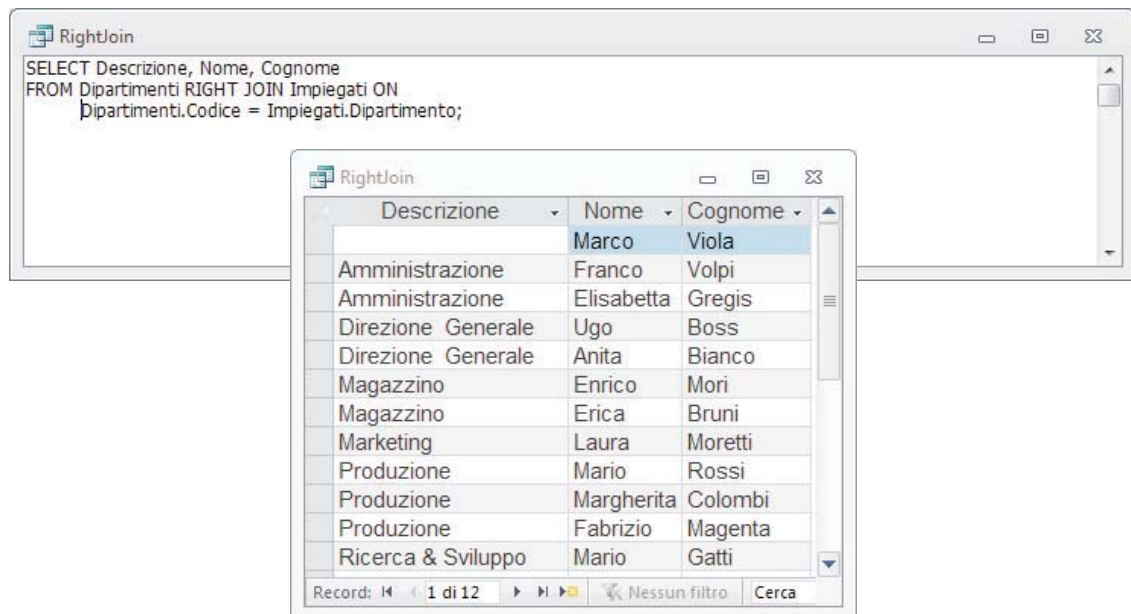


Altri tipi di join esterni e operazioni insiemistiche

Ci sono tre join esterni: *left join*, *right join* e *full join*. 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.



The screenshot shows a SQL query window titled 'RightJoin' with the following query:

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

Below the query, a table window displays the results of the query. The table has three columns: 'Descrizione', 'Nome', and 'Cognome'. The first row shows a null value for 'Descrizione' and the name 'Marco Viola'. The subsequent rows show various departments and their corresponding employees.

Descrizione	Nome	Cognome
	Marco	Viola
Amministrazione	Franco	Volpi
Amministrazione	Elisabetta	Gregis
Direzione Generale	Ugo	Boss
Direzione Generale	Anita	Bianco
Magazzino	Enrico	Mori
Magazzino	Erica	Bruni
Marketing	Laura	Moretti
Produzione	Mario	Rossi
Produzione	Margherita	Colombi
Produzione	Fabrizio	Magenta
Ricerca & Sviluppo	Mario	Gatti

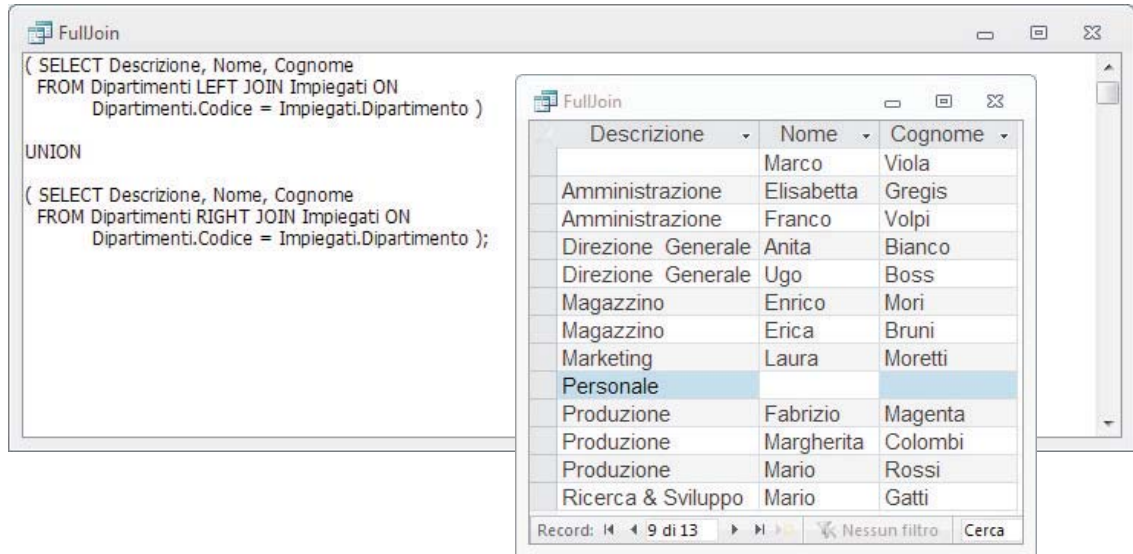
La figura mostra il codice SQL con il *right join* sopra descritto e la tabella prodotta dalla sua esecuzione. Nella prima riga del *right join* compare *Marco Viola* che non presenta alcun valore nel campo *Descrizione*. Si tratta, com'è noto, del caso di un dipendente che non è stato assegnato ad alcun 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:

