Azzolini Riccardo 2020-04-28

Integrazione di XML in SQL

1 XML nei database

In origine, si pensava che XML sarebbe stato adottato come modello dei dati per i database, mediante l'introduzione di DBMS nativi per dati semi-strutturati.

In realtà, è andata a finire come per i modelli object-oriented: i DBMS XML nativi non hanno avuto successo, ma piuttosto SQL è stato esteso con alcune funzionalità che supportano la gestione/interrogazione di dati semi-strutturati all'interno delle tabelle. Lo standard che definisce queste estensioni è **SQL/XML**, che ha diverse versioni:

- SQL/XML:2003 ha aggiunto un tipo di dato XML (basato sul modello dei dati Infoset) ai tipi predefiniti di SQL, e la possibilità di costruire e memorizzare valori di tale tipo;
- con SQL/XML:2006, il supporto è stato esteso al modello dei dati utilizzato da XQuery 1.0 (e quindi XPath 2.0), ed è stata aggiunta la possibilità di eseguire interrogazioni XQuery.

Attualmente, i principali DBMS commerciali supportano (come sempre, con differenze sintattiche e semantiche anche significative) la possibilità di:

- includere alberi XML come valori nelle tabelle;
- scrivere interrogazioni SQL che contengono, come sotto-interrogazioni, delle espressioni XQuery.

2 Interazione tra XML e tabelle prima di SQL/XML

Senza lo standard SQL/XML, per rappresentare un albero XML all'interno di un database relazionale è tipicamente necessario "scomporre" l'albero in un insieme di tabelle:

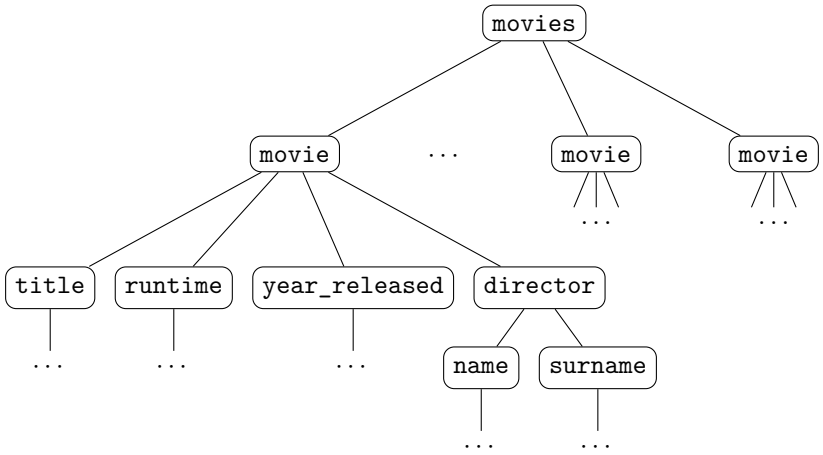
- gli elementi con lo stesso nome vengono messi in un'unica tabella, che può contenere valori nulli in corrispondenza di eventuali nodi mancanti nell'albero (dato che XML è appunto un modello semi-strutturato);
- il legame tra un elemento e i suoi sottoelementi viene rappresentato con un vincolo di chiave esterna.

Questa trasformazione prende il nome di **XML shredding**.

Invece, la trasformazione inversa, cioè la rappresentazione in XML di dati relazionali, era un'operazione molto meno frequente prima dell'introduzione di SQL/XML.

2.1 Esempio

Per illustrare lo shredding, si considera un albero XML contenente dati relativi a dei film:



Un modo per rappresentare quest'albero con un insieme di tabelle è il seguente:

Tabella `movies`

movie_id
...
...
...

Tabella `movie`

movie_id	title	runtime	year_released	director_id
...	...	...	...	...
...	...	NULL	...	...
...	...	...	...	...

Tabella `director`

director_id	name	surname
...	...	...
...	...	...

Nota: Questo esempio è un caso particolare di albero XML molto ben strutturato (nonostante ammetta comunque una certa flessibilità, poiché certi elementi possono essere omessi), quindi la traduzione al modello relazionale risulta abbastanza "naturale". Spesso, invece, una strutturazione del genere non è presente, e allora l'operazione di shredding risulta poco efficace.

