Join esterni, self-join e operazioni insiemistiche

Il **self-join** si attua quando si congiunge una tabella con se stessa. Questa esigenza si presenta nel caso delle associazioni ricorsive, per esempio nel caso di una gerarchia aziendale o della distinta base che descrive oggetti composti da diverse parti.

ESEMPIO

Mostriamo come si possa realizzare un **self-join** su una tabella. Si consideri l'associazione esistente tra l'entità *Oggetto* di attributi: *ID*, *Descrizione* e *Qta*, e le sue parti (che sono altri oggetti). La tabella che rappresenta gli oggetti e tale associazione ha lo schema sotto riportato assieme alla figura della tabella *Oggetti* con alcune righe di esempio.

Oggetti (ID, Descrizione, Qta, *ComponenteDi*)

Il bottone (*ID=4*) è una componente della camicia (*ID=1*) e in una camicia ci sono 10 bottoni.

Oggetti

ID	Descrizione	Qta	ComponenteDi
1	Camicia	1	
2	Manica	2	1
3	Colletto	1	1
4	Bottone	10	1
5	Taschino	1	1

Record: 4 di 20 Nessun filtro Cerca

L'oggetto camicia è composto da:
1 colletto, 2 maniche, 10 bottoni, 1 taschino e altri oggetti.

Se si vogliono ottenere le informazioni delle diverse parti che compongono un oggetto, per esempio la camicia, bisogna congiungere la tabella *Oggetti* con se stessa, cioè operare un *self-join*. Per esprimere la condizione che una riga di *Oggetti* deve essere abbinata a una riga, sempre di *Oggetti*, con il valore di *ID* uguale a quello di *ComponenteDi* nell'altra, occorre usare gli **alias** per il nome della tabella.

Oggetti **AS Parti**

ID	Descrizione	Qta	ComponenteDi
1	Camicia	1	
2	Manica	2	1
3	Colletto	1	1
4	Bottone	10	1
5	Taschino	1	1

Record: 4 di 20 Nessun filtro Cerca

Oggetti **AS Composto**

ID	Descrizione	Qta	ComponenteDi
1	Camicia	1	
2	Manica	2	1
3	Colletto	1	1

La ridenominazione di *Oggetti* precisa il differente ruolo giocato dalla tabella *Oggetti* nell'associazione ricorsiva.

L'interrogazione SQL che elenca le parti che compongono la camicia è mostrata in figura assieme alla tabella visualizzata al momento dell'esecuzione della query.

```
SELECT Parti.Descrizione AS Componente,
       Parti.Qta,
       Composto.Descrizione AS Prodotto
FROM Oggetti AS Parti, Oggetti AS Composto
WHERE Parti.ComponenteDi = Composto.ID AND
      Composto.Descrizione = 'Camicia';
```

SelfJoin

Componente	Qta	Prodotto
Manica	2	Camicia
Colletto	1	Camicia
Bottone	10	Camicia
Taschino	1	Camicia

Record: 3 di 4 Nessun filtro Cerca

I **join esterni** includono nella congiunzione anche quelle righe di una delle due tabelle che non hanno corrispondenti nell'altra. Si distinguono in *left join*, *right join*, *full join*.

ESEMPIO

Un'azienda è articolata in un certo numero di dipartimenti. I dipendenti sono assegnati ai diversi dipartimenti che hanno a capo un manager responsabile della gestione.

Le entità individuate sono:

- *Impiegato* con le informazioni sui dipendenti: un codice identificativo (chiave primaria), nome, cognome, residenza e stipendio. Stipendio e codice identificativo sono attributi di tipo numerico, gli altri sono di tipo carattere.
- *Dipartimento* con le informazioni sui dipartimenti: un codice che individua il dipartimento (la chiave primaria), il nome del dipartimento e il suo indirizzo. Tutti gli attributi sono di tipo carattere.

Il database è composto da due tabelle come nello schema seguente (gli attributi sottolineati rappresentano le chiavi primarie, mentre le chiavi esterne sono indicate in corsivo e colore).

Dipartimento contiene il codice del dipartimento di appartenenza di un dipendente.

Manager contiene il valore della chiave primaria del dipendente che dirige il dipartimento.

Impiegati (ID, Nome, Cognome, Residenza, Stipendio, *Dipartimento*)
Dipartimenti (Codice, Descrizione, Sede, *Manager*)

Nella seguente figura sono mostrate le tabelle *Impiegati* e *Dipartimenti* con alcuni dati di esempio.

L'impiegato con *ID* = 6 lavora nel dipartimento *Magazzino*, diretto da *Margherita Colombi*.

L'impiegato con *ID* = *5* non è assegnato ad alcun dipartimento. Si tratta di un dipendente appena assunto e non ancora assegnato ad alcun dipartimento.

