



# **Linguaggio SQL prima parte**

**A. Lorenzi, E. Cavalli**

**INFORMATICA PER ISTITUTI TECNICI TECNOLOGICI**

---

**Copyright © Istituto Italiano Edizioni Atlas**

---

# Linguaggio SQL

# Il linguaggio SQL

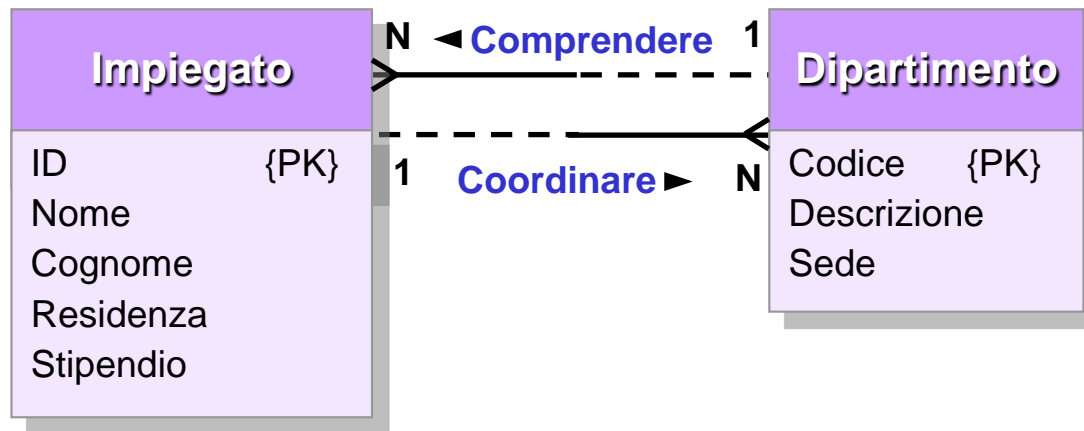
Un **DBMS** deve disporre di un linguaggio per:

- definire e creare il database: **DDL** - Data Definition Language
- inserire, cancellare e modificare i dati: **DML** - Data Manipulation Language
- interrogare il database per estrarre informazioni: **QL** - Query Language

*Il linguaggio deve permettere di fare tutto questo facilmente ed essere basato su costrutti semplici e facili da imparare. Le sue caratteristiche, infine, devono essere standardizzate in modo che un utente, cambiando DBMS, non debba apprendere un nuovo linguaggio per usare la base dati.*

- **SQL** – Structured Query Language è il linguaggio standardizzato che assolve a funzioni di **DDL**, **DML** e **QL**
  - Portabile da un DBMS a un altro
  - Utile e necessario anche con le query QBE di Access
  - Query QBE di Access → codice SQL
  - Codice SQL → Query QBE di Access

# Il database degli esempi (1)



**Impiegati** ( ID, Nome, Cognome, Residenza, Stipendio, *Dipartimento* )

**Dipartimenti** ( Codice, Descrizione, Sede, *Manager* )

# Il database degli esempi (2)

## 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

## 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



# SQL un esempio di query

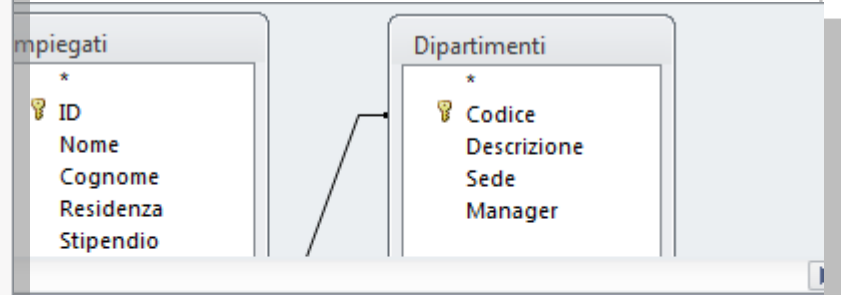
Nome, Cognome, Stipendio e sede di lavoro dei dipendenti con retribuzione superiore a 50000 euro

```
SELECT Nome, Cognome, Stipendio, Sede
FROM Impiegati INNER JOIN Dipartimenti ON
     Impiegati.Dipartimento = Dipartimenti.Codice
WHERE Stipendio > 50000;
```

Query1

Nome	Cognome	Stipendio	Sede
Margherita	Colombi	65000	Torino
Franco	Volpi	61000	Roma
Ugo	Boss	85000	Roma
Mario	Gatti	57000	Torino
Laura	Moretti	52600	Milano
Erica	Bruni	61500	Torino
*			

Record: 1 di 6



Campo:	Nome	Cognome	Dipartimento	Sede	Stipendio
Tabella:	Impiegati	Impiegati	Impiegati	Dipartimenti	Impiegati
Ordinamento:					
Mostra:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Criteri:					>50000

# Caratteristiche generali di SQL (1)

- Linguaggio **standard** per la gestione di data base relazionali
- Linguaggio **dichiarativo**:
  - si dichiara cosa si vuole ottenere e non come ottenerlo
- Estensione dell'algebra relazionale: **calcoli**, **ordinamenti**, **raggruppamenti**
- Visione tabellare dei dati:
  - opera su gruppi di righe o sull'intera tabella, non su una riga per volta
- **Identificatori**:
  - nomi di tabelle e di colonne di lunghezza massima di 18 caratteri
- Colonne specificate con la **dot notation**:
  - *NomeTabella.NomeColonna*
  - **Obbligatoria** solo in caso di ambiguità

# Caratteristiche generali di SQL (2)

- Operatori **aritmetici** e **relazionali**
  - `+` `-` `*` `/` `^` `&` `>` `<` `=` `<=` `>=` `<>` **Between** **IN** **LIKE** ...
- Operatori **logici**:
  - **AND** **OR** **NOT** **XOR** ...
- **Stringhe** di caratteri delimitate con `'` oppure `"`
- Date delimitate con `#` (Access)
- Assenza di informazioni, valore nullo: **NULL**
  - Controllato con il predicato **IS NULL**
  - Esempio: `Dipartimento IS NULL`
  - **Errore** ➔: `Dipartimento = NULL`
- Tipi per i dati:
  - **INTEGER**, **DECIMAL**, **FLOAT**, **CHARACTER**, **DATE**, **TIME**



