34 query realizzate con MS Access 97

MS Access è sicuramente il prodotto DBMS per PC più diffuso. Gli utilizzatori, considerando anche il fatto che il prodotto fa parte della Suite Office della Microsoft, sono ormai oltre un milione. Una minima parte di costoro, di categoria programmatori, usa Access per sviluppare applicazioni gestionali "chiuse", mentre la maggioranza, costituita da utilizzatori finali, che per definizione non sanno programmare, lo usa per gestire proprie piccole banche dati o per "postprocessare" dati che arrivano in altra maniera, anche in grosse quantità.
In questo secondo caso, tipico delle aziende in cui si persegue la tecnologia del DataWarehouse (magazzino dei dati aziendali accessibile a tutti gli utenti), l'utilizzatore non realizza in proprio le tabelle, né tanto meno si occupa della digitazione dei dati. Deve solo manipolarli allo scopo di trarne statistiche, proiezioni, andamenti, e quant'altro necessario per prendere decisioni.
Lo strumento Access che si usa per la manipolazione dei dati a questi fini è la Query.
Il nostro articolo non è che un campionario di query, di complessità "media", non troppo facili né troppo complesse, che possono servire a dare qualche spunto ai lettori che si trovano ad utilizzare Access.
Le abbiamo realizzate con Access 97, l'ultimo della serie, ma sono eseguibili con qualsiasi altra versione, la 95, la 2.0 e la 1.1.

di Francesco Petroni

Le due o tre cose che diamo per note

Diamo per note una serie di "cose" che elenchiamo qui di seguito. Inoltre diamo per già costruito il database di lavoro, che mostriamo nelle figure e descriviamo nelle didascalie. Si può ricostruire a mano, in modo che gli esercizi proposti possano essere eseguiti fedelmente, oppure si possono adattare gli esercizi stessi ai vostri casi reali, modificandoli opportunamente.

Diamo per note, anzi per ben note, le procedure di costruzione delle **strutture delle tabelle** e di impostazione delle **relazioni**. Quindi partiamo da un database già correttamente realizzato e già correttamente caricato, pena la non significatività degli esercizi.

In particolare occorre conoscere il significato della **relazione uno a molti**, l'unica che utilizzeremo nei nostri esercizi, e dei controlli di **integrità referenziale** che la stessa esegue sui nostri dati.

Diamo per noto l'utilizzo dell'ambiente **QbE di Access**, l'ambiente grafico in cui si costruiscono le query. Di molte delle nostre query mostreremo proprio la riproduzione dell'ambiente QbE. In particolare le cose che diamo per note sono:

- la scelta del tipo di query ed il significato di ciascun tipo (menu Query)
- la scelta delle tabelle e l'impostazione, se necessario, delle relazioni dinamiche
- la scelta dei campi
- l'inserimento dei criteri di selezioni (riga Criteri nel QbE)
- l'inserimento delle regole di ordinamento (riga Ordinamento nel QbE)
- la definizione delle proprietà dell'intera query
- l'inserimento di campi calcolati, anche facendo ricorso alle Functions disponibili
- la definizione delle proprietà del singolo campo (tasto destro sulla colonna e poi Proprietà)
- la definizione e l'uso dei parametri, da usare come criteri oppure, in generale, nelle formule

- l'impostazione delle regole di raggruppamento e delle conseguenti tipologie di calcolo.

Diamo inoltre per noti i principi fondamentali dell'SQL, ovvero il significato delle clausole SELECT, FROM, WHERE, ORDER BY, JOIN, anche se, in teoria, per realizzare i nostri esercizi, sarà possibile ricopiare semplicemente i comandi SQL nel nostro Database.

E' evidente che, nel caso monotabella, l'istruzione SQL sarà molto più breve e quindi più facile da scrivere di quella del caso pluritabella.

Per eseguire i vari esercizi il lettore potrà lavorare indifferentemente nell'ambiente QbE, che è più interattivo e guidato, oppure nella pagina bianca dell'SQL, nella quale dovrà scrivere a manina il comando SQL, al limite, ricopiandolo dall'elenco che segue.

Per quanto riguarda l'SQL, ed in particolare le istruzioni associate a ciascun esercizio, le riportiamo tutte in maiuscolo, inoltre non scriviamo, a meno che non sia necessario, il nome della tabella prima del nome del campo.