Impiegati

ID	Nome	Cognome	Residenza	Stipendio	Dipartimento
1	Mario	Rossi	Torino	32000	Prod
5	Marco	Viola	Palermo	28300	
6	Enrico	Mori	Torino	25000	Mag
10	Margherita	Colombi	Roma	65000	Prod
11	Fabrizio	Magenta	Torino	41000	Prod
12	Franco	Volpi	Bari	61000	Amm
13	Ugo	Boss	Cagliari	85000	Direz
14	Mario	Gatti	Firenze	57000	R&S
16	Elisabetta	Gregis	Roma	29000	Amm
17	Laura	Moretti	Venezia	52600	Mkt
18	Erica	Bruni	Firenze	61500	Mag
19	Anita	Bianco	Perugia	39000	Direz

Record: 3 di 12 | Nessun filtro | Cerca

Dipartimenti

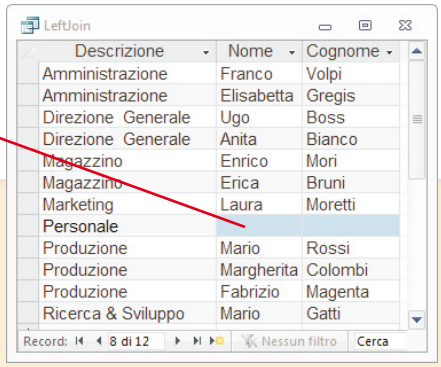
Codice	Descrizione	Sede	Manager
Amm	Amministrazione	Roma	12
Direz	Direzione Generale	Roma	13
Mag	Magazzino	Torino	10
Mkt	Marketing	Milano	13
Prod	Produzione	Torino	10
R&S	Ricerca & Sviluppo	Torino	14
Pers	Personale	Roma	12

Record: 3 di 7 | Nessun filtro | Cerca

Margherita Colombi (*ID*=10) è a capo sia del *Magazzino* che della *Produzione*.

Tra i join esterni consideriamo il **left join** e mostriamo la sua utilità nella soluzione di un tipico problema di assenza: elencare i dipartimenti che non hanno dipendenti loro assegnati. Per farlo congiungiamo la tabella *Dipartimenti* e la tabella *Impiegati* con un *left join*.

Il **left join** include comunque tutte le righe della tabella a sinistra (*Dipartimenti*) anche in mancanza di record associati. La riga del dipartimento *Personale* è stata completata con *valori nulli* nei campi *Nome* e *Cognome* perché non ci sono dipendenti ad esso assegnati.

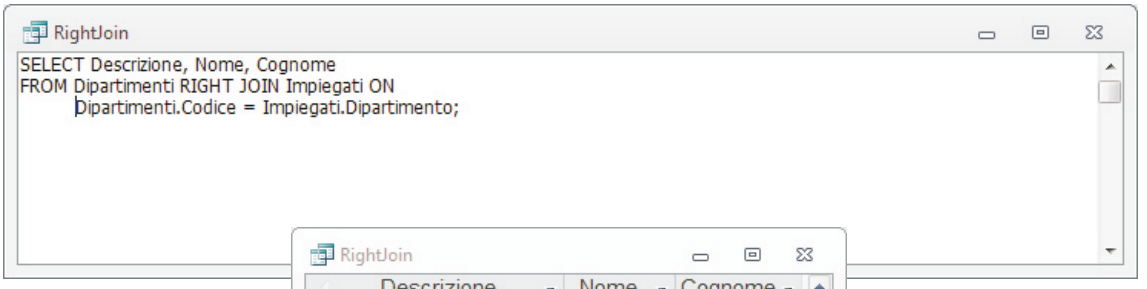


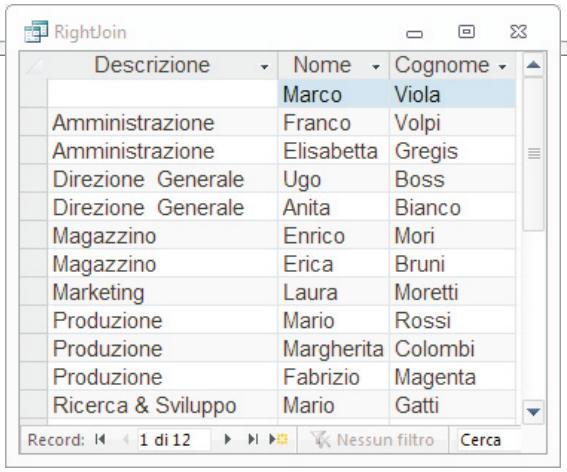

```
SELECT Descrizione, Nome, Cognome
FROM Dipartimenti D LEFT JOIN Impiegati I ON
   D.Codice = I.Dipartimento;

SELECT Descrizione
FROM Dipartimenti D LEFT JOIN Impiegati I ON
   D.Codice = I.Dipartimento
WHERE Cognome IS NULL;
```

I dipartimenti senza impiegati si riconoscono per la presenza di valori nulli nei campi che provengono da *Impiegati*. Per il controllo si usa il predicato **IS NULL**.