---

## **SQL come DDL e DML**

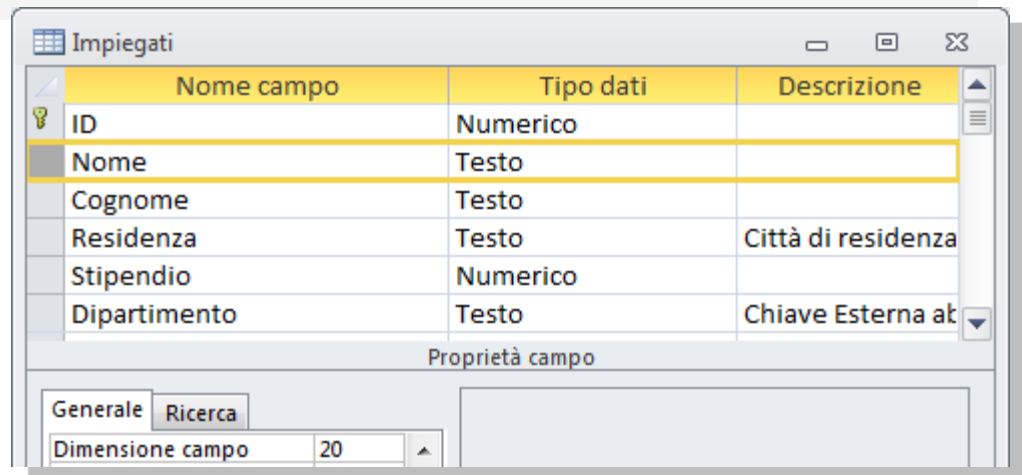
# Definizione delle tabelle (1)

## Creazione della tabella **Impiegati**

```
CREATE TABLE Impiegati (  
  ID          smallint primary key,  
  Nome        char(20) not null,  
  Cognome     char(30) not null,  
  Residenza   char(20) default '*** Manca Residenza',  
  Stipendio   decimal(9,2),  
  Dipartimento char(5) references Dipartimenti(Codice)  
);
```

Integrità  
referenziale

Access fa uso di  
interfacce grafiche →

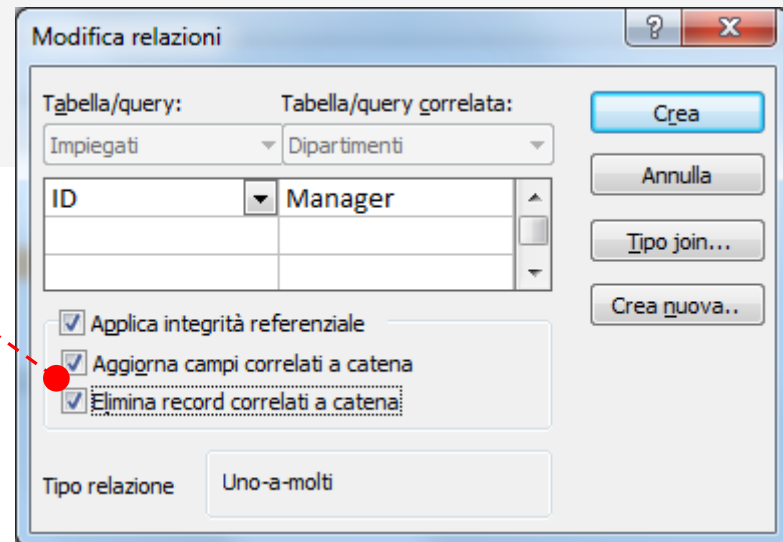


# Definizione delle tabelle (2)

## Creazione della tabella **Dipartimenti**

```
CREATE TABLE Dipartimenti (  
    Codice                char(5),  
    Descrizione           char(20) not null,  
    Sede                  char(20),  
    Manager                smallint,  
    Primary Key (Codice),  
    Unique (Descrizione),  
    Foreign Key (Manager) references Impiegati(ID)  
        On Delete set null  
        On Update cascade  
);
```

Per cancellazioni  
e variazioni di ID  
→ **Sconsigliato**



# Cambiamento della struttura di una tabella

- Aggiunta del campo *Nascita* a **Impiegati**

```
ALTER TABLE Impiegati  
ADD Nascita date;
```

- Eliminazione di *Residenza* da **Impiegati**

```
ALTER TABLE Impiegati  
DROP Residenza;
```

- La coppia di attributi: *Cognome*, *Nome* indicizzata e non duplicabile

```
CREATE UNIQUE INDEX IndiceImpiegati  
ON Impiegati(Cognome, Nome);
```

- Eliminazione della tabella **Impiegati**

```
DROP TABLE Impiegati;
```

# Manipolazione dei dati (1)

- Inserimento di un record nella tabella **Impiegati**

```
INSERT INTO Impiegati  
(ID, Nome, Cognome, Residenza, Stipendio, Dipartimento)  
VALUES(20, 'Mario', 'Rossini', 'Caserta', 31500, 'Mag');
```

```
INSERT INTO Impiegati  
VALUES(21, 'Enrico', 'Grossi', 'Nuoro', 28800, 'Prod');
```

Comandi equivalenti

- Inserimento di un record con campi mancanti:

```
INSERT INTO Impiegati (ID, Nome, Cognome, Stipendio)  
VALUES(22, 'Bruno', 'Locatelli', 33000);
```

- *Rossini* lavora in *Produzione*, non in magazzino:

```
UPDATE Impiegati  
SET Dipartimento = 'Prod'  
WHERE Matricola = 20;
```

# Manipolazione dei dati (2)

- Aumento del 5% ai dipendenti della *Produzione*

```
UPDATE Impiegati  
SET Stipendio = Stipendio * 1.05  
WHERE Dipartimento = 'Prod';
```

- Eliminazione del dipendente con *ID = 20*:

```
DELETE FROM Impiegati  
WHERE ID = 20;
```

- Cancellazione di tutti i dipendenti del reparto *R&S*:

```
DELETE FROM Impiegati  
WHERE Dipartimento = 'R&S';
```

```
UPDATE Impiegati SET Stipendio = Stipendio * 1.15;  
DELETE FROM Impiegati;
```

Cosa fanno?