Ad esempio l'istruzione SQL del primo esercizio Access la presenterebbe così:

```
SELECT DISTINCTROW PERSONE.COGNOME, PERSONE.CITTA
FROM PERSONE
WHERE NOT PERSONE.CITTA="ROMA"
ORDER BY PERSONE.COGNOME;
```

se la scriviamo a mano, nella finestra SQL dell'editor di Access, la possiamo semplificare in:

```
SELECT COGNOME, CITTA FROM PERSONE
WHERE NOT CITTA="ROMA"
ORDER BY COGNOME;
```

Diamo per note le funzioni e le modalità d'uso delle funzioni. Ad esempio ci capiterà di utilizzare la funzione IIF. Non è obbligatorio conoscerla, ci basterà che la impariate la prima volta che vi costringeremo ad utilizzarla.

Diamo per noto il fatto che una query, anche una query molto complessa, anche una query parametrizzata, può essere utilizzata come fonte di dati per una **Maschera** e per un **Report**.

In altre parole un problema complesso riguardante una maschera o un report può essere risolto a monte, direttamente dalla query che passa i dati alla maschera o al report stessi.

E' frequente il caso in cui una query può risolvere le necessità di una maschera o di un report mentre è raro il contrario, anzi nel caso del report il contrario è impossibile.

Alcuni degli esercizi approfondiscono uno dei temi ora accennati, altri li mischia-

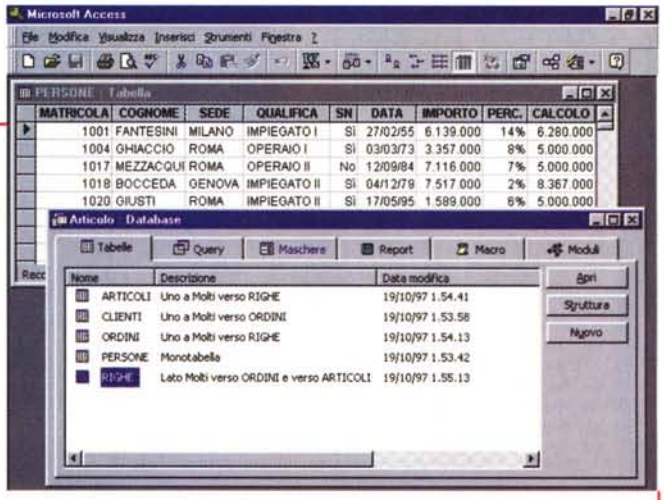


Figura 1 - Il nostro database - La tabella Persone.
Scopo di questo articolo è quello di mostrare una serie di query realizzate con Access 97. Si tratta di query di media difficoltà nel senso che diamo per scontato il fatto che il lettore sappia già costruire delle query facili, scegliendo le tabelle, definendo i legami relazionali tra di loro, scegliendo i campi, impostando i criteri di selezione, inserendo le regole di ordinamento e di raggruppamento, ecc. Non si tratta neanche di query troppo difficili da non poter essere eseguite in pochi minuti sul proprio PC. Le prime 17, delle 34 query, elaborano una sola tabella, che si chiama Persone e che appare nella figura in secondo piano, le altre 17 invece utilizzano le altre quattro tabelle (ne vediamo i nomi nella finestra database) e quindi affrontano soprattutto problematiche relazionali.

