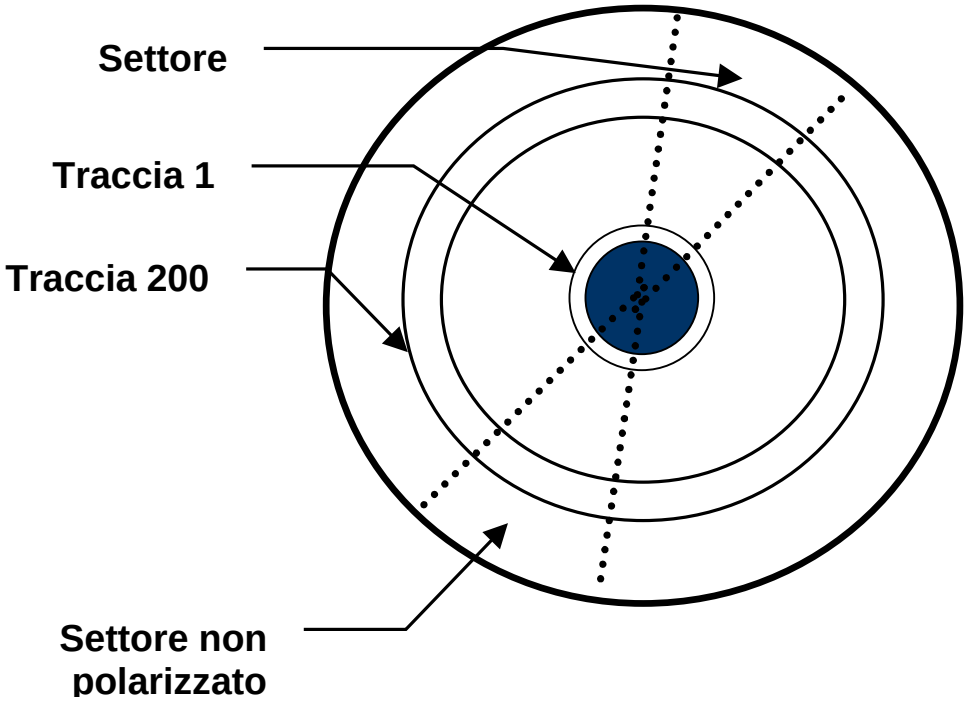
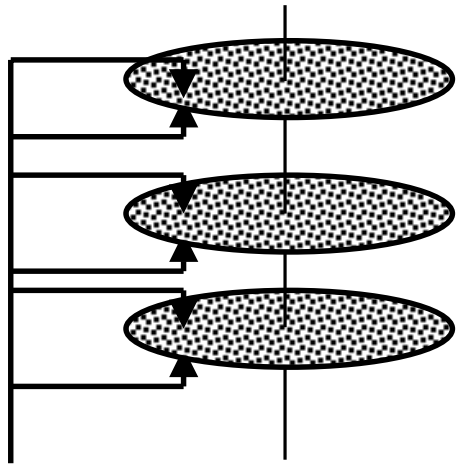
Memorie di massa

Dischi

Floppy disk ed hard disk

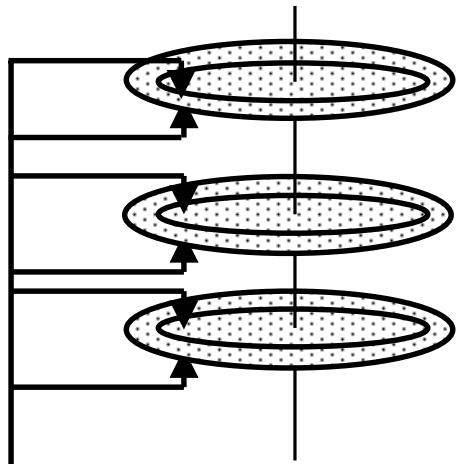


Accesso casuale

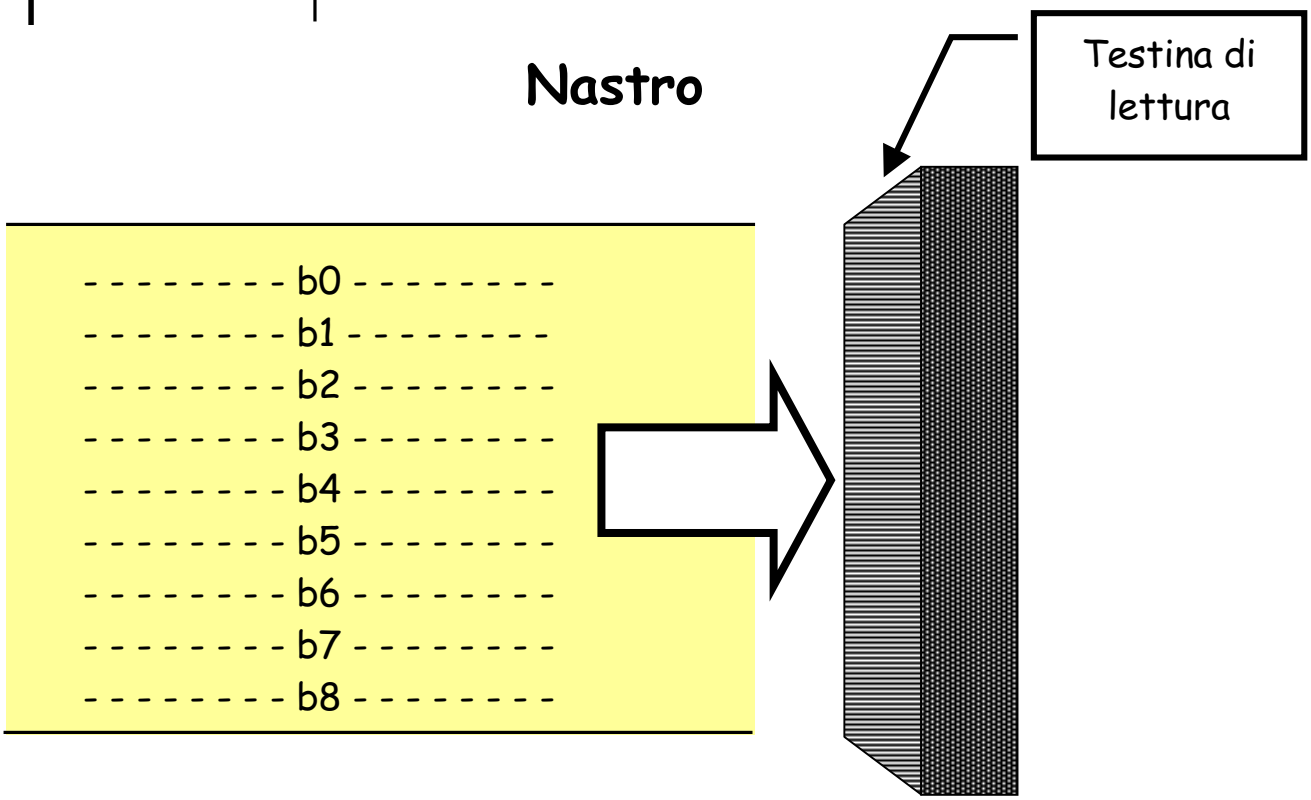


Permette di accedere direttamente ad una specifica informazione collocata ovunque sul disco al contrario dell'accesso sequenziale (nastri) nel quale bisogna scorrere tutto il supporto fino a trovare il dato richiesto

Cilindri



Insieme di tutte le tracce sovrapposte nei vari dischi e che possono essere lette contemporaneamente senza spostare il braccio di lettura



Accesso sequenziale alle informazioni

Operazioni di lettura/scrittura

$$t_{i/u} = t_{seek} + t_{lat} + t_{trasf}$$

Tempo necessario allo spostamento radiale delle testine

Tempo per la rotazione fino al raggiungimento del settore voluto

Tempo di trasferimento dei dati

COSA E' UN DATA-BASE (DB) ?

è l'insieme di dati relativo ad un sistema informativo

COSA CARATTERIZZA UN DB ?

- la struttura dei dati
- le relazioni fra i dati

I REQUISITI DI UN DB SONO:

1. la ridondanza minima
2. i dati non devono essere inutilmente duplicati per problemi di:
   - spazio,
   - gestione,
   - manutenzione,
   - affidabilità,
   - coerenza
3. la permanenza dei dati
4. la base di dati è protetta contro eventi che possano minacciarne l'esistenza e/o l'integrità
5. la condivisione dei dati

   piu' utenti devono potere ad un tempo usare la stessa base di dati (supporto unico, aggiornamento unico, coerenza dei dati, affidabilità, ...)

Schema di una base di dati

Descrizione della struttura dei dati di uno specifico contesto applicativo

Istanza (o occorrenza) di una base di dati

Valore assunto dai dati di un certo DB in un certo istante

Organizzazione degli archivi

Le informazioni contenute in memoria vengono organizzate in record logici

Num Record	Nome	Indirizzo	Telefono
1	Rossi Carlo	Via dei Tigli, 32	02-33249187

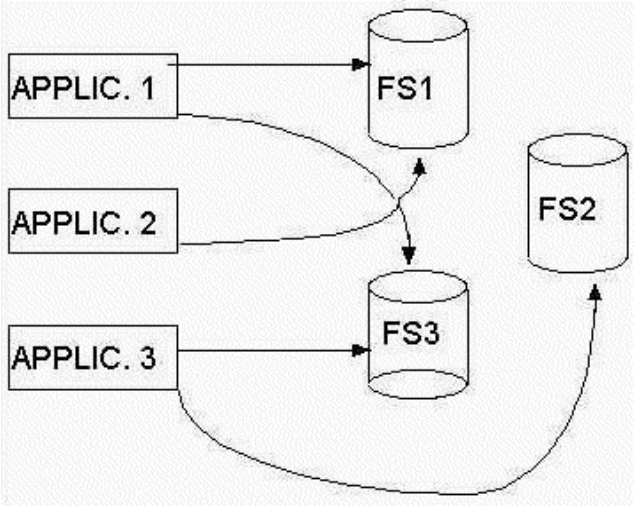
Un record è composto da campi

Più record possono essere contenuti in un file

E' POSSIBILE CONDIVIDERE I DATI?