---

**SQL come QL**

# Il comando Select

Per estrarre informazioni dal database si usa il comando **SELECT**. Nella sua struttura base ha la seguente sintassi:

```
SELECT Exp1, Exp2, .. , ExpN  
FROM Tabelle  
WHERE Condizioni ;
```

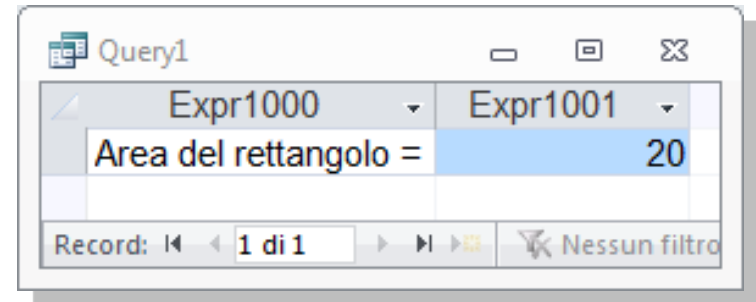
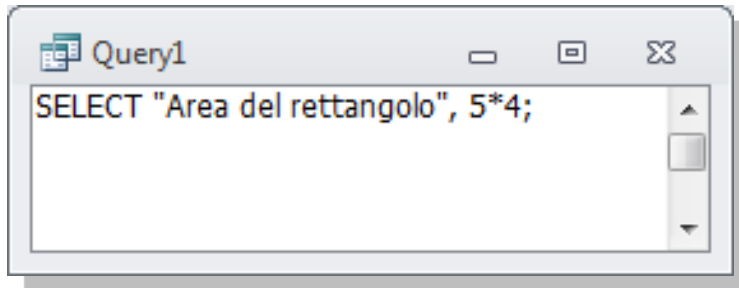
- *Exp1, Exp2, ...* espressioni sui valori delle colonne (e non solo)
- Estensione delle interrogazioni dell'algebra relazionale
  - **Esecuzione di calcoli**
  - **Ordinamenti**
  - **Raggruppamenti**
- SELECT significa Visualizza, Mostra
- Formato libero; come tutti i comandi SQL termina con “;”



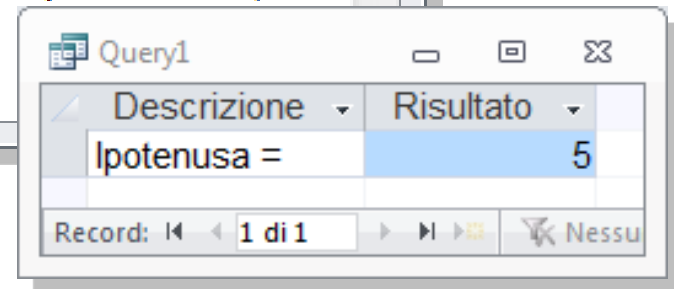
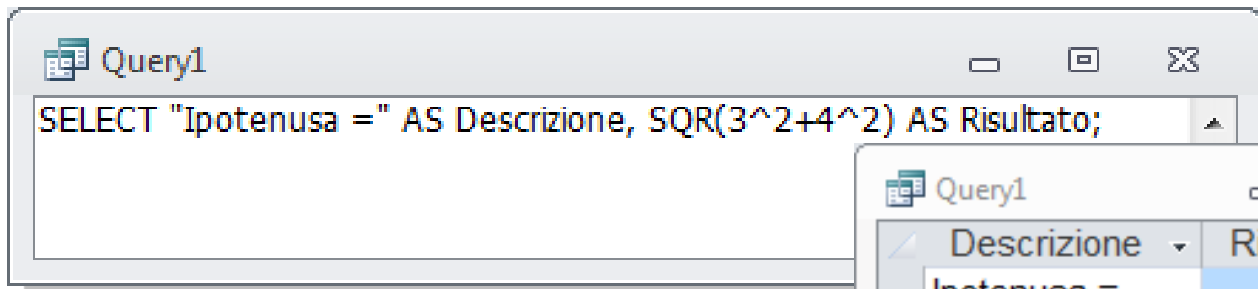
# SELECT come Calcolatrice

- In alcuni DBMS, Access compreso, **SELECT** può essere usato senza altre clausole, per visualizzare il valore di un'espressione

## 1. Area del rettangolo di lati 4 e 5

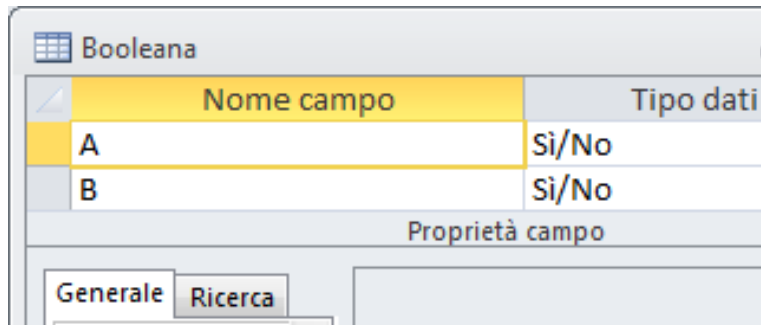


## 2. Ipotenusa di un triangolo rettangolo con cateti 3 e 4

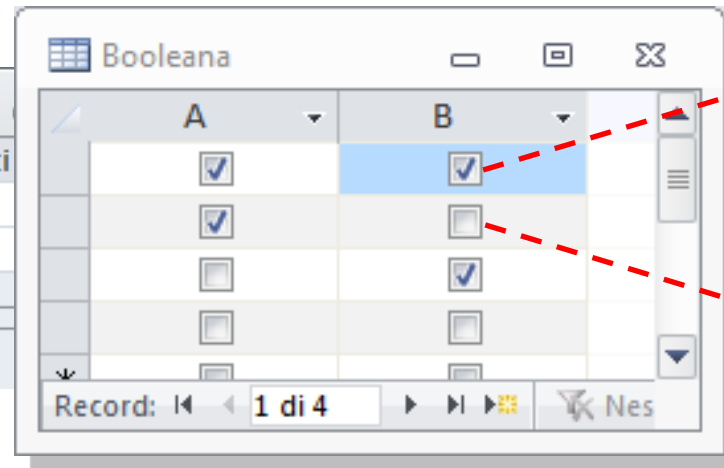


# Le tavole di verità di AND, OR, ..

- La tabella **Booleana** contiene 4 righe con i valori delle grandezze booleane *A* e *B* come nella figura