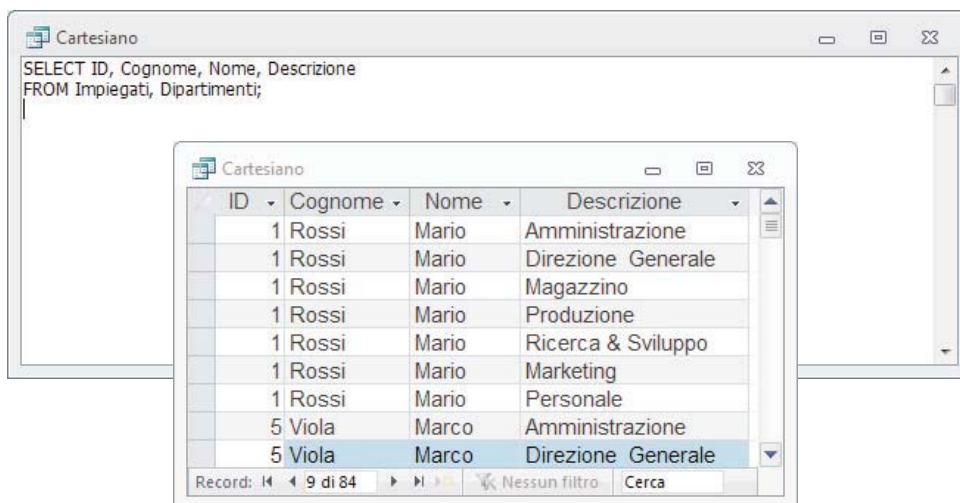


Oltre all'unione di tabelle (con colonne compatibili), lo standard SQL prevede anche le operazioni di **intersezione** e **differenza**. Date due tabelle **T1** e **T2**, con uguale numero di colonne e con colonne ordinatamente del medesimo tipo, le due operazioni di intersezione e differenza tra **T1** 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. Sotto certe condizioni, le due operazioni sono realizzabili con interrogazioni nidificate, come mostrato nel materiale on line "3. Intersezione e differenza con il predicato IN".

Si consideri infine l'interrogazione in figura:



La query, eseguita con le tabelle dell'esempio produce una tabella con 84 righe ottenute combinando le 12 righe di *Impiegati* con le 7 di *Dipartimenti* in tutti i modi possibili. In altre parole la tabella prodotta non è altro che il **prodotto cartesiano** di *Impiegati* per *Dipartimenti*. Come si è già osservato il prodotto cartesiano di tabelle non è di alcuna utilità. È utile, invece, per interpretare il comportamento dell'istruzione *Select*:

```
SELECT ElencoColonne  
FROM Tabella1, Tabella2  
WHERE Condizioni
```

Le righe di *Tabella1* e *Tabella2* sono combinate in tutti i modi possibili e tra le righe così prodotte vengono scelte quelle che soddisfano le condizioni espresse nella clausola *Where*. Il risultato viene proiettato sulle colonne della clausola *Select*.

Tutto questo da un punto di vista concettuale. In pratica le operazioni sono eseguite in modo differente: per ottimizzare l'esecuzione dell'interrogazione il DBMS anticipa, nel limite del possibile, le selezioni per limitare il numero di possibili combinazioni tra le righe delle due tabelle.