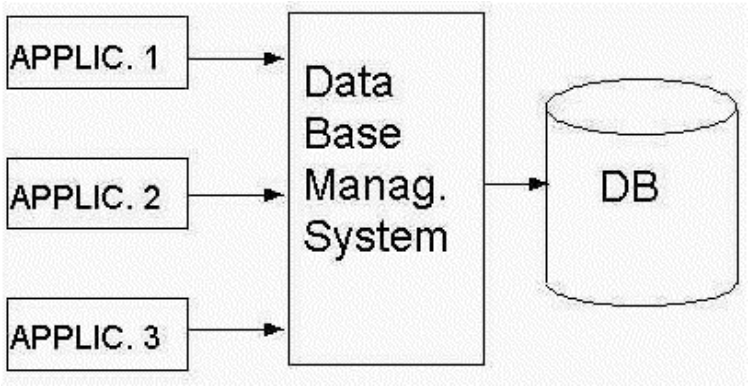
In un sistema informativo ogni utente fa uso dei vari tipi di dati in misura diversa ed in modo diverso (applicazioni diverse)

IERI



Applicazioni diverse, file-system diversi

OGGI



Applicazioni diverse, DBMS, DB unico

Differenze tra archivi separati e basi di dati

Problemi degli archivi separati

1. **Inconsistenza e ridondanza dei dati**

   Vi possono essere differenze tra i valori relativi ad una stessa entità ma riportati in archivi diversi

   La duplicazione di dati crea spreco di memoria

2. **Riservatezza dei dati**

3. **Integrità dei dati**

   L'integrità dei dati viene assicurata dai 'vincoli di consistenza'

   Ad esempio un campo non può assumere valore negativo

   Con archivi separati, l'integrità dei dati viene affidata a programmi applicativi (soggetti ad errori)

4. **Concorrenza**

   Gestire gli accessi contemporanei alla stessa informazione.

Vantaggi dei DBMS

1. **I dati non sono duplicati**
2. **L'accesso ai dati avviene in base a privilegi fissati dal DBMS**
3. **I vincoli di consistenza possono essere fissati all'interno del DBMS**
4. **L'accesso concorrente ai dati è controllato dal DBMS che gestisce la mutua esclusione dei programmi**

Modelli di dati

Gerarchico

Basato sugli alberi

Reticolare

Basato sui grafi

Relazionale

Basato sugli insiemi, dati strutturati in tabelle

Ad oggetti

Basato sulle proprietà degli oggetti

Schema di una base di dati

Descrizione della struttura dei dati di uno specifico contesto applicativo

Istanza (o occorrenza) di una base di dati

Valore assunto dai dati di un certo DB in un certo istante

Linguaggi per la gestione dei dati

DDL (data definition language)

Per definire lo schema della base di dati

(→ Le definizioni dello schema costituiscono il 'Dizionario dei dati')

DML (data manipulation language)

Per inserire, cancellare, modificare i dati

Per effettuare query

Il modello relazionale

- Una base di dati relazionale è una collezione di relazioni
- Una relazione è una tabella costituita da un numero fisso di colonne (dette attributi) e un numero variabile di righe (dette tuple)
- Ciascuna colonna assume valori estratti da uno stesso dominio
- Il numero di colonne di una relazione si chiama grado, il numero di righe cardinalità

Schema di una relazione

E' la descrizione della struttura della relazione

Schema = Nome della relazione + [nome di attributo + domino dell'attributo]

Esempio

```
Relation Conto_corrente (Numero_cc: integer,
Nome: char(20), Indirizzo: char(20), saldo:
integer)
```

Istanza di una relazione

Insieme delle tuple presenti nella base di dati in un certo istante.

Nella definizione formale del modello relazionale è richiesto che le tuple siano tutte distinte.

Restrizione di una tupla

La restrizione di un tupla sugli attributi A della relazione R (indicata con t[A]), è data dalla lista dei valori assunti da t sugli attributi A

Esempio

Sia t=(1, Rossi, v. Anemoni 5, L. 3.678.000) la tupla.

Allora una possibile restrizione è: t[Numero_cc, Nome]=(1, Rossi)

Chiave di una relazione

E' un sottoinsieme K degli attributi che soddisfa le proprietà:

- unicità

  in qualunque istanza di R, non possono esistere due tuple distinte la cui restrizione su K sia uguale

- minimalità

  non è possibile sottrarre a K un attributo senza violare la condizione di unicità

In generale una relazione può avere più di una chiave

Esempio

Record	Nome	Indirizzo	Telefono
1	Rossi Carlo	Via dei Tigli, 32	02-33249187

Numero_cc è una chiave

Nome non è una chiave

(Nome, Indirizzo) non è una chiave

Chiave primaria

Corrisponde alla chiave usata più frequentemente per accedere ai dati

In genere si indica sottolineando gli attributi che la costituiscono:

Conto_corrente (Numero_cc, Nome, Indirizzo, saldo)

Progettazione dei DB

Le fasi di progettazione di un DB sono tre:

1 - Progetto CONCETTUALE

(-> Schema concettuale)

2 - Progetto LOGICO

(-> Schema logico)

3 - Progetto FISICO

(-> Schema fisico)

---

1 - Progetto CONCETTUALE

Lo schema concettuale è la rappresentazione più astratta, la più vicina alla logica umana nella definizione di dati e relazioni. Spesso vengono usati i modelli entità-relazioni

2 - Progetto LOGICO

Lo schema logico dipende fortemente dal DBMS e dal suo modello logico dei dati. Esistono ad esempio DBMS gerarchici, reticolari e relazionali.

Nello schema logico vengono definite anche le viste (dette anche schemi esterni) cioè le parti del DB messe a disposizione delle applicazioni.

3 - Progetto FISICO

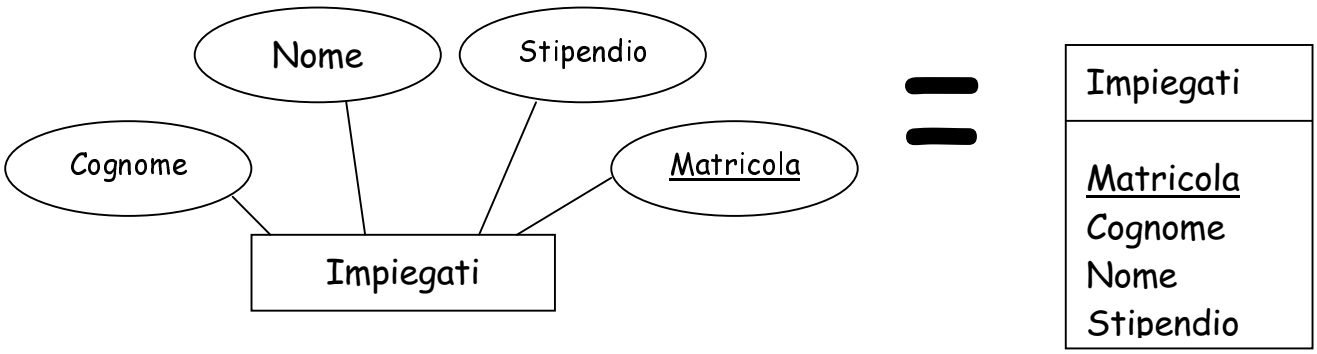
Lo schema fisico definisce come le strutture definite nel progetto logico vanno implementate nell'archivio

Diagrammi entità relazioni

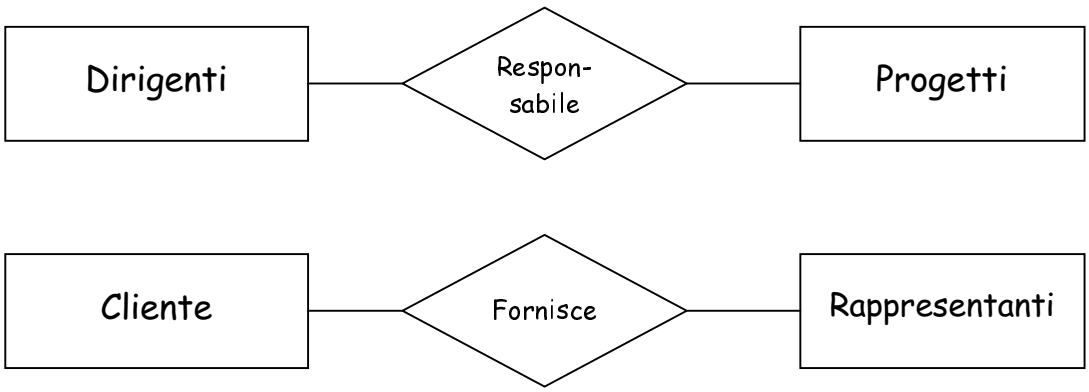
Entità un qualsiasi oggetto concettuale che caratterizza l'applicazione in questione e che può essere individuato e distinto dagli altri

Attributi insieme di valori che caratterizzano un'entità

Attributi chiave insieme degli attributi sufficienti ad identificare univocamente un'entità all'interno di un certo insieme



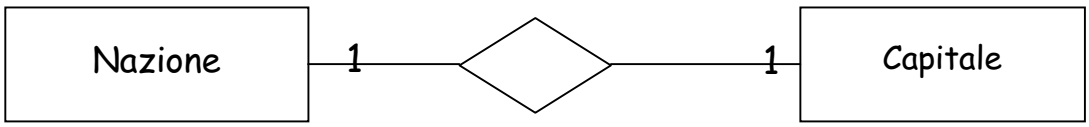
Relazioni dipendenze o associazioni di interesse informativo tra le entità rappresentate