Nome campo	Tipo dati
A	Si/No
B	Si/No



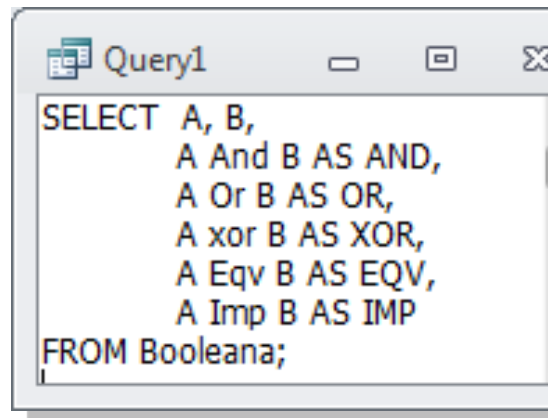
A	B
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>

1

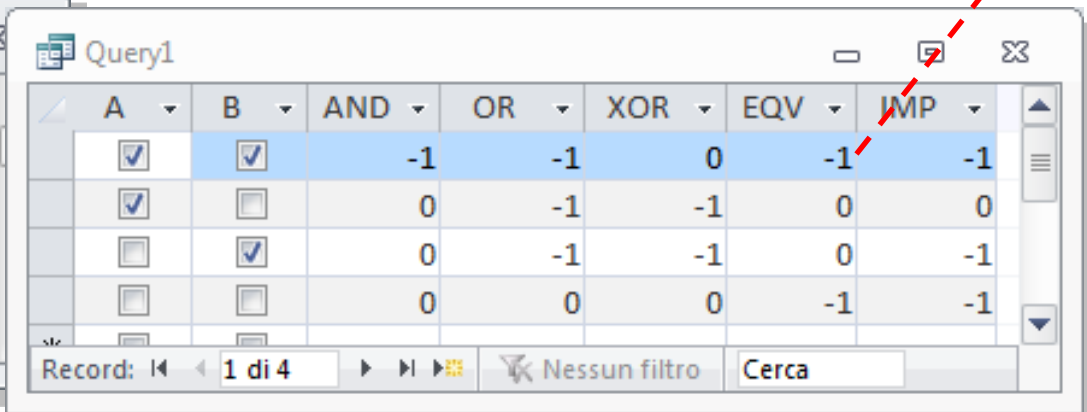
0

V

- Uso degli operatori logici AND, OR, XOR, EQV, IMP



```
SELECT A, B,  
       A And B AS AND,  
       A Or B AS OR,  
       A xor B AS XOR,  
       A Eqv B AS EQV,  
       A Imp B AS IMP  
FROM Booleana;
```



A	B	AND	OR	XOR	EQV	IMP
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	-1	-1	0	-1	-1
<input checked="" type="checkbox"/>	<input type="checkbox"/>	0	-1	-1	0	0
<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	-1	-1	0	-1
<input type="checkbox"/>	<input type="checkbox"/>	0	0	0	-1	-1

---

## **Interrogazioni su una sola tabella**

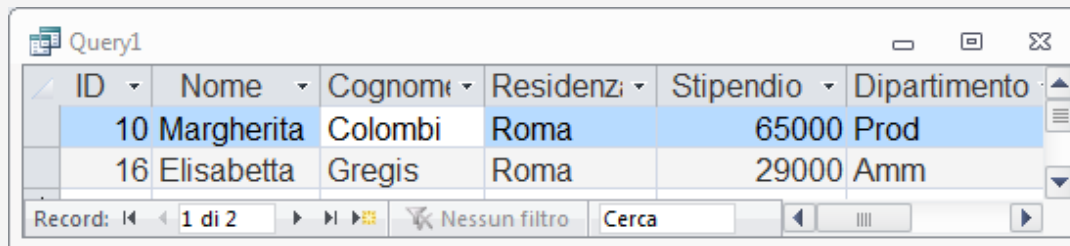
# Interrogazioni su una sola tabella (1)

- Algebra relazionale: **proiezioni** e **selezioni**
- *ID, Cognome* e *Nome* dei dipendenti torinesi della produzione

```
SELECT ID, Cognome, Nome
FROM Impiegati
WHERE Dipartimento = 'Prod' AND Residenza = 'Torino';
```

- Tutti i dati dei dipendenti di Roma

```
SELECT *
FROM Impiegati
WHERE Residenza = 'Roma';
```



ID	Nome	Cognome	Residenza	Stipendio	Dipartimento
10	Margherita	Colombi	Roma	65000	Prod
16	Elisabetta	Gregis	Roma	29000	Amm

```
SELECT * FROM Impiegati WHERE Residenza = 'Roma';
```

Formato libero!

# Interrogazioni su una sola tabella (2)


- **DISTINCT** per non avere righe duplicate

```
SELECT Residenza  
FROM Impiegati;
```

Forme equivalenti

```
SELECT ALL Residenza  
FROM Impiegati;
```

```
SELECT DISTINCT Residenza  
FROM Impiegati;
```



Residenza
Bari
Cagliari
Firenze
Palermo
Perugia
Roma
Torino
Venezia

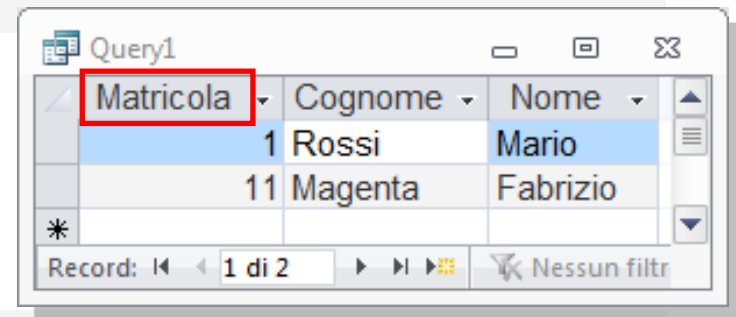


Residenza
Torino
Palermo
Torino
Roma
Torino
Bari
Cagliari
Firenze
Roma
Venezia
Firenze
Perugia

# Interrogazioni su una sola tabella (3)

- **Ridenominazione** dei campi: **AS**

```
SELECT ID AS Matricola, Cognome, Nome  
FROM Impiegati  
WHERE Dipartimento = 'Prod' AND Residenza = 'Torino';
```



