Azzolini Riccardo 2020-03-10

Object-relational data model – Collection type

1 Principali collection type

Una delle funzionalità più utilizzate del modello object-relational (al contrario dell'ereditarietà) sono i **collection type**. Oltre ai row type, già presentati, alcuni dei principali sono:

- array: una lista ordinata¹ di valori che ha una lunghezza massima prefissata, ma può anche essere più corta;
- multiset (multi-insieme): una lista non ordinata di valori, che ammette duplicati e non ha una dimensione massima.

Questi due tipi sono stati introdotti dagli standard SQL:1999 e SQL:2003, rispettivamente. Come al solito, però, i vari DBMS li implementano con molte varianti sintattiche.

2 Uso nella definizione e nell'inserimento dei dati

Nell'esempio seguente si usano array e multiset (oltre ai row type) per rappresentare i dati relativi a un libro:

```
CREATE TYPE Publisher AS (
name VARCHAR(20),
branch VARCHAR(20)
);

CREATE TYPE Book AS (
title VARCHAR(20),
author_array VARCHAR(20) ARRAY[10],
publication_date DATE,
publisher Publisher,
keyword_set VARCHAR(20) MULTISET
);
```

¹Un array è ordinato nel senso che ciascun valore si trova in una posizione numerata (analogamente agli array presenti in molti linguaggi di programmazione), e non nel senso che l'ordine dei valori sia crescente, decrescente, ecc.

CREATE TABLE Books OF Book;

- Un valore dell'attributo author_array è un array contenente da 0 a 10 elementi, ciascuno di tipo VARCHAR(20).
- keyword_set contiene un multi-insieme di elementi VARCHAR(20).
- I nomi di questi due attributi illustrano la convenzione, spesso adottata, di contrassegnare gli attributi di tipi complessi mediante opportuni suffissi (qui _array e _set).

Per inserire nella tabella Books, così definita, un'istanza del tipo Book, si usa un normale comando INSERT, nel quale servono però apposite sintassi per la specifica dei valori di tipi complessi:

```
INSERT INTO Books VALUES (
   'Compilers',
   ARRAY['Smith', 'Jones'],
   '2020-03-12',
   NEW Publisher('McGraw-Hill', 'NY'),
   MULTISET['parsing', 'analysis']
);
```

Nota: A seconda del DBMS, l'invocazione del costruttore di un row type (come, in questo esempio, Publisher(...)) può richiedere o meno la parola chiave NEW.

3 Interrogazioni: UNNEST

Per poter interrogare una tabella con attributi di tipo array e/o multiset, è necessario un costrutto che permetta di accedere ai valori contenuti all'interno di tali collezioni. A tale scopo, è disponibile UNNEST (letteralmente, "de-nidificazione"), che "appiattisce" un array o un multiset, estraendone i valori.

Di fatto, UNNEST crea una tabella temporanea contenente i valori della collezione. Infatti, all'interno delle interrogazioni, esso può essere usato in tutti i posti in cui può apparire una sotto-interrogazione.

3.1 Esempi

Data la tabella Books definita sopra,

1. ottenere i titoli dei libri che hanno "database" tra le keyword:

```
SELECT title
FROM Books
WHERE 'database' IN (UNNEST(keyword_set));
```

2. ottenere il titolo di ciascun libro e i nomi dei rispettivi autori, sotto forma di tuple del tipo (titolo, autore):

```
SELECT b.title, a.author
FROM Books AS b, UNNEST(b.author_array) AS a(author);
```

Esempio di risultati di questa query

title	author
Compilers	Smith
Compilers	Jones
Algorithms	Sedgewick

3. creare una versione "piatta" (senza attributi complessi) di Books:

```
SELECT
  b.title,
  a.author,
  b.publication_date,
  b.publisher.name, b.publisher.branch,
  k.keyword
FROM
  Books AS b,
  UNNEST(b.author_array) AS a(author),
  UNNEST(b.keyword_set) AS k(keyword);
```

Versione "piatta" di Books restituita dalla query

title	author	<pre>publisher.name</pre>	<pre>publisher.branch</pre>	keyword
Compilers	Smith	McGraw-Hill	NY	parsing
Compilers	Smith	McGraw-Hill	NY	analysis
Compilers	Jones	McGraw-Hill	NY	parsing
Compilers	Jones	McGraw-Hill	NY	analysis
	•••	•••		

Osservazione: Interrogazioni simili agli esempi 2 e 3 sono tipiche degli ambiti in cui è necessario esportare i dati verso un altro DBMS, che potrebbe non supportare lo stesso formato dei dati. Allora:

- 1. per prima cosa, si usa UNNEST per trasformare i dati in un formato "piatto", ampiamente supportato (ma, solitamente, caratterizzato da una notevole ridondanza dei dati, che lo rende non adatto a essere utilizzato direttamente come rappresentazione "principale");
- 2. si trasferiscono i dati appiattiti al DBMS di destinazione;
- 3. infine, nel DBMS di destinazione, si ricrea un'organizzazione complessa per i dati importati, in base ai formati che tale DBMS supporta.

3.2 Posizioni degli elementi di un array

Facendo l'UNNEST di un array, si perde l'informazione relativa alla posizione degli elementi. Se, invece, la si vuole mantenere, bisogna aggiungere la clausola WITH ORDINALITY, con la quale UNNEST genera una tabella temporanea di *due colonne*: la prima contiene ancora i valori degli array, mentre la seconda contiene gli indici corrispondenti.

Ad esempio, con l'interrogazione

```
SELECT b.title, a.author, a.position
FROM
Books AS b,
UNNEST(b.author_array) WITH ORDINALITY AS a(author, position);
si ottiene un risultato del tipo:
```

title	author	position
Compilers	Smith	1
Compilers	Jones	2
Algorithms	Sedgewick	1

4 Accesso agli array

Per accedere ai singoli elementi di un array, non è necessario l'UNNEST: si può utilizzare semplicemente una sintassi analoga a quella tipicamente presente nei linguaggi di programmazione.

Ad esempio, per ottenere gli autori del libro intitolato "Compilers", sapendo che esso ha due autori, si può usare la query:

```
SELECT author_array[1], author_array[2]
FROM Books
WHERE title = 'Compilers';
```

²Un'interrogazione che appiattisce i dati viene spesso salvata come vista.

Osservazione: Gli indici partono da 1, e non da 0 (almeno secondo lo standard, poiché questo è uno degli aspetti che potrebbero facilmente variare in base al DBMS).