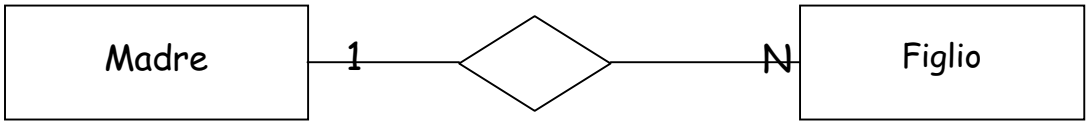
Cardinalità delle relazioni

La relazione R che lega due entità E1 ed E2 può essere classificata in base alla sua cardinalità:

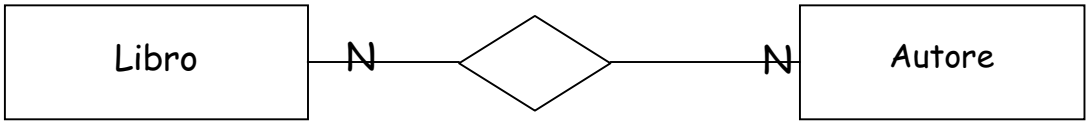
1. R ha cardinalità 1:1 (uno a uno) se ad un elemento di E1 può corrispondere un solo elemento di E2



2. R ha cardinalità 1:N (uno a molti) se ad ogni elemento di E1 possono corrispondere più elementi di E2

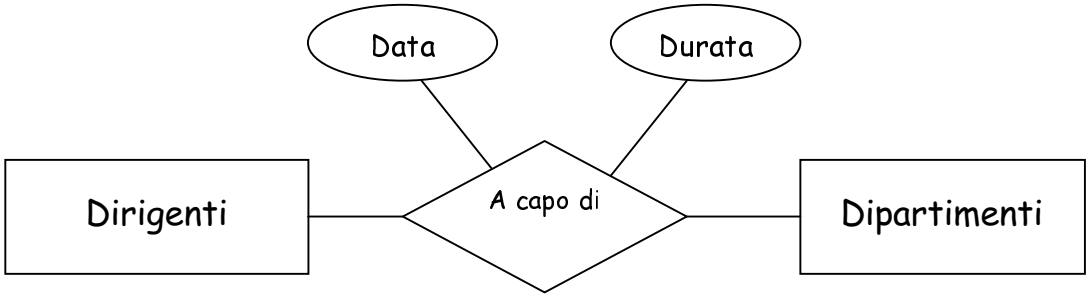


3. R ha cardinalità N:N (molti a molti) se ad ogni elemento di E1 possono corrispondere molti elementi di E2 e viceversa

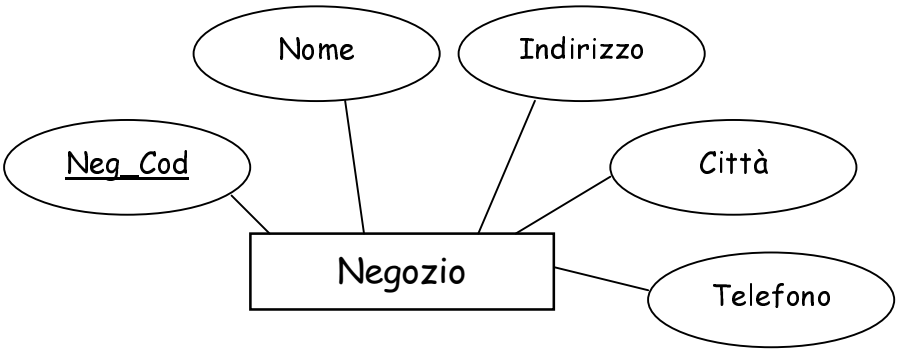


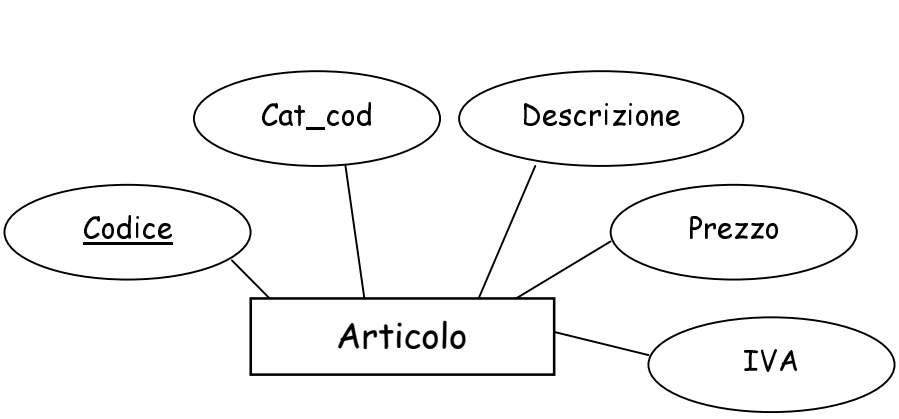
Attributi delle relazioni

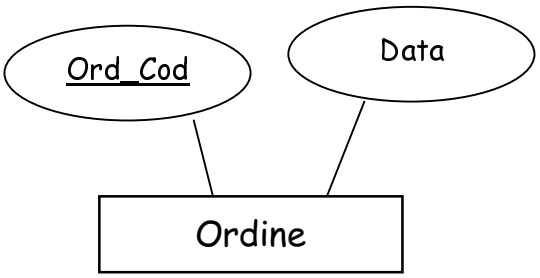
Le relazioni possono essere pensate come insiemi di elementi e quindi è possibile assegnare ad esse delle relazioni

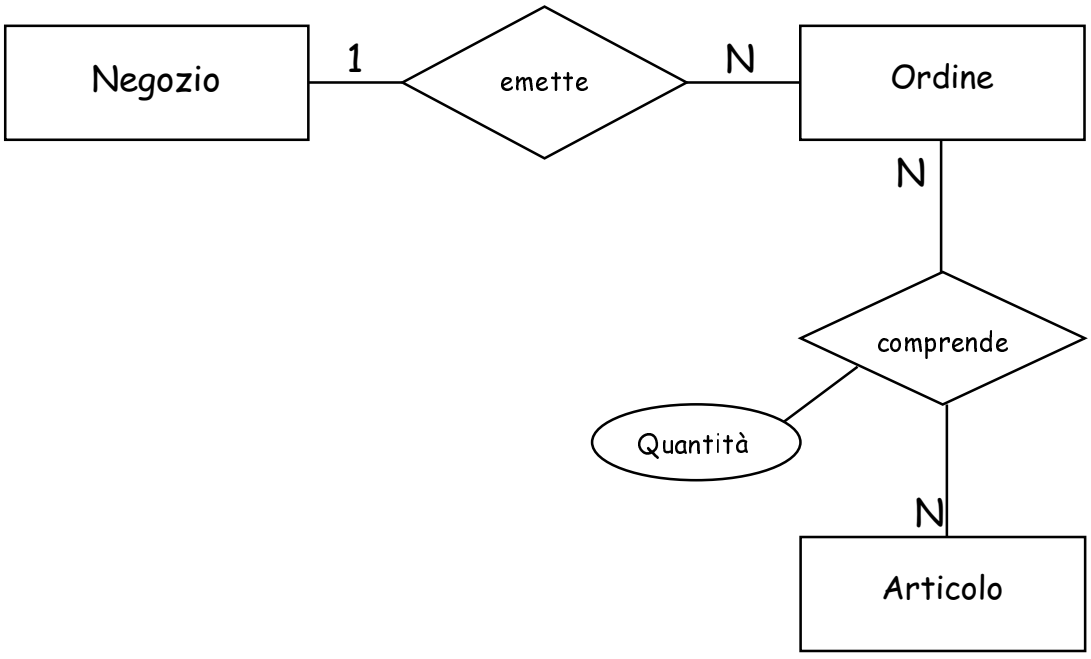


Esempio


Nome
Indirizzo
Neg_Cod
Città
Negozio
Telefono


Cat_cod
Descrizione
Codice
Prezzo
Articolo
IVA


Ord_Cod
Data
Ordine


Negozio
1
emette
N
Ordine
N
comprende
Quantità
N
Articolo

Dagli ERD alle tabelle

Regole per passare dallo schema concettuale (ERD) allo schema logico relazionale (tabella):

1 - Ad ogni entità dell'ERD corrisponde una tabella

2 - Le relazioni fra entità dell'ERD (tabelle T1 e T2) sono affidate al fatto che vi siano particolari colonne in comune (chiavi primarie ed esterne)

2a - Se la relazione è 1:1 agli attributi di T1 si aggiungono quelli che sono chiave primaria di T2 (chiave esterna di T2) e viceversa

2b - Se la relazione è 1:N agli attributi di T2 si aggiungono quelli che sono chiave primaria di T1 (chiave esterna di T2) e non viceversa

2c - Se la relazione è N:N si crea una tabella T3 le cui colonne sono le chiavi primarie di T1 e T2. La chiave primaria di T3 è l'insieme delle sue colonne ed è l'insieme delle chiavi esterne di T1 e T2

La normalizzazione dei dati

La prima forma normale

Condizione: in una tabella gli elementi delle colonne devono essere ad un sol valore, non possono essere array.

Normalizzazione: nel caso di presenza di array, normalizzare la tabella vuol dire produrre tante righe quanti sono gli elementi del vettore

N. protocollo	Uffici interessati					

NO

N. protocollo	Uffici interessati

Si

La seconda forma normale

Condizione: in una tabella con chiave a più attributi, ogni colonna non appartenente alla chiave deve dipendere dall'insieme delle colonne-chiave e non solo da una parte di esse.

Normalizzazione: Normalizzare vuol dire produrre tante tabelle che soddisfino la condizione nel caso in cui in quella originaria essa non sia soddisfatta

Codice città	Codice via	Città	Via
01	01	Roma	Verdi
01	02	Roma	Bianchi
01	03	Roma	Rossi
02	01	Milano	Gialli

NO

Città dipende soltanto da 'Codice città'

Codice città	Codice via	Via
01	01	Verdi
01	02	Bianchi
01	03	Rossi
02	01	Gialli

SI

Codice città	Città
01	Roma
01	Roma
01	Roma
02	Milano

SI

La terza forma normale

Condizione: in una tabella la dipendenza fra le colonne deve essere basata soltanto sulla chiave primaria.