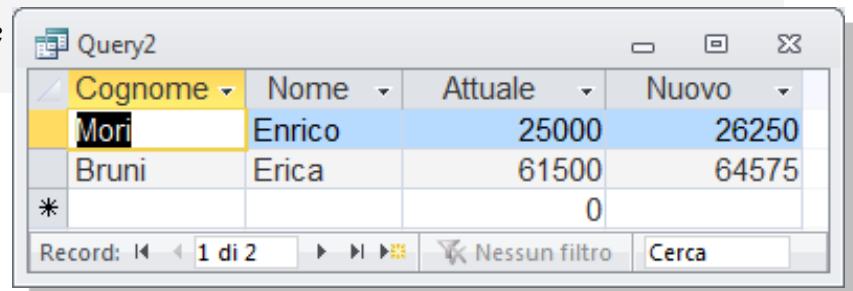
Query1

Matricola	Cognome	Nome
1	Rossi	Mario
11	Magenta	Fabrizio

Record: 1 di 2

- Esecuzione di **calcoli** sui campi

```
SELECT Cognome, Nome, Stipendio AS Attuale,  
       Stipendio*1.05 AS Nuovo  
FROM Impiegati  
WHERE Dipartimento = 'Mag';
```



Query2

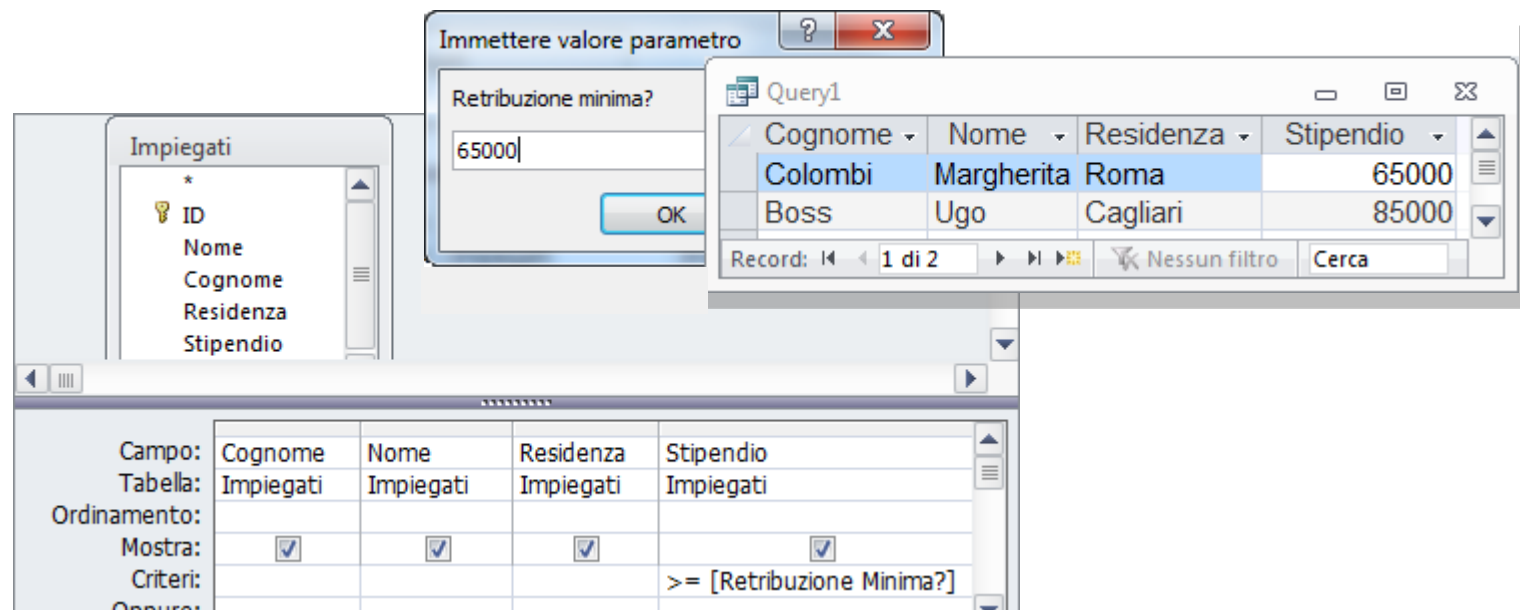
Cognome	Nome	Attuale	Nuovo
Mori	Enrico	25000	26250
Bruni	Erica	61500	64575

Record: 1 di 2

# Interrogazioni parametriche

- Interrogazioni **parametriche** (con Access)

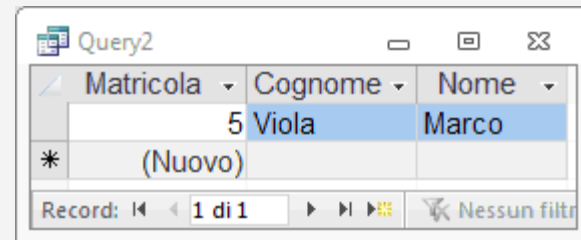
```
SELECT Cognome, Nome, Residenza, Stipendio  
FROM Impiegati  
WHERE Stipendio >= [Retribuzione minima];
```



# Ricerca di valori nulli: IS NULL

- Ricerca di valori mancanti: **IS NULL**

```
SELECT ID AS Matricola, Cognome, Nome  
FROM Impiegati  
WHERE Dipartimento IS NULL;
```

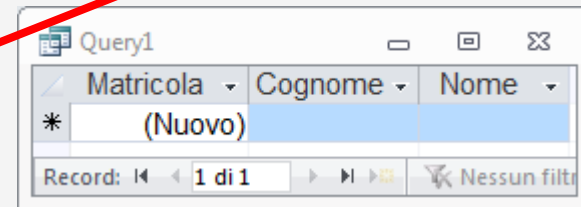


Matricola	Cognome	Nome
5	Viola	Marco
*	(Nuovo)	

Record: 1 di 1 Nessun filtro

Da evitare

```
SELECT ID AS Matricola, Cognome, Nome  
FROM Impiegati  
WHERE Dipartimento = NULL;
```



Matricola	Cognome	Nome
5	Viola	Marco
*	(Nuovo)	

Record: 1 di 1 Nessun filtro

```
SELECT ID AS Matricola, Cognome, Nome  
FROM Impiegati  
WHERE Dipartimento = "";
```



# Proiezioni e selezioni con SQL

- **Proiezioni** → SELECT elenco colonne

- **Proiezione di Impiegati** su *Cognome, Nome, ID*

```
SELECT Cognome, Nome, ID  
FROM Impiegati;
```

Ci sono righe duplicate?

- **Selezioni** → WHERE condizione di selezione

- **Selezione di Impiegati** per *Stipendio < 31000*

```
SELECT *  
FROM Impiegati  
WHERE Stipendio < 31000;
```

Ci sono righe duplicate?