3 Funzionalità di SQL/XML

Lo standard SQL/XML è stato definito con i seguenti obiettivi:

- **XML publishing**: interrogare dati relazionali e produrre come risultato elementi/alberi XML;
- **shredding trasparente**: immagazzinare elementi/alberi XML in un database relazionale;
- fornire un mapping tra i tipi di dati object-relational e quelli di XQuery;
- interrogare alberi XML, con la possibilità di immagazzinare i risultati in tabelle relazionali.

Di questi, i primi due obiettivi sono stati raggiunti dalla prima versione, SQL/XML:2003, mentre le restanti funzionalità sono definite da SQL/XML:2006.

4 XML publishing

Per illustrare le principali funzionalità di XML publishing fornite da SQL/XML, si presentano in seguito alcuni esempi, riferiti al database relazionale dei film mostrato sopra.

- Estrarre il titolo di ciascun film e inserirlo in un elemento XML `title`.

  ```
  SELECT XMLELEMENT(NAME "title", title) AS "Movie Titles"
  FROM movie;
  ```

  Il risultato è una tabella con un singolo attributo (qui si è scelto di chiamarlo `Movie Titles`), che, per ogni riga, contiene un elemento XML `title`:

Movie Titles
<title>...</title>
<title>...</title>
...

L'elemento principale di quest'interrogazione è la funzione predefinita `XMLELEMENT`: per ogni tupla del risultato dell'interrogazione, essa costruisce un elemento XML con il nome e il contenuto specificati. In generale, il contenuto dell'elemento è determinato da un'espressione SQL; in questo caso, tale espressione è costituita semplicemente dal valore dell'attributo `title`, ma si potrebbero usare anche espressioni più complesse, come ad esempio una sotto-interrogazione.

- Estrarre, come prima, il titolo di ciascun film, inserendolo in un elemento XML `title` che, in più, abbia anche due attributi: la durata del film e il cognome del regista.

```
SELECT XMLELEMENT(NAME "title",
  XMLATTRIBUTES(
    m.runtime AS "runtime",
    (
      SELECT d.surname
      FROM director d
      WHERE d.director_id = m.director_id
    ) AS "director"
  ),
  'This movie title is ' || m.title
) AS "Movie Titles"
FROM movie m;
```

Il risultato è ancora una tabella con una singola colonna e un elemento per riga:

```
Movie Titles
<title runtime="..." director="...">This movie title is ...</title>
<title runtime="..." director="...">This movie title is ...</title>
...
```

Come mostrato in questo esempio, gli attributi di un elemento vengono definiti con la funzione `XMLATTRIBUTES`, con una stessa sintassi analoga a quella usata per definire gli attributi del risultato di una "normale" interrogazione SQL. Questa funzione è ammessa solo all'interno di `XMLELEMENT`.

- Estrarre i titoli dei film, inserire ciascuno in un elemento `title`, e raccogliere tutti questi elementi all'interno di un elemento `all-titles`, formando un unico albero XML (invece di uno separato per ogni film, come prima).

```
SELECT XMLELEMENT(NAME "all-titles",
  XMLAGG(
    XMLELEMENT(NAME "title", title)
    ORDER BY year_released
  )
) AS "Movie Titles"
FROM movie;
```

La relazione risultante ha ancora una singola colonna, ma, questa volta, anche una sola riga, che contiene il singolo albero XML costruito dalla query:

```
Movie Titles
-------------------
<all-titles>
  <title>...</title>
  <title>...</title>
  ...
</all-titles>
```

`XMLAGG` è una funzione di aggregazione (analoga alle classiche `COUNT`, `SUM`, `MAX`, ecc.) che raccoglie gli elementi prodotti per tutte le righe della tabella, formando una sequenza che può essere passata come argomento a `XMLELEMENT`, creando così un singolo elemento con una sequenza di sottoelementi.

Se, invece, non si usasse `XMLAGG`,

```
SELECT XMLELEMENT(NAME "all-titles",
  XMLELEMENT(NAME "title", title)
) AS "Movie Titles"
FROM movie;
```

si otterrebbe un albero XML separato per ogni film (che qui non è il risultato voluto):

```
Movie Titles
-----------------------------------------
<all-titles><title>...</title></all-titles>
<all-titles><title>...</title></all-titles>
...
```

Questo perché, come già detto, di per sé la funzione `XMLELEMENT` viene valutata – e quindi costruisce un elemento – per ogni tupla selezionata dall'interrogazione.

Nota: Gli elementi/alberi XML contenuti nelle tabelle risultanti da queste interrogazioni sono valori del tipo di dato XML definito da SQL/XML.