Normalizzazione: Normalizzare vuol dire produrre tante tabelle che soddisfino la condizione nel caso in cui in quella originaria essa non sia soddisfatta

N. protocollo	Mittente	Tipo	Urgenza
1	001	1	Si
2	001	2	No
3	003	1	Si
4	002	2	No
5	003	3	Si

NO

Tipo	Descrizione
1	Lettera
2	Memo
3	Telegramma

NO

L'urgenza dipende dal tipo e non dal numero di protocollo

Tipo	Descrizione	Urgenza
1	Lettera	Si
2	Memo	No
3	Telegramma	Si

SI

Il linguaggio SQL

(structured query language)

La creazione delle relazioni

Costruire una relazione in SQL: CREATE TABLE

Sintassi

```
CREATE TABLE tabella (campo1 tipo
[(dimensioni)] [NOT NULL] [indice1] [,
campo2 tipo [(dimensioni)] [NOT NULL]
[indice2] [, ...]])
```

Esempio

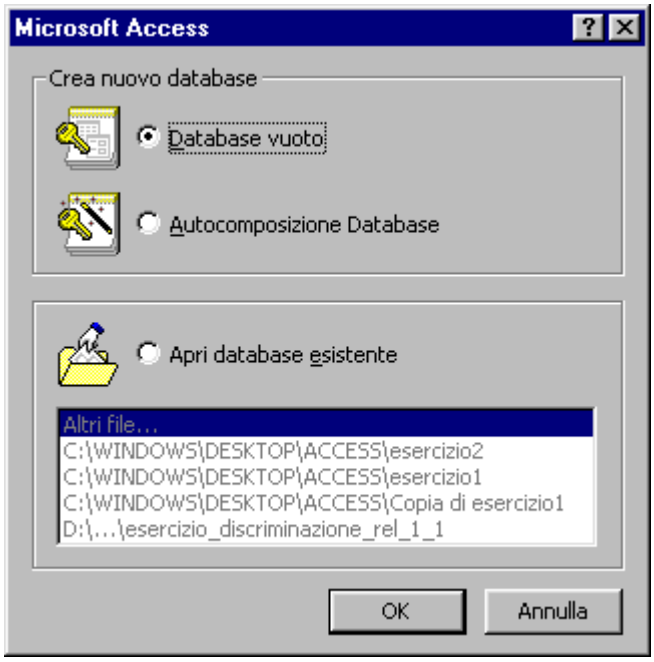
```
CREATE TABLE CONTO_CORRENTE
(NUMERO_CC: INTEGER, PRIMARY KEY, NOT NULL,
NAME:CHAR(20), NOT NULL,
INDIRIZZO: CHAR (20),
SALDO: INTEGER, NOT NULL)
```

Costruire una relazione con Access

Creare un nuovo database

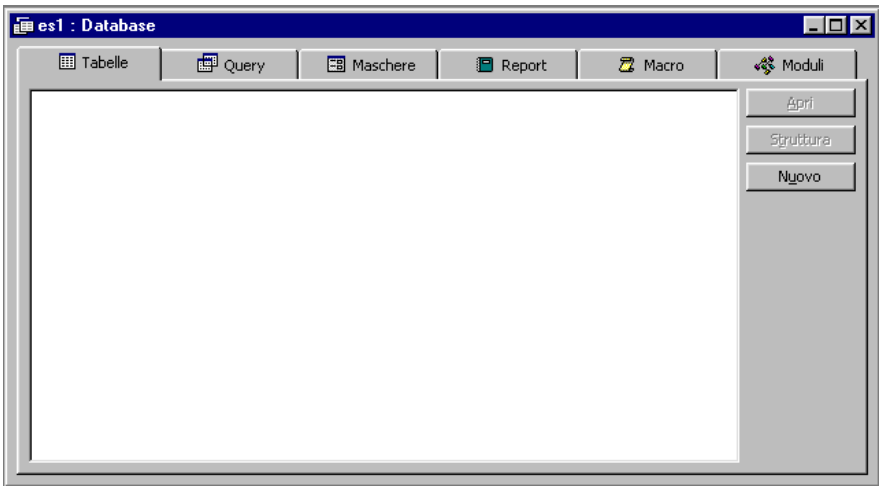
Avviare Access

Scegliere database vuoto nella finestra:



Salvare il file

Apparirà la finestra principale del db

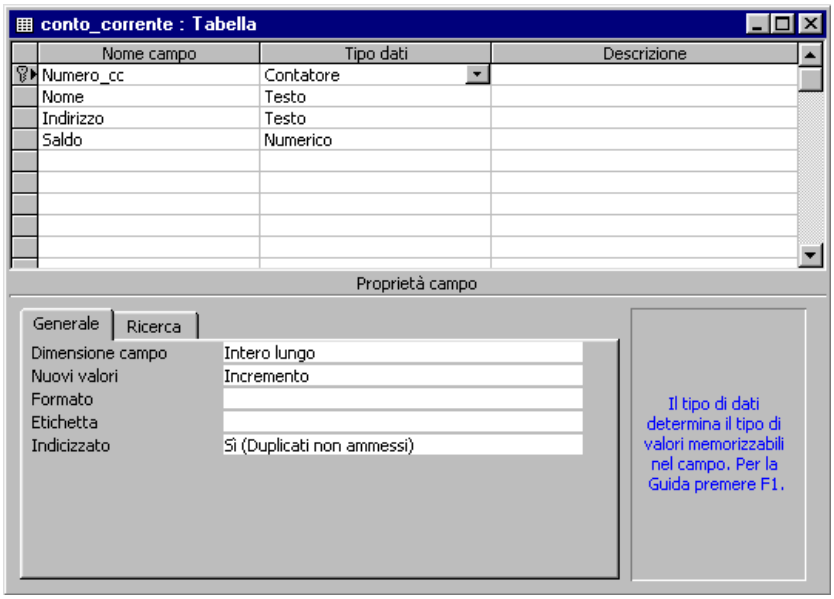


Creare la tabella "conto_corrente"

Scegliere nuovo nella finestra del database (cartella Tabelle)

Nella finestra nuova tabella scegliere Visualizzazione struttura

Nella finestra struttura della tabella inserire le proprietà dei campi (vedi dopo)



Dettaglio dei campi della tabella

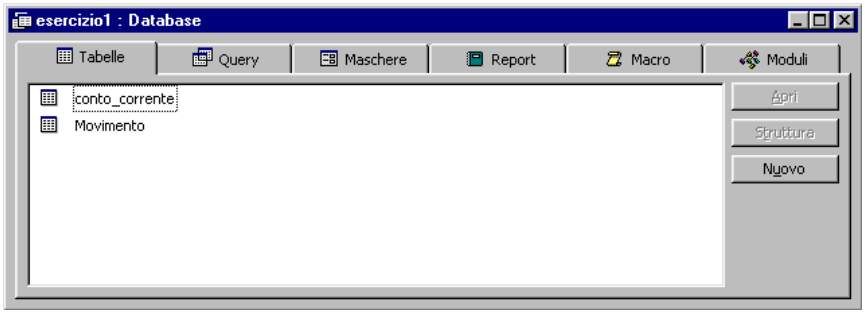
Nome	Tipo	Altre proprietà
Numero_cc	Contatore	Indicizzato: Sì (Duplicati non ammessi)
Nome	Testo	Dimensione campo: 50; richiesto: Si; Consenti lunghezza zero: No; Indicizzato: Si
Indirizzo	Testo	Dimensione campo: 50; richiesto: No; Consenti lunghezza zero: No; Indicizzato: No
Saldo	Numerico	Dimensione campo: intero lungo; Formato: valuta; Richiesto: No; Indicizzato: No

Creazione della chiave primaria

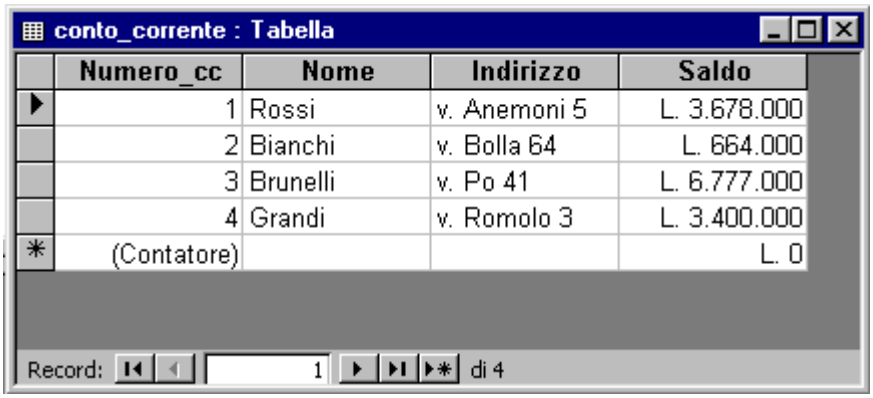
1. Evidenziare la riga "Numero_cc"
2. Dal menu "Modifica" scegliere "Chiave primaria"
3. Chiudere la finestra dal bottone di chiusura
4. Salvare la tabella con nome "conto_corrente"

Inserire i dati nella tabella "conto_corrente"

Nella finestra del database, selezionare la tabella "conto_corrente" e fare click sul bottone "Apri"



Apparirà la tabella (inizialmente vuota) dove si potranno inserire i dati



E' possibile passare dalla visualizzazione 'struttura' della tabella a quella 'foglio dati' mediante il bottone  e viceversa mediante 

La creazione degli indici

Un indice è una struttura dati che il DBMS associa alle tabelle per poter eseguire con maggiore efficienza la ricerca dei dati.