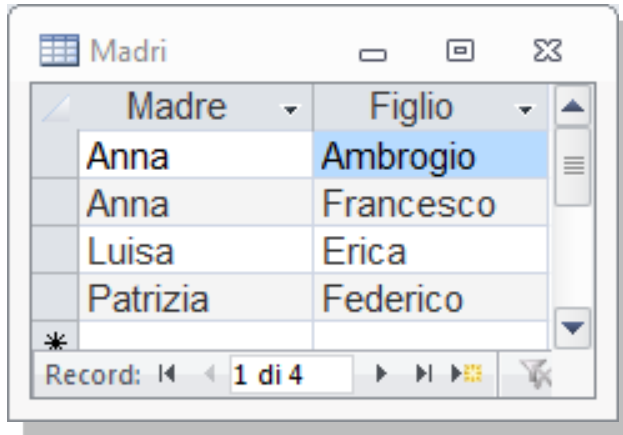
- **DISTINCT** garantisce l'assenza di righe duplicate

---

## **Interrogazioni su più tabelle**

# JOIN ed SQL

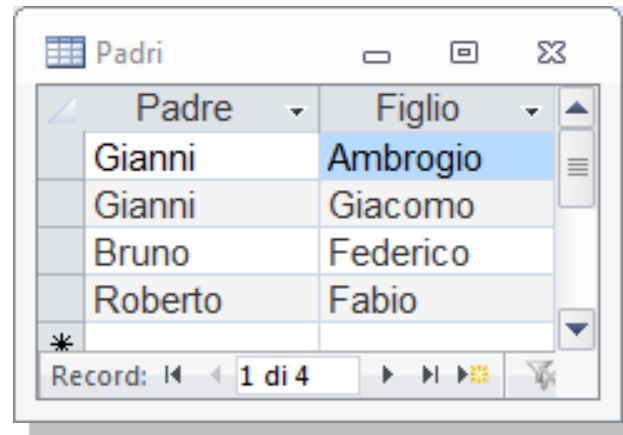
- Per capire come agisce SELECT con più tabelle usiamo :



Madri

Madre	Figlio
Anna	Ambrogio
Anna	Francesco
Luisa	Erica
Patrizia	Federico

Record: 1 di 4



Padri

Padre	Figlio
Gianni	Ambrogio
Gianni	Giacomo
Bruno	Federico
Roberto	Fabio

Record: 1 di 4

- Componiamo **Madri** e **Padri** per ottenere:
  - Prodotto Cartesiano
  - Equi Join
  - Left Join
  - Right Join

# Prodotto Cartesiano

Dati privi di valore informativo

```
Query1
SELECT Madre, Madri.Figlio, Padri.Figlio, Padre
FROM Madri, Padri;
```

Query1

	Madri	Padri
	* Madre Figlio	* Padre Figlio

Campo:	Madre	Figlio	Figlio	Padre
Tabella:	Madri	Madri	Padri	Padri
Ordinamento:				
Mostra:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteri:				

Query1

Madre	Madri.Figlio	Padri.Figlio	Padre
Anna	Ambrogio	Ambrogio	Gianni
Anna	Francesco	Ambrogio	Gianni
Luisa	Erica	Ambrogio	Gianni
Patrizia	Federico	Ambrogio	Gianni
Anna	Ambrogio	Giacomo	Gianni
Anna	Francesco	Giacomo	Gianni
Luisa	Erica	Giacomo	Gianni
Patrizia	Federico	Giacomo	Gianni
Anna	Ambrogio	Federico	Bruno
Anna	Francesco	Federico	Bruno
Luisa	Erica	Federico	Bruno
Patrizia	Federico	Federico	Bruno
Anna	Ambrogio	Fabio	Roberto
Anna	Francesco	Fabio	Roberto
Luisa	Erica	Fabio	Roberto
Patrizia	Federico	Fabio	Roberto

Record: 3 di 16 Nessun filtro Cerca

# Equi Join: due sintassi

AS opzionale

```
SELECT Madre, M.Figlio, P.Figlio, Padre  
FROM Madri AS M, Padri AS P  
WHERE M.Figlio = P.Figlio;
```

Query1

Madre	M.Figlio	P.Figlio	Padre
Anna	Ambrogio	Ambrogio	Gianni
Patrizia	Federico	Federico	Bruno

Record: 1 di 2

Sintassi usata  
da Access

Query1

Madri

Madre

Figlio

Padri

Padre

Figlio

Query1

```
SELECT M.Madre, M.Figlio, P.Figlio, P.Padre  
FROM Madri AS M INNER JOIN Padri AS P ON M.Figlio = P.Figlio;
```

Campo:	Madre	Figlio	Figlio	Padre
Tabella:	Madri	Madri	Padri	Padri
Ordinamento:				
Mostra:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteri:				

# Left Join: tutte le madri

Query1

```
SELECT Madre, M.Figlio, P.Figlio, Padre
FROM Madri M LEFT JOIN Padri P ON M.Figlio = P.Figlio;
```

Query1

Madre	M.Figlio	P.Figlio	Padre
Patrizia	Federico	Federico	Bruno
Luisa	Erica		
Anna	Francesco		
Anna	Ambrogio	Ambrogio	Gianni

Record: 1 2 di 4 Nessun filtro Cerca

Madri

Madre	Figlio
*	
Madre	
Figlio	

Padri

Padre	Figlio
*	
Padre <td></td>	
Figlio <td></td>	

Campo: Madre Figlio Figlio Padre  
Tabella: Madri Madri Padri Padri  
Ordinamento:  
Mostra: ☒ ☒ ☒ ☒  
Criteri:

Valori Nulli

# Right Join: tutti i padri

Query1

```
SELECT Madre, M.Figlio, P.Figlio, Padre
FROM Madri M RIGHT JOIN Padri P ON M.Figlio = P.Figlio;
```

Query1

Madre	Madri.Figl	Padri.Figli	Padre
Patrizia	Federico	Federico	Bruno
Anna	Ambrogio	Ambrogio	Gianni
		Fabio	Roberto
		Giacomo	Gianni

Record: 3 di 4

Nessun filtro

Cerca

Valori Nulli

Query1

Madri

Padri

Madre

Figlio

Padre

Figlio

Campo: Madre Figlio Figlio Padre

Tabella: Madri Madri Padri Padri

Ordinamento:

Mostra: ☒ ☒ ☒ ☒

Criteri:

