
SELECT XMLELEMENT(NAME "movie-details",
  XMLFOREST(
    title AS "title",
    year_released AS "year"
    -- altri...
  )
```

```
) AS "Movie Details"
FROM movie;
```

Ciascuna di queste soluzioni ha alcuni vantaggi rispetto all'altra:

- Un vantaggio di `XMLFOREST` è che, in presenza di valori nulli, gli elementi corrispondenti vengono semplicemente omessi (sfruttando così i vantaggi del modello dei dati semi-strutturato), mentre con `XMLELEMENT` si creerebbero degli elementi vuoti.
- Invece, è possibile specificare attributi per gli elementi creati (con `XMLATTRIBUTES`) solo se si utilizza `XMLELEMENT`.

2 Estrazione di dati relazionali dagli alberi XML

Le funzioni di XML publishing definite da SQL/XML costituiscono un insieme minimale di strumenti che permettono di passare da dati relazionali a alberi XML.

Viceversa, può essere necessario anche effettuare il contrario: creare una rappresentazione relazionale (cioè sotto forma di tabelle "classiche") dei dati contenuti in degli alberi XML. A tale scopo, gli standard SQL/XML:2003 e 2006 mettono a disposizione molte funzioni predefinite per estrarre dati dagli alberi XML (mediante interrogazioni XQuery / XPath), permettendo di ottenere come risultato delle query una rappresentazione relazionale di tali dati. Le principali sono:

- `XMLQUERY`, usata tipicamente nella clausola `SELECT`;
- `XMLTABLE`, usata nel `FROM`;
- `XMLEXISTS`, usata nel `WHERE`.

2.1 `XMLQUERY`

La funzione predefinita `XMLQUERY` permette di specificare un'interrogazione XQuery / XPath, con la quale vengono estratti dei nodi dagli alberi XML contenuti (solitamente) in una colonna di una tabella. La funzione restituisce poi tali nodi, che possono quindi essere usati, ad esempio, come valori di una colonna della relazione restituita dalla query SQL.

In seguito, sono riportati due esempi semplici (ma significativi) di `XMLQUERY`, riferiti a una tabella `movies_xml` che, per ogni riga, contiene un `id` e un albero XML relativo a un film:

id	movie_xml
001	<movie> <title>...</title> <running-time>...</running-time> ... </movie>
002	<movie> <title>...</title> <running-time>...</running-time> ... </movie>

- Estrarre gli elementi `title` che contengono i titoli dei film.

```
SELECT XMLQUERY(
  'for $m in $col/movie
  return $m/title'
  PASSING movie_xml AS "col"
  RETURNING CONTENT
  NULL ON EMPTY
) AS result
FROM movies_xml;
```

Il risultato ha la forma seguente:

result
<title>...</title>
<title>...</title>
...

In quest'interrogazione, `XMLQUERY` è stata usata per specificare i valori della colonna `result`, valutando la funzione per ciascuna tupla di `movies_xml`:

  - l'interrogazione XQuery (che, in questo caso, usa solo funzionalità già presenti in XPath 2.0) viene scritta sotto forma di stringa;
  - `PASSING movie_xml AS "col"` lega alla variabile `$col` il valore della colonna `movie_xml` nella tupla corrente, rendendolo disponibile all'interno dell'interrogazione XQuery;
  - `RETURNING CONTENT` indica che il valore restituito è un albero XML con un singolo nodo radice[1] (e non una sequenza di alberi);

[1] Questo nodo radice è di tipo "documento": può avere come figli un numero qualsiasi di elementi (e/o attributi, dati testuali, ecc.), ma non rappresenta esso stesso un elemento. Quindi, anche se si restituisce un singolo albero, esso può comunque contenere, di fatto, una "sequenza" di elementi.

  - `NULL ON EMPTY` specifica che verrà restituito un valore nullo se l'interrogazione XQuery non seleziona alcun nodo.

- Calcolare la durata media dei film.

```
SELECT AVG(
  XMLCAST(XMLQUERY(
    'for $m in $col/movie
    return $m/running-time/text()'
    PASSING movie_xml AS "col"
    RETURNING CONTENT
    NULL ON EMPTY
  ) AS DECIMAL(8, 1))
) AS avg_running_time
FROM movies_xml;
```

Qui i valori estratti da `XMLQUERY` per tutte le righe della tabella `movies_xml` vengono passati alla funzione di aggregazione `AVG`. È però necessario l'utilizzo della funzione predefinita `XMLCAST`, per convertire i valori XML restituiti da `XMLQUERY` (in questo caso, essi sono di fatto delle stringhe) in un tipo di dato numerico (qui si è scelto `DECIMAL(8, 1)`, che rappresenta numeri decimali di massimo 8 cifre, di cui 1 dopo la virgola), sul quale sia possibile calcolare la media.

Un esempio di risultato è:

avg_running_time
103.5