Creare un indice in SQL: CREATE INDEX … ON …

Sintassi

```
CREATE [UNIQUE] INDEX indice
ON tabella (campo [ASC|DESC][, campo
[ASC|DESC], ...])
```

La parola riservata UNIQUE impedisce la presenza di duplicati nel campo indicizzato

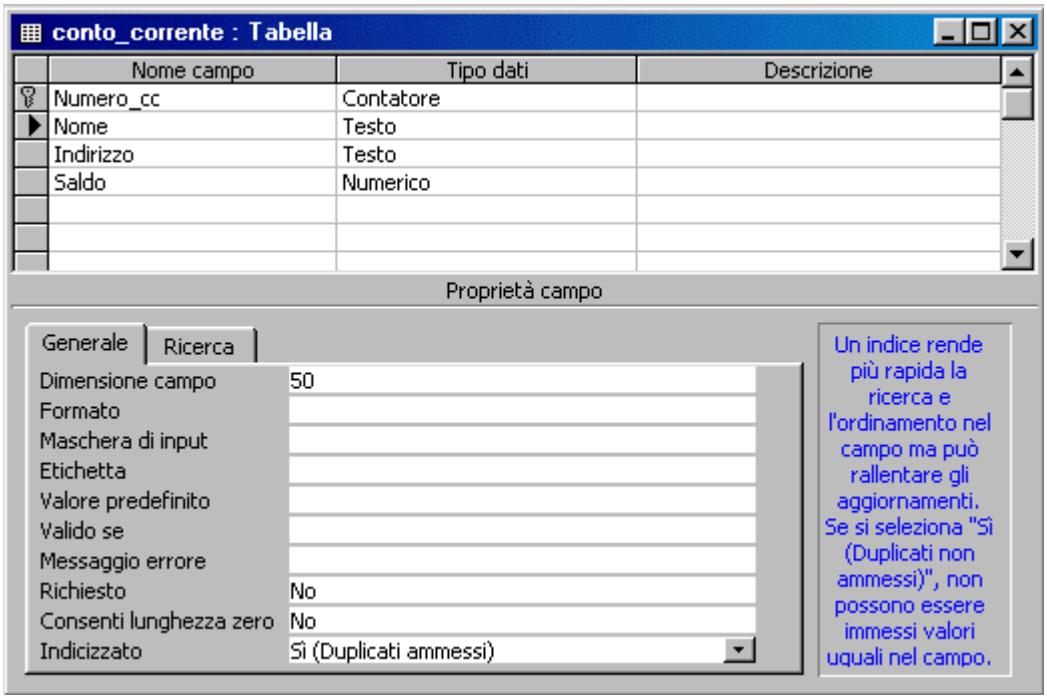
Esempio

```
CREATE INDEX CONTO_CORRENTE_KEY ON
CONTO_CORRENTE (NUMERO_CC)
```

Creare un indice con Access

Indicizzare la relazione conto_corrente in base all'attributo Nome

1) aprire la tabella in modalità struttura
2) selezionare il campo Nome
3) Dal riquadro delle proprietà del campo, alla voce Indicizzato scegliere “Si (Duplicati ammessi)”



Le operazioni di selezione

Le operazioni di selezione in SQL: SELECT

Seleziona dalle relazioni *nomi_relazioni* le tuple degli attributi *nomi_attributi* che soddisfano le condizioni *condizioni_ricerca*

Sintassi

```
SELECT nomi_attributi FROM nomi_relazioni
WHERE condizioni_ricerca
```

Esempio

```
SELECT SALDO FROM CONTO_CORRENTE WHERE
NUMERO_CC = 2
```

Le operazioni di selezione con Access

Esempio 1 (SELECT)

Descrizione

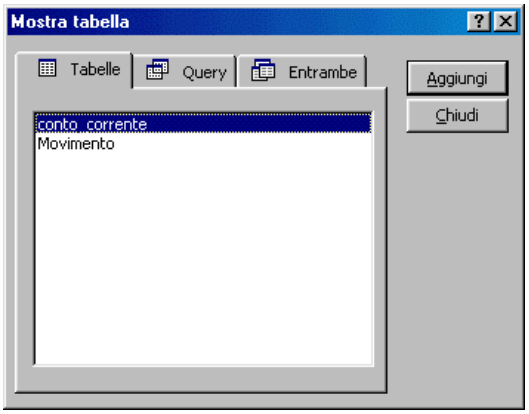
estrarre nomi e indirizzi di tutti i correntisti

La query da realizzare

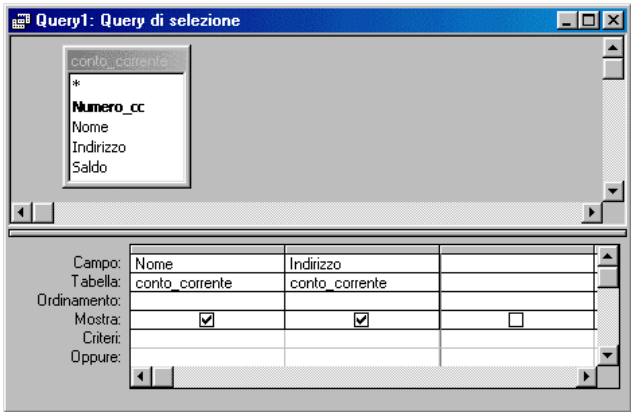
```
SELECT nome, indirizzo FROM conto_corrente
```

La procedura in Access

- Dalla finestra del database selezionare la cartella Query, fare click su Nuovo e su visualizzazione Struttura.



- Selezionare la tabella conto_corrente e poi il bottone Aggiungi.
- Nella finestra 'Query di selezione' scegliere nel rigo relativo al Campo: Nome e Indirizzo, come sotto illustrato.



Risultato della query

Per eseguire la query fare click sul bottone  .

La proiezione della relazione sui due attributi scelti sarà:

Esempio 2 (WHERE)

Descrizione

Estrarre il valore del saldo del conto corrente numero 2

La query da realizzare

```
SELECT saldo FROM conto_corrente WHERE
numero_cc = 2
```

La struttura della query in Access

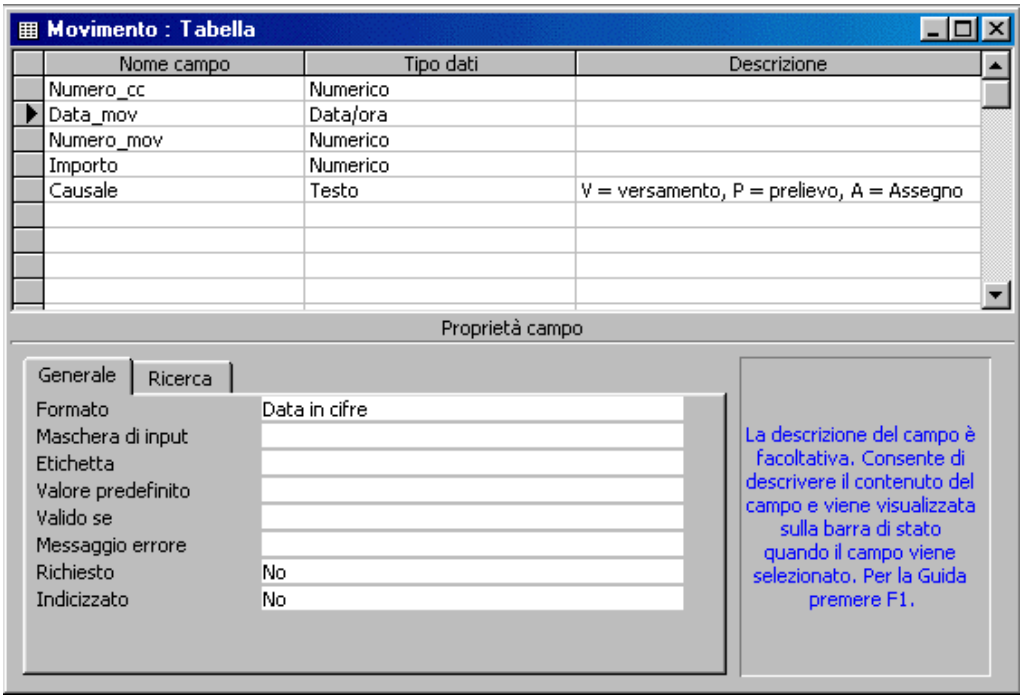
Creare una nuova query scegliendo i campi saldo e numero_cc. Alla voce Criteri per la colonna numero_cc inserire =2 e disabilitare l'opzione Mostra (per non far apparire il campo nel risultato della query)

Risultato della query

Eseguire la query. Il risultato sarà:

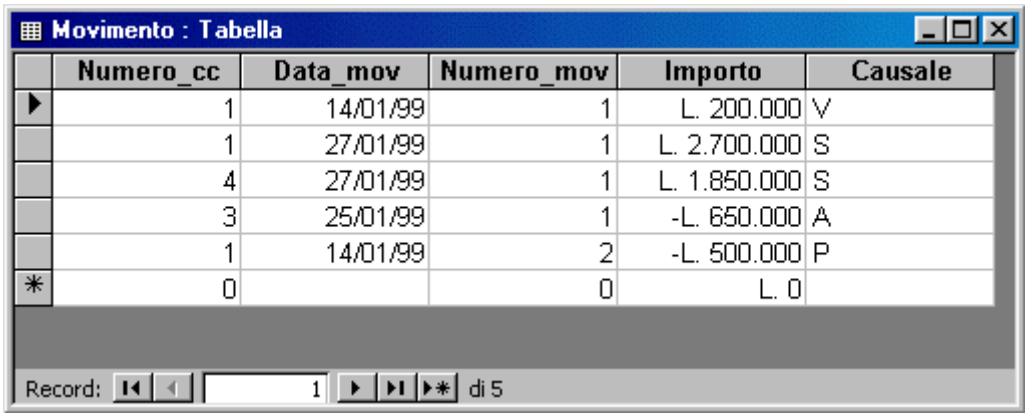
Esempio 3 (connettivi logici in WHERE)

Creazione della relazione Movimento



Descrizione dei campi (attributi) della tabella (relazione):

Nome	Tipo	Altre proprietà
Numero_cc	Numerico	Dimensione campo: intero lungo; Richiesto: No; Indicizzato: No
Data_mov	Data/ora	Dimensione campo: data in cifre; Richiesto: No; Indicizzato: No
Numero_mov	Numerico	Dimensione campo: intero lungo; Richiesto: No; Indicizzato: No
Importo	Numerico	Dimensione campo: intero lungo; Formato: valuta; Richiesto: No; Indicizzato: No
Causale	Testo	Dimensione campo: 1; Richiesto: Si; Indicizzato: No



Movimento : Tabella

Numero_cc	Data_mov	Numero_mov	Importo	Causale
1	14/01/99	1	L. 200.000	V
1	27/01/99	1	L. 2.700.000	S
4	27/01/99	1	L. 1.850.000	S
3	25/01/99	1	-L. 650.000	A
1	14/01/99	2	-L. 500.000	P
0		0	L. 0	

Record: 1 di 5

Descrizione della query

Estrarre nome e indirizzo dei correntisti che hanno un movimento in data 27-1-99

La query da realizzare

```
SELECT nome, indirizzo FROM conto_corrente,
movimento WHERE data_mov = 27-1-99 AND
conto_corrente.numero_cc =
movimento.numero_cc
```

La procedura da seguire in Access

Creare una nuova query, aggiungere le tabelle conto_corrente e movimento. Impostare il criterio di selezione sulla data.