# Osservazioni sul Join

- **EQUI JOIN**: include le righe del prodotto cartesiano con certe proprietà
- **NATURAL JOIN**: in Access non c'è
  - `SELECT Madre, M.Figlio AS Figlio, Padre`  
`FROM Madri M INNER JOIN Padri P ON M.Figlio = P.Figlio;`
- Partecipano al JOIN solo le righe con corrispondenti: **alcune spariscono**
- **LEFT JOIN**: chi sono le madri senza padri corrispondenti
  - `Select Madre From Madri M LEFT Join Padri P ... Where P.Figlio IS NULL;`
- **RIGHT JOIN**: chi sono i padri senza madri corrispondenti
  - `Select Padre From Madri M RIGHT Join Padri ... WHERE M.Figlio IS NULL;`
- **FULL JOIN**: in Access non c'è
  - `( SELECT ... LEFT JOIN ... ) UNION ( SELECT ... RIGHT JOIN ... );`
  - Non serve a nulla (poco)



---

## **Esempi di interrogazioni**

# Esempi (1)

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



- Elenco dei dipendenti che lavorano in un dipartimento di *Roma*, con *Cognome*, *Nome* e descrizione del dipartimento

Congiunte le due tabelle, si opera una selezione per *Sede* = “*Roma*” e una proiezione sui campi richiesti

```
SELECT Cognome, Nome, Descrizione
FROM Impiegati, Dipartimenti
WHERE Dipartimento = Codice AND Sede = 'Roma';
```

```
SELECT Cognome, Nome, Descrizione
FROM Impiegati INNER JOIN Dipartimenti ON
    Dipartimento = Codice
WHERE Sede = 'Roma';
```

Sintassi più elegante

# Esempi (1 bis)

Query1

```
SELECT Impiegati.Cognome, Impiegati.Nome, Dipartimenti.Descrizione  
FROM Impiegati INNER JOIN Dipartimenti ON  
Impiegati.Dipartimento = Dipartimenti.Codice  
WHERE (((Dipartimenti.Sede)="Roma"));
```

Query1

Cognome	Nome	Descrizione
Volpi	Franco	Amministrazione
Boss	Ugo	Direzione Generale
Gregis	Elisabetta	Amministrazione
Bianco	Anita	Direzione Generale

Record: 2 di 4

Nessun filtro

Cerca

Query1

Impiegati

- ID
- Nome
- Cognome
- Residenza
- Stipendio
- Dipartimento

Dipartimenti

- Codice
- Descrizione
- Sede
- Manager
- Campo1

Campo: Cognome

Tabella: Impiegati

Ordinamento:

Mostra: ☒

Criteri:

Oppure:

Campo:	Nome	Descrizione	Sede
Cognome	Nome	Descrizione	Sede
Tabella: Impiegati	Impiegati	Dipartimenti	Dipartimenti
Ordinamento:			
Mostra: <input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Criteri:			"Roma"
Oppure:			

## Esempi (2)

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



- *Cognome, Nome, Stipendio* e *Descrizione* del dipartimento dei dipendenti che lavorano a *Torino* e hanno retribuzione superiore a 30000.

Congiunte le due tabelle, si opera una selezione per *Sede* = “*Torino*”, *Stipendio* > 30000 e si esegue una proiezione sui campi richiesti

```
SELECT Cognome, Nome, Stipendio, Descrizione
FROM Impiegati INNER JOIN Dipartimenti ON
      Dipartimento = Codice
WHERE Sede = 'Torino' AND Stipendio > 30000;
```

## Esempi (3)

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



- *Cognome, Nome, Residenza, Descrizione* del dipartimento e *Sede* dei dipendenti che lavorano in una città diversa da quella dove risiedono.

Congiunte le due tabelle, si opera una selezione per *Residenza* <> *Sede* e si esegue una proiezione sui campi richiesti

```
SELECT Cognome, Nome, Residenza, Descrizione, Sede
FROM Impiegati, Dipartimenti
WHERE Dipartimento = Codice AND
      Residenza <> Sede;
```

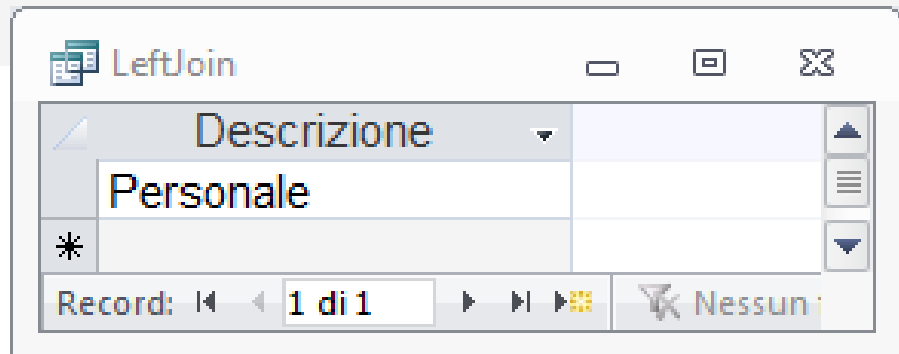
## Esempi (4)

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

- Dipartimenti senza dipendenti assegnati

Congiunte **Dipartimenti** e **Impiegati** con un LEFT JOIN si selezionano le righe con valori nulli in corrispondenza del campo *ID* e si esegue una proiezione ...

```
SELECT Descrizione  
FROM Dipartimenti LEFT JOIN Impiegati ON  
      Codice = Dipartimento  
WHERE ID IS NULL;
```



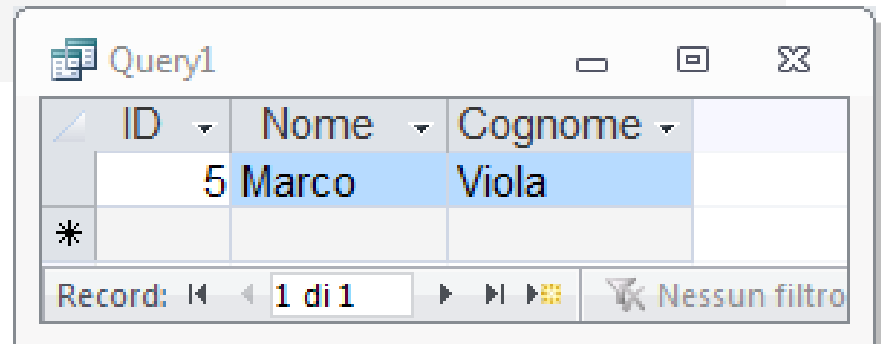
# Esempi (5)

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

- *ID*, *Nome* e *Cognome* degli impiegati non assegnati ad alcun dipartimento

Si esegue una Selezione su **Impiegati** per *Dipartimento* a valori nulli e si esegue una proiezione sui campi richiesti

```
SELECT ID, Nome, Cognome  
FROM Impiegati  
WHERE Dipartimento IS NULL;
```



ID	Nome	Cognome
5	Marco	Viola

Record: 1 di 1 Nessun filtro

# Esempi (6)

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



- *ID*, *Nome*, *Cognome*, *Stipendio*, codice del dipartimento, oltre a *ID*, *Nome* e *Cognome* del rispettivo capo per gli impiegati che hanno una retribuzione superiore a 40000 euro.