Mostriamo come usare il *right join* per risolvere il problema di identificare i dipendenti che sono associati a dipartimenti inesistenti. Va osservato che una tale situazione non si può presentare se è stata implementata l'*integrità referenziale*. Questo controllo è comunque necessario quando si vuole verificare la correttezza dei valori immessi se l'integrità referenziale non è già garantita dal DBMS.

Si congiungono le tabelle *Dipartimenti* e *Impiegati* con un *right join*. Questo implica che nella congiunzione siano incluse tutte le righe con gli attributi di *Impiegati*; i campi di *Dipartimenti* che provengono dalle righe che non hanno corrispondenti in *Impiegati* sono riempiti con *valori nulli*.

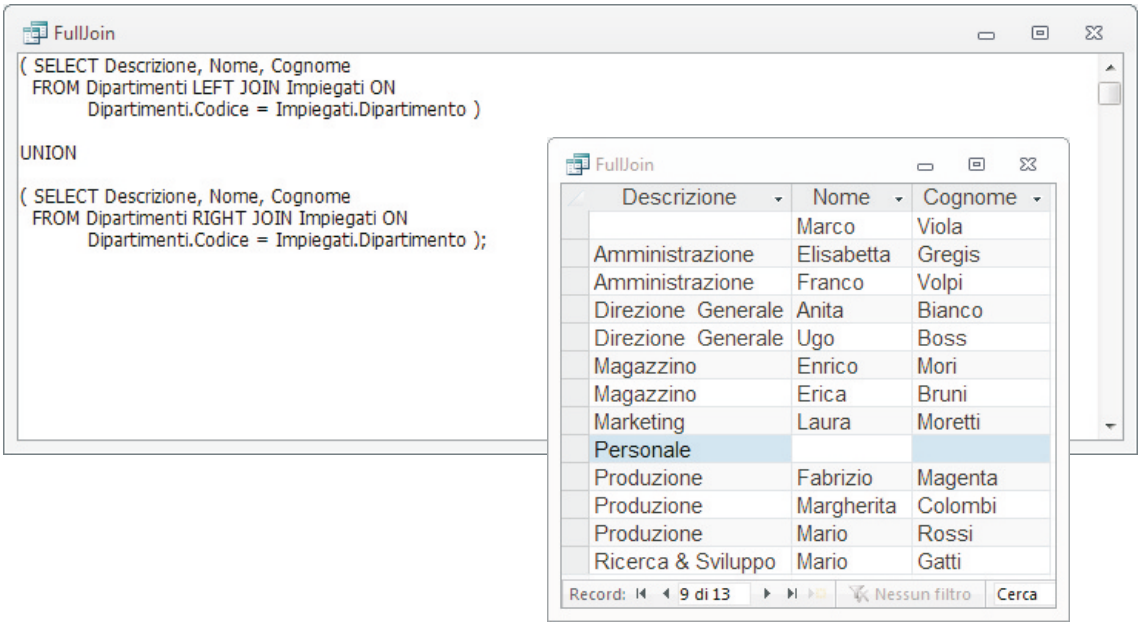




La figura precedente mostra il codice SQL con il *right join* sopra descritto e la tabella prodotta dalla sua esecuzione. Nella prima riga del *right join* compare *Marco Viola* che non presenta alcun valore nel campo *Descrizione*. Si tratta del caso di un dipendente che non è stato assegnato ad alcun dipartimento. Per riconoscere gli impiegati senza dipartimento o con un codice dipartimento non corretto è sufficiente cercare nel *right join* i valori nulli nel campo *Descrizione* con il seguente codice:

```
SELECT Nome, Cognome
FROM Dipartimenti RIGHT JOIN Impiegati ON Dipartimenti.Codice =
     Impiegati.Dipartimento
WHERE Descrizione IS NULL;
```

Lo standard SQL prevede anche l'operazione di *full join* che però non è presente nella versione di Access.
Il *full join* è realizzabile con SQL in Access attraverso l'operazione insiemistica di **unione**. Infatti il *full join* tra due tabelle deve comprendere sia le righe che compaiono nel *left join*, sia quelle del *right join*.
In sostanza, per includere nella congiunzione tutte le righe di *Impiegati* e di *Dipartimenti*, bisogna scrivere il comando in figura che evidenzia anche la sintassi da seguire per costruire l'operazione insiemistica di **unione** (**Union**) tra tabelle:



Oltre all'unione di tabelle (con colonne compatibili), lo standard SQL prevede anche le operazioni di **intersezione** e **differenza**. Date due tabelle **T1** e **T2**, con uguale numero di colonne e con colonne ordinatamente del medesimo tipo, le due operazioni di intersezione e differenza tra *T1* e *T2* sono indicate in SQL con i comandi **Intersect** ed **Except**:

```
T1 INTERSECT T2;          Intersezione
T1 EXCEPT T2;             Differenza
```

Nella versione SQL di Access le operazioni di intersezione e differenza non sono ammesse.