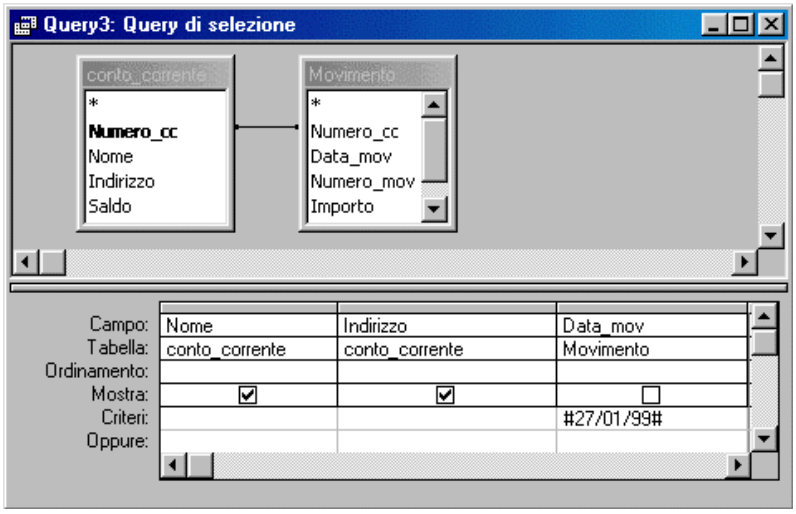
La condizione SQL “conto_corrente.numero_cc = movimento.numero_cc” viene automaticamente interpretata da Access come un Join.

JOIN

L'operazione di **JOIN** combina due relazioni in una sola concatenando le tuple che soddisfano la condizione di Join.

La relazione risultante ha grado pari alla somma dei gradi delle relazioni di partenza

La struttura della query in Access



Risultato della query

Il codice SQL generato

Scegliendo dal menu Visualizza l'opzione 'Visualizzazione SQL' si potrà esaminare il codice SQL prodotto dall'interazione grafica con il programma:

```
SELECT conto_corrente.Nome,
conto_corrente.Indirizzo
FROM conto_corrente INNER JOIN Movimento ON
conto_corrente.Numero_cc =
Movimento.Numero_cc
WHERE (((Movimento.Data_mov)=#1/27/99#));
```

La clausola INNER JOIN

Access ha interpretato la corrispondenza dei numeri di conto corrente come una relazione di JOIN.

I join interni combinano i record da due tabelle ogni volta che in un campo comune ad entrambe le tabelle vengono individuati valori corrispondenti.

È possibile utilizzare l'operazione INNER JOIN con le tabelle Reparti e Impiegati per selezionare tutti gli impiegati di ciascun reparto. Al contrario, per selezionare tutti i reparti, anche se alcuni non hanno impiegati, oppure per selezionare tutti gli impiegati, anche se alcuni non sono assegnati ad un reparto, è possibile utilizzare l'operazione LEFT JOIN o RIGHT JOIN per creare un join esterno.

Esempio 7 (connettivi logici in WHERE)

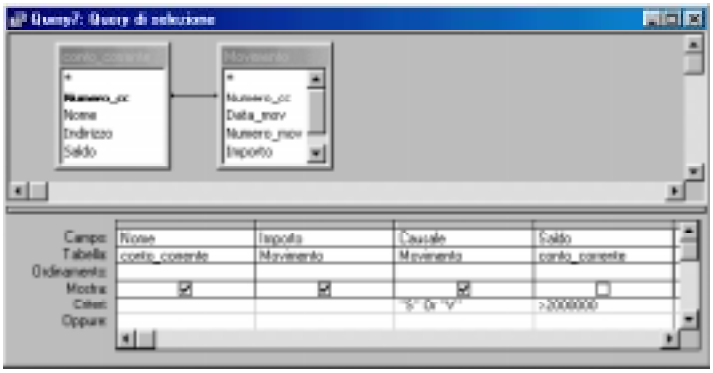
Descrizione della query

Estrarre nome del correntista, importo e causale dei movimenti di tutti i correntisti con un saldo maggiore di due milioni e con un movimento che sia un versamento o un accredito di stipendio

La query da realizzare è:

```
SELECT nome, importo, causale FROM
movimento, conto_corrente WHERE
saldo>2.000.000 AND (causale=V OR causale
=S) AND conto_corrente.numero_cc =
movimento.numero_cc
```

La struttura della query in Access



Il risultato della query

Query7: Query di selezione

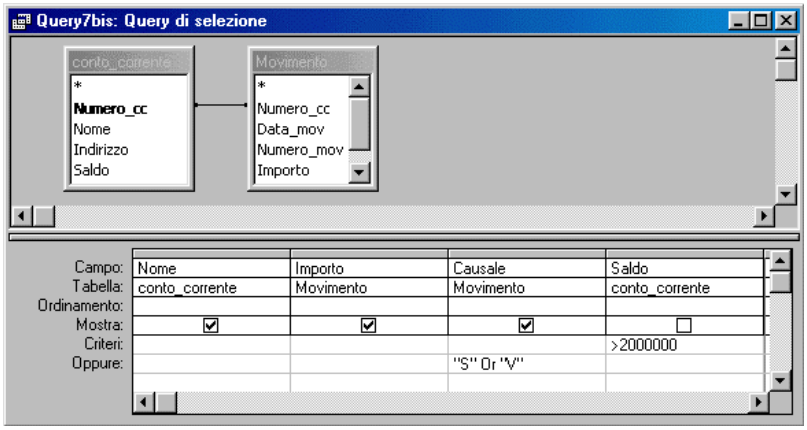
	Nome	Importo	Causale
▶	Rossi	L. 200.000	V
	Rossi	L. 2.700.000	S
	Grandi	L. 1.850.000	S
*			

Record: 1 di 3

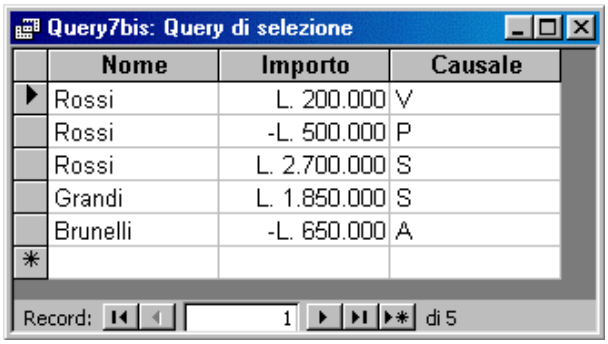
Il codice SQL generato

```
SELECT conto_corrente.Nome,
Movimento.Importo, Movimento.Causale
FROM conto_corrente INNER JOIN Movimento ON
conto_corrente.Numero_cc =
Movimento.Numero_cc
WHERE (((Movimento.Causale)="S" Or
(Movimento.Causale)="V") AND
((conto_corrente.Saldo)>2000000));
```

Si noti la differenza con la seguente interrogazione



Il risultato della query



Il codice SQL generato

```
SELECT conto_corrente.Nome,
Movimento.Importo, Movimento.Causale
FROM conto_corrente INNER JOIN Movimento ON
conto_corrente.Numero_cc =
Movimento.Numero_cc
WHERE (((conto_corrente.Saldo)>2000000)) OR
(((Movimento.Causale)="S" Or
(Movimento.Causale)="V"));
```

Esempio 8 (operazioni in SELECT)

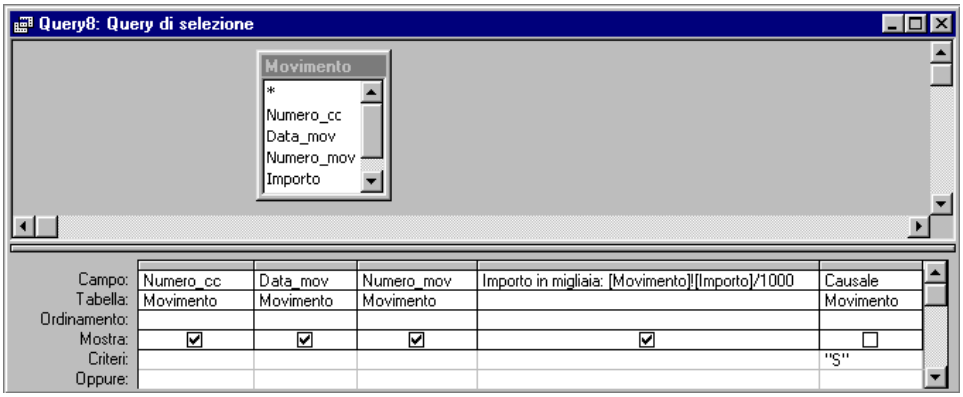
Descrizione della query

Estrarre numero del conto, data, numero di movimento ed importo in migliaia di lire dei movimenti dovuti ad un accredito di stipendio

La query da realizzare

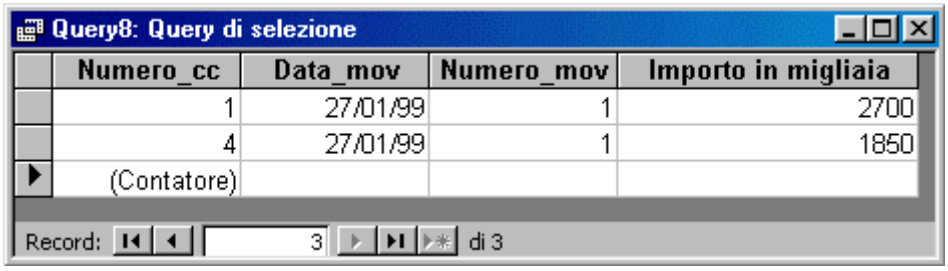
```
SELECT numero_cc, data_mov, numero_mov,
importo/1000
FROM movimento
WHERE causale =S
```

La struttura della query in Access



Il campo “Importo in migliaia” è stato introdotto con il generatore di espressioni accessibile mediante il bottone 

Il risultato della query



Il codice SQL generato

```
SELECT Movimento.Numero_cc,
Movimento.Data_mov, Movimento.Numero_mov,
[Movimento]![Importo]/1000 AS [Importo in
migliaia]
FROM Movimento
WHERE (((Movimento.Causale)="S"));
```

La clausola ORDER BY

Ordina i record risultanti da una query in base a uno o più campi specificati in ordine crescente o decrescente.

