

# Operatore COLLECT

## 1 Introduzione

L'operatore UNNEST, presentato nella lezione precedente, permette di “appiattare” una relazione contenente attributi di tipi complessi. L'operazione contraria (cioè di “nidificazione”, “nesting”) è realizzata dall'operatore COLLECT.

Prima di introdurre questo operatore, è utile ripassare il funzionamento della clausola GROUP BY, dato che COLLECT si basa su di essa.

## 2 Riassunto di GROUP BY

Si considerino, come esempio, una tabella employees,

id	branch	salary
1	NY	1000
2	Washington	1200
3	Washington	900
4	NY	2000
5	NY	1000
...	...	...

e la seguente interrogazione su di essa:

```
SELECT branch, AVG(salary)
FROM employees
GROUP BY branch;
```

L'esecuzione della clausola GROUP BY si articola in due passaggi:

1. La relazione viene partizionata in gruppi distinti, in funzione degli attributi specificati nel GROUP BY:

id	branch	salary
1	NY	1000
4	NY	2000
5	NY	1000
2	Washington	1200
3	Washington	900
...	...	...

2. Per ognuno di questi gruppi, viene calcolato un valore aggregato (in questo caso la media):

branch	AVG(salary)
NY	1333.33
Washington	1050.00
...	...

### 3 COLLECT

L'operatore `COLLECT`, che viene usato insieme al `GROUP BY`, raccoglie in un multi-insieme i valori che un attributo assume all'interno delle tuple di ciascun gruppo.

Ad esempio, data una tabella "piatta" `book_authors`, contenente gli autori di ciascun libro,

title	author
Compilers	Smith
Compilers	Jones
Algorithms	Sedgewick
...	...

la seguente interrogazione ricava una tabella che contiene una sola tupla per ogni libro, "impacchettando" tutti gli autori di tale libro in un multiset:

```
SELECT title, COLLECT(author) AS author_set
FROM book_authors
GROUP BY title;
```

title	author
Compilers	{'Smith', 'Jones'}
Algorithms	{'Sedgewick'}
...	...

In generale, quindi, `COLLECT` permette di passare da relazioni piatte a relazioni aventi valori su tipi complessi,<sup>1</sup> ovvero effettua la trasformazione inversa rispetto all'`UNNEST`.

### 3.1 Esempio: Ricostruzione di `books` da `flat_books`

La tabella riportata in seguito, chiamata ad esempio `flat_books`, è una versione “appiattita” (ottenuta mediante `UNNEST`) di una tabella originale `books`: nella quale i dati relativi ad autori, editore e parole chiave erano rappresentato attraverso dei tipi complessi.

<code>title</code>	<code>author</code>	<code>publisher_name</code>	<code>publisher_branch</code>	<code>keyword</code>
Compilers	Smith	McGraw-Hill	NY	parsing
Compilers	Smith	McGraw-Hill	NY	analysis
Compilers	Jones	McGraw-Hill	NY	parsing
Compilers	Jones	McGraw-Hill	NY	analysis
...	...	...	...	...

Per riottenere la tabella originale, ovvero

<code>title</code>	<code>author_set</code>	<code>publisher</code>	<code>keyword_set</code>
Compilers	{'Smith', 'Jones'}	('McGraw-Hill', 'NY')	{'parsing', 'analysis'}
...	...	...	...

dove i valori di `publisher` sono istanze del tipo

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

è necessaria la seguente interrogazione:

```
SELECT
  title,
  COLLECT(author) AS author_set,
  Publisher(publisher_name, publisher_branch) AS publisher,
  COLLECT(keyword) AS keyword_set
FROM flat_books
GROUP BY title, publisher_name, publisher_branch;
```

<sup>1</sup>Per la precisione, `COLLECT` permette (come già detto) la costruzione di valori complessi di tipo multiset. Invece, per la creazione delle istanze dei row type, si usano i rispettivi costruttori. Nel passaggio da una relazione piatta a una più strutturata, vengono spesso usati entrambi questi strumenti.

## 4 Supporto di UNNEST e COLLECT

Il supporto di UNNEST e COLLECT è estremamente variabile tra diversi DBMS. In particolare, tali operatori possono:

- non essere supportati pienamente;
- avere una sintassi diversa;
- anche a parità di sintassi, avere effetti diversi.