Per collegare un impiegato con i dati del rispettivo capo:

1. T1 = Impiegati **JOIN** Dipartimenti con *Dipartimento* = *Codice*
2. T2 = T1 **JOIN** Impiegati1 *Manager* = *Impiegati1.ID*
3. ... questa è materia per sole selezioni e proiezioni ...

```
SELECT I.ID,I.Nome,I.Cognome,I.Stipendio,I.Dipartimento,  
       M.ID,M.Nome,M.Cognome  
FROM Impiegati I, Dipartimenti D, Impiegati M  
WHERE I.Dipartimento = D.Codice AND D.Manager = M.ID AND  
       I.Stipendio > 40000;
```



# Esempi (6 bis)

Query1

I.ID	I.Nome	I.Cognom	Stipend	Dipartim	M.ID	M.Nome	M.Cognon
10	Margherita	Colombi	65000	Prod	10	Margherita	Colombi
11	Fabrizio	Magenta	41000	Prod	10	Margherita	Colombi
12	Franco	Volpi	61000	Amm	12	Franco	Volpi
13	Ugo	Boss	85000	Direz	13	Ugo	Boss
14	Mario	Gatti	57000	R&S	14	Mario	Gatti
17	Laura	Moretti	52600	Mkt	13	Ugo	Boss
18	Erica	Bruni	61500	Mag	10	Margherita	Colombi

Record: 2 di 7

```
SELECT I.ID,I.Nome,I.Cognome,I.Stipendio,I.Dipartimento,
       M.ID,M.Nome,M.Cognome
FROM Impiegati I, Dipartimenti D, Impiegati M
WHERE I.Dipartimento = D.Codice AND D.Manager = M.ID AND
       I.Stipendio > 40000 AND I.ID <> M.ID;
```

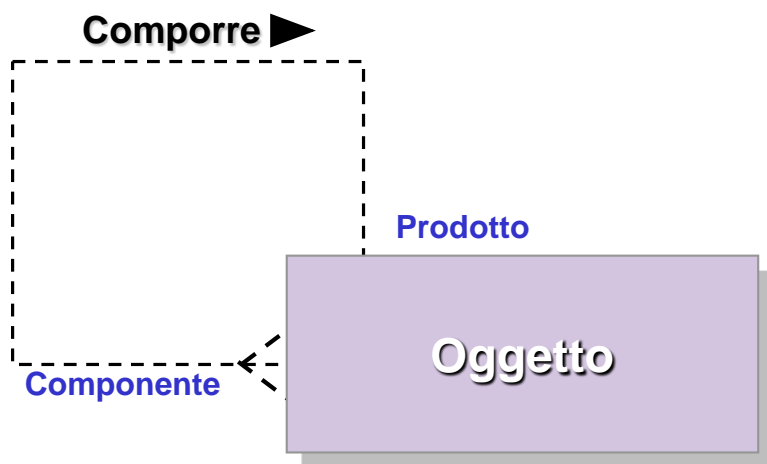
Query1

I.ID	I.Nome	I.Cognom	Stipend	Dipartim	M.ID	M.Nome	M.Cognon
11	Fabrizio	Magenta	41000	Prod	10	Margherita	Colombi
17	Laura	Moretti	52600	Mkt	13	Ugo	Boss
18	Erica	Bruni	61500	Mag	10	Margherita	Colombi

Record: 1 di 3

# Esempi (7)

Oggetti (ID, Descrizione, Qta, ComponenteDi)



Di quali parti è composta una camicia?

ID	Descrizione	Qta	ComponenteDi	F
1	Camicia	1		
2	Manica	2	1	
3	Colletto	1	1	
4	Bottone	10	1	
5	Taschino	1	1	
6	Giacca	1		
7	PartediGiacca	2	6	
8	ParteDiGiacca	5	6	
9	ParteDiGiacca	2	6	
10	ParteDiGiacca	2	6	
11	ParteDiGiacca	3	6	
12	Cappotto	1		
13	ParteDiCappotto	2	12	
14	ParteDiCappotto	2	12	
15	ParteDiCappotto	2	12	
16	ParteDiCappotto	6	12	
17	ParteDiCappotto	2	12	
18	Pantalone	1		
19	ParteDiPantalone	2	18	
20	ParteDiPantalone	2	18	

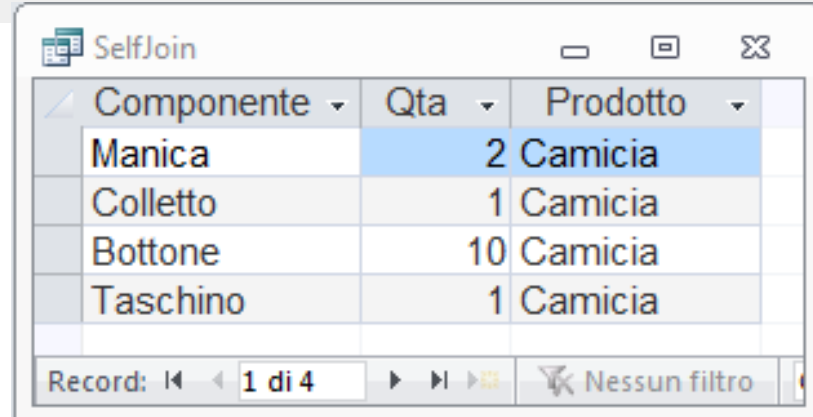
Record: 3 di 20

# Esempi (7 bis)

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

- Non c'è una operazione di self join esplicita ma spesso, per esempio con le tabelle derivate da associazioni ricorsive, serve congiungere una tabella con se stessa, come succede per elencare le componenti di una camicia.

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



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

Record: 1 di 4

Nessun filtro