Sintassi

```
SELECT elencocampi
FROM tabella
WHERE criteriselezione
[ORDER BY campo1 [ASC | DESC ][, campo2 [ASC
| DESC ]][, ...]]]
```

Esempio 10 (ORDER)

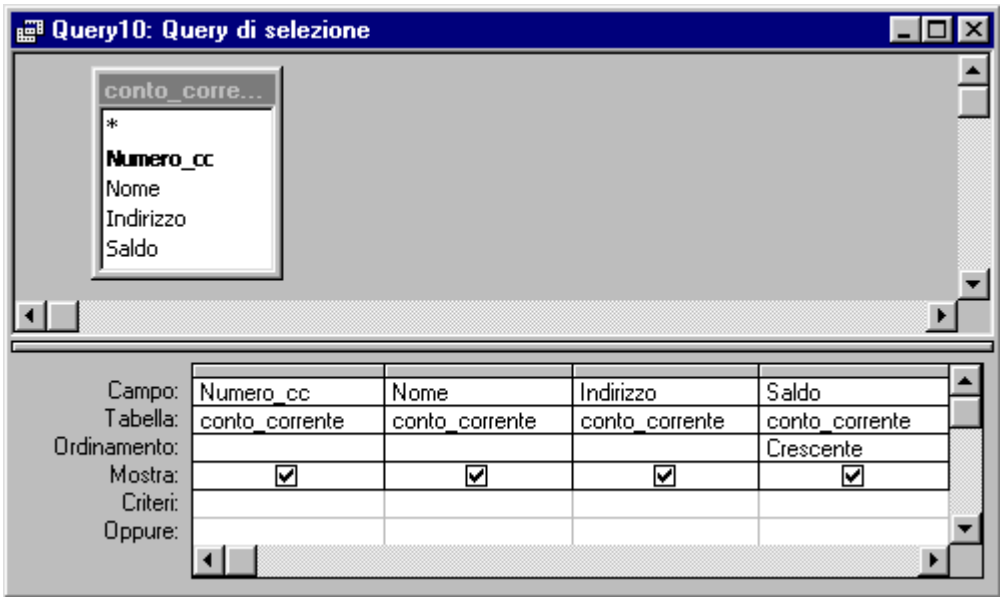
Descrizione della query

Estrarre l'intera relazione conto_corrente in cui i conti siano ordinati per valori crescenti del saldo

La query da realizzare

```
SELECT *
FROM conto_corrente
ORDER BY saldo
```

La struttura della query in Access



Il risultato della query

Il codice SQL generato

```
SELECT conto_corrente.Numero_cc,
conto_corrente.Nome,
conto_corrente.Indirizzo,
conto_corrente.Saldo
FROM conto_corrente
ORDER BY conto_corrente.Saldo;
```

Si noti che:

nella finestra della struttura della query si può impostare l'ordinamento in modo crescente o descrescente per ciascun campo

La clausola GROUP BY

Raggruppa tutte le tuple che assumono lo stesso valore su alcuni attributi (detti attributi di raggruppamento) costituendo una partizione della relazione in classi di equivalenza.

Sintassi

```
SELECT elencocampi
FROM tabella
WHERE criteri
GROUP BY elencocampiraggruppamento
```

Le funzioni aggregatrici

Su una partizione si possono applicare delle funzioni aggregatrici dei valori numerici:

MIN (calcola il minimo valore dell'attributo cui è applicata per ciascuna delle classi di equivalenza)

MAX (calcola il massimo valore dell'attributo cui è applicata per ciascuna delle classi di equivalenza)

SUM (calcola la somma dei valori dell'attributo cui è applicata per ciascuna delle classi di equivalenza)

AVG (calcola la media dei valori dell'attributo cui è applicata per ciascuna delle classi di equivalenza)

COUNT (conta il numero di tuple che costituiscono ciascuna delle classi di equivalenza)

La clausola HAVING

Permette di specificare condizioni sulle classi di equivalenza create dalla clausola GROUP BY.

Sintassi

```
SELECT elencocampi
FROM tabella
WHERE criteriselezione
GROUP BY elencocampiraggruppamento
HAVING criteriraggruppamento
```

Esempio 11 (GROUP)

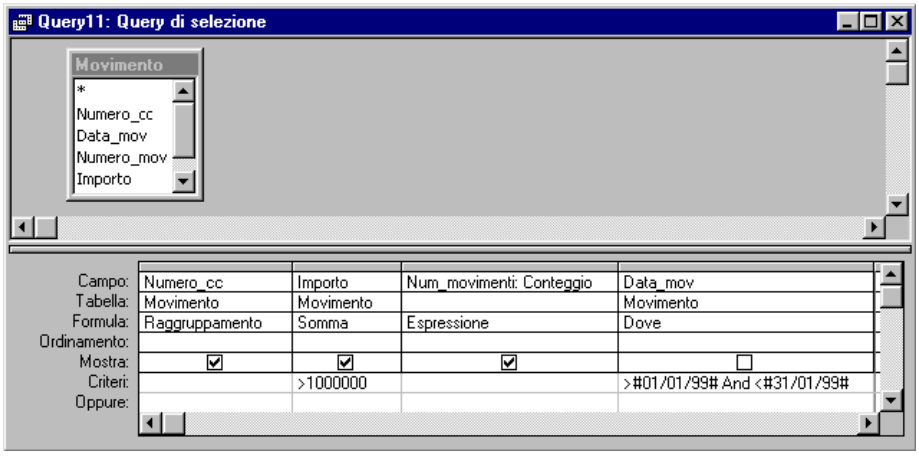
Descrizione della query

Estrarre numero corrente, soma degli importi e numero dei movimenti di tutti i conti correnti i cui movimenti abbiano superato la cifra complessiva di un milione nel corso del gennaio 1999

La query da realizzare

```
SELECT numero_cc, SUM(importo), count(*)
FROM movimento
WHERE data_mov > 1-1-99 AND data_mov < 31-1-99
GROUP BY numero_cc
HAVING SUM (Importo) > 1.000.000
```

La struttura della query in Access



Per far apparire la riga relativa al campo Formula fare click con il pulsante destro sulla griglia della finestra e scegliere Totali dal menu.

Il risultato della query

Il codice SQL generato è:

```
SELECT Movimento.Numero_cc,
Sum(Movimento.Importo) AS SommaDiImporto,
Count(*) AS Num_movimenti
FROM Movimento
WHERE (((Movimento.Data_mov)>#1/1/99# And
(Movimento.Data_mov)<#1/31/99#))
GROUP BY Movimento.Numero_cc
HAVING (((Sum(Movimento.Importo))>1000000));
```

Annidamento di blocchi di istruzioni SQL

Il connettore IN permette di annidare blocchi di istruzioni SQL.

Nell'esempio seguente, vi sono due blocchi di istruzioni connessi mediante IN.

```
SELECT nome, indirizzo
FROM conto_corrente
WHERE numero_cc IN
SELECT numero_cc
FROM movimento
WHERE data_mov=27-1-99
```

Esecuzione

1) viene eseguito il secondo blocco (che restituisce una serie di numeri di conto)

2) viene eseguito il primo blocco (che seleziona gli attributi nome e indirizzo dalle tuple il cui numero di cc è stato selezionato nel secondo blocco)

Esempio 12 (annidamento di blocchi)

Descrizione della query

Estrarre nome e indirizzo dei correntisti che hanno un movimento in data 27-1-99

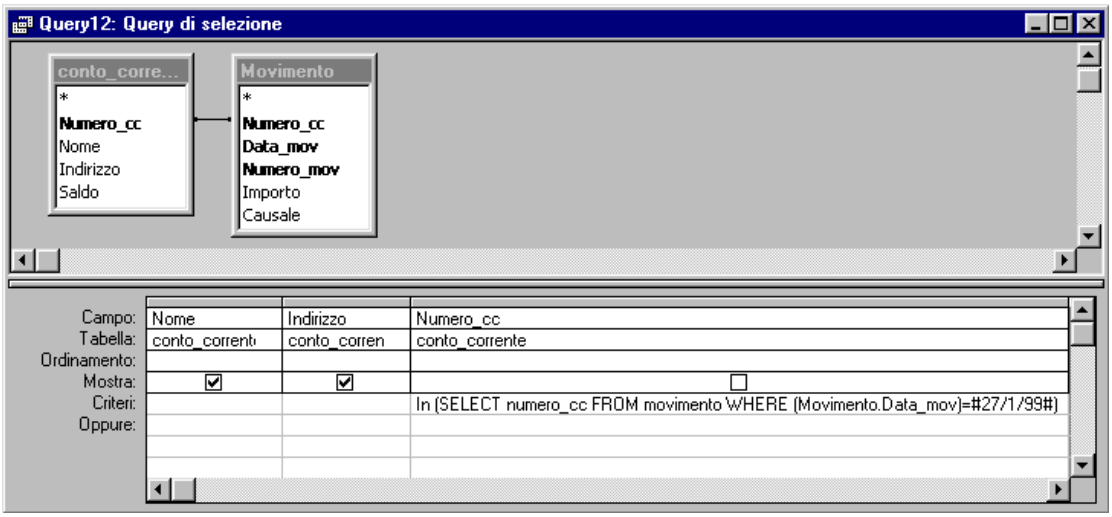
(interrogazione già proposta nell'esempio 3)

La query da realizzare

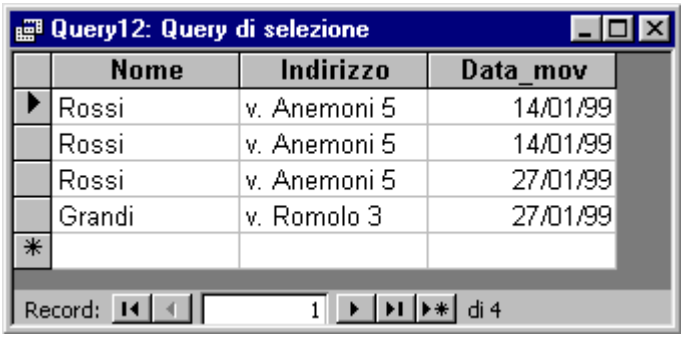
```
SELECT DISTINCT nome, indirizzo
FROM conto_corrente
WHERE numero_cc IN
SELECT numero_cc
FROM movimento
WHERE data_mov=27-1-99
```

Il predicato DISTINCT esclude i record che contengono dati duplicati nei campi selezionati.

La struttura della query in Access



Il risultato della query



Il codice SQL generato

```
SELECT DISTINCT conto_corrente.Nome,
conto_corrente.Indirizzo
FROM conto_corrente AS conto_corrente_1,
Movimento AS Movimento_1, conto_corrente
INNER JOIN Movimento ON
conto_corrente.Numero_cc =
Movimento.Numero_cc
WHERE (((conto_corrente.Numero_cc) In
(SELECT numero_cc FROM movimento WHERE
(Movimento.Data_mov)=#27/1/99#)));
```

INSERT

Inserisce in una relazione le tuple specificate

Sintassi

```
INSERT INTO Relazione VALUES (<tupla>,
<tupla>, …)
```

Esempio 14 (INSERT)

Descrizione della query

Inserire un nuovo correntista nella tabella conto_corrente

La query da realizzare

```
INSERT INTO conto_corrente
VALUES (<5, Cannetti, v. Tirolo 5, 100.000);
```

La struttura della query in Access

Si tratta di una query di accodamento. Dopo aver creato la nuova query, scegliere "Query di accodamento" dal menu "Query". Nella finestra seguente scegliere conto_corrente dal menu a tendina 'Nome tabella'.

Passare alla visualizzazione SQL della query.

Introdurre il seguente codice:

```
INSERT INTO conto_corrente
VALUES ('5', 'Cannetti', 'v. Tirolo 5',
'100.000');
```

Il risultato della query

Tabella conto_corrente prima dell'esecuzione della query:

conto_corrente : Tabella

Numero_cc	Nome	Indirizzo	Saldo
1	Rossi	v. Anemoni 5	L. 3.678.000
2	Bianchi	v. Bolla 64	L. 664.000
3	Brunelli	v. Po 41	L. 6.777.500
4	Grandi	v. Romolo 3	L. 3.400.000
(Contatore)			0

Record: 5 di 5

Tabella conto_corrente dopo l'esecuzione della query (si noti che la query va eseguita con il bottone ):

conto_corrente : Tabella

Numero_cc	Nome	Indirizzo	Saldo
1	Rossi	v. Anemoni 5	L. 3.678.000
2	Bianchi	v. Bolla 64	L. 664.000
3	Brunelli	v. Po 41	L. 6.777.500
4	Grandi	v. Romolo 3	L. 3.400.000
5	Cannetti	v. Tirolo 5	L. 100.000
(Contatore)			L. 0

Record: 6 di 6

Il codice SQL generato

```
INSERT INTO conto_corrente
VALUES ('5', 'Cannetti', 'v. Tirolo 5',
'100.000');
```

DELETE

Cancella da una relazione le tuple che soddisfano un predicato di selezione

Sintassi

```
DELETE FROM Relazione WHERE Predicato
```

Esempio 15 (DELETE)

Descrizione della query:

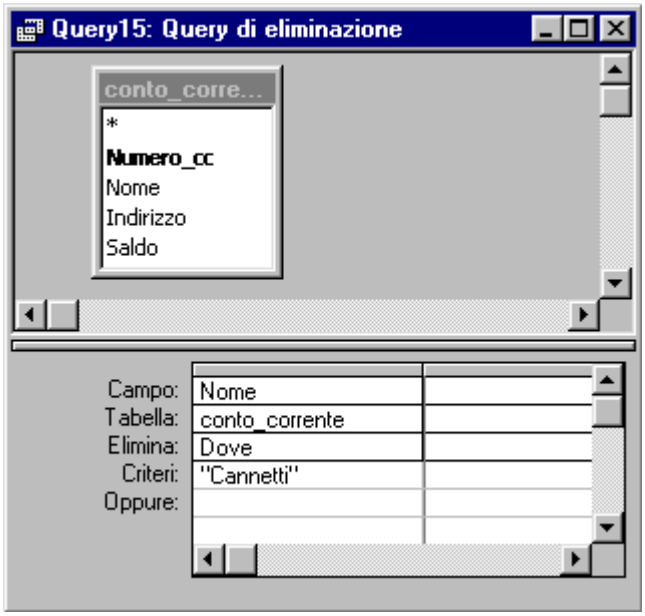
Eliminare il conto intestato a Cannetti

La query da realizzare

```
DELETE FROM conto_corrente
WHERE nome = Cannetti
```

La struttura della query in Access

Si tratta di una query di eliminazione. Dopo aver creato la nuova query, scegliere "Query di eliminazione" dal menu "Query".



Il risultato della query

Tabella conto_corrente prima dell'esecuzione della query:

conto_corrente : Tabella

Numero_cc	Nome	Indirizzo	Saldo
1	Rossi	v. Anemoni 5	L. 3.678.000
2	Bianchi	v. Bolla 64	L. 664.000
3	Brunelli	v. Po 41	L. 6.777.500
4	Grandi	v. Romolo 3	L. 3.400.000
5	Cannetti	v. Tirolo 5	L. 100.000
(Contatore)			L. 0

Record: 6 di 6

Tabella conto_corrente dopo l'esecuzione della query (si noti che la query va eseguita con il bottone ):

conto_corrente : Tabella

Numero_cc	Nome	Indirizzo	Saldo
1	Rossi	v. Anemoni 5	L. 3.678.000
2	Bianchi	v. Bolla 64	L. 664.000
3	Brunelli	v. Po 41	L. 6.777.500
4	Grandi	v. Romolo 3	L. 3.400.000
(Contatore)			

Record: 5 di 5

Il codice SQL generato:

```
DELETE conto_corrente.Nome
FROM conto_corrente
WHERE (((conto_corrente.Nome)="Cannetti"));
```

UPDATE

Modifica il contenuto di alcuni attributi di una relazione per tutte le tuple che soddisfano una certa condizione

Sintassi

```
UPDATE Relazione
SET Attributo = Espressione
WHERE Predicato
```

Esempio 16 (UPDATE)

Descrizione della query

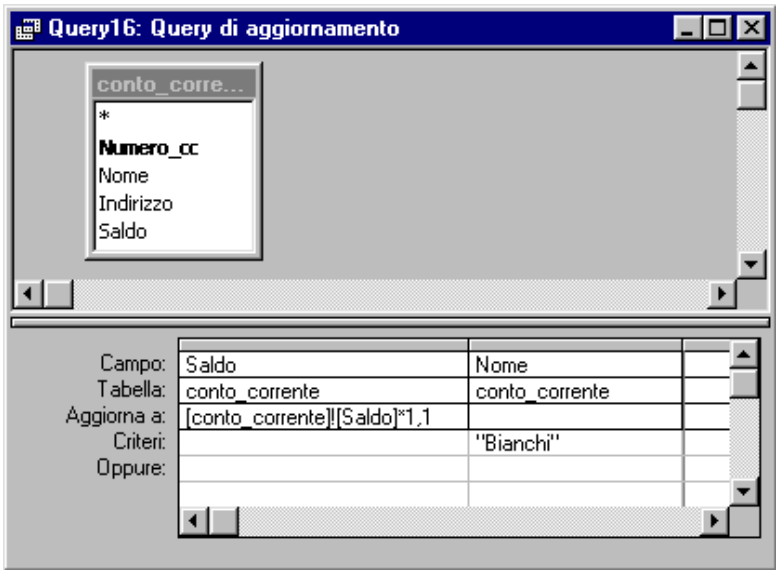
Aumentare del 10% il saldo del conto del sig. Bianchi

La query da realizzare

```
UPDATE conto_corrente
SET saldo = saldo *1,1
WHERE nome = Bianchi
```

La struttura della query in Access

Si tratta di una query di aggiornamento. Dopo aver creato una nuova query scegliere "Query di aggiornamento" dal menu "Query"



Il valore della riga 'Aggiorna a' per la colonna del campo Saldo è stato introdotto con il generatore di espressioni

Il risultato della query

Tabella conto_corrente prima dell'esecuzione della query:

Tabella conto_corrente dopo l'esecuzione della query (si noti che la query va eseguita con il bottone):

Il codice SQL generato

```
UPDATE conto_corrente SET
conto_corrente.Saldo =
[conto_corrente]![Saldo]*1.1
WHERE (((conto_corrente.Nome)="Bianchi"));
```