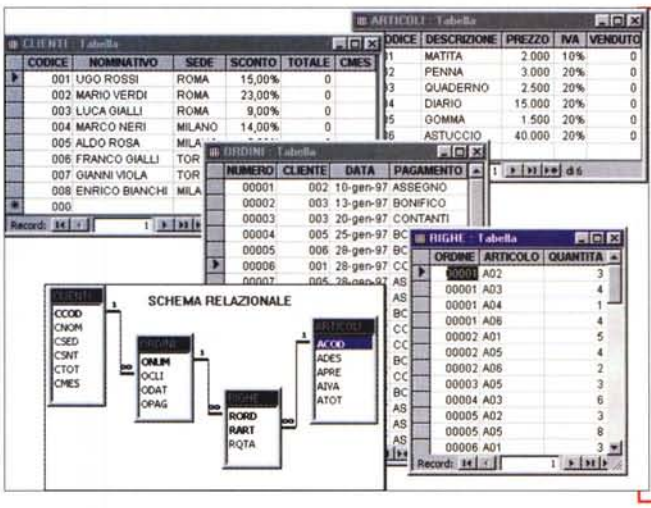


Figura 2 - Il nostro database - Le quattro tabelle: Clienti, Ordini, Righe e Articoli.
Affrontiamo una problematica standard ben comprensibile da tutti. Ci sono dei Clienti (sono 8) che inoltrano degli Ordini (che sono 20). Tra clienti ed ordini c'è una relazione uno a molti (un cliente può fare molti ordini, un ordine è fatto da un solo cliente). In ogni ordine sono venduti uno o più Articoli, un articolo può essere venduto in più ordini. Questo rapporto molti a molti si risolve con la tabella "intermedia" Righe, che indica anche la quantità di quell'articolo venduto in quell'ordine. Gli articoli sono 6 e le righe 45. Per eseguire i nostri esercizi partiamo da tabelle costruite correttamente e già piene di dati.

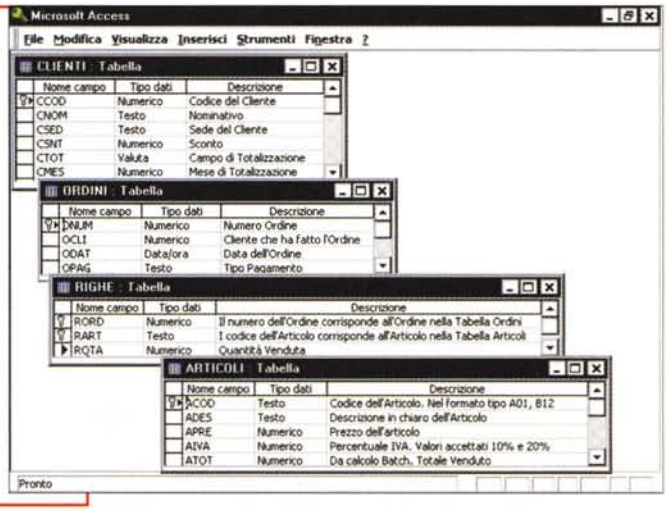


Figura 3 - Le strutture delle nostre tabelle.
Per eseguire correttamente gli esercizi occorre che le cinque tabelle (quella "single" e quella del caso a quattro tabelle) siano correttamente costruite. In questa figura vediamo quindi un collage con le strutture delle quattro tabelle. Ovviamente si tratta di un caso studio estremamente semplificato non utilizzabile in applicazioni vere. I passi operativi da compiere prima di eseguire le varie query sono tre: costruzione delle strutture, impostazione delle relazioni, inserimento dei dati. I dati li potete inventare, ma devono essere significativi, nel senso che le query che faremo devono dare risultati significativi.

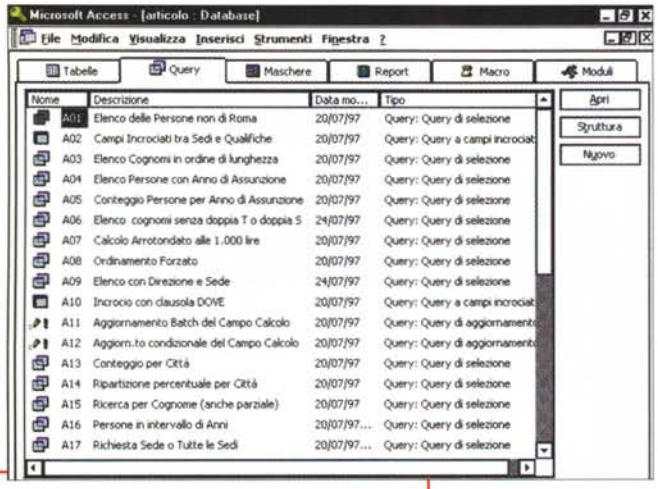


Figura 4 - Elenco delle prime 17 query - Il caso monotabella.
Le prime 17 query si poggiano, come detto, solo sulla tabella Persone. Le abbiamo chiamate A01, A02, ecc., e le abbiamo ben descritte nelle proprietà. La finestra database che elenca le query mostra anche la data di creazione e quella di modifica ed il tipo di query. Vedremo query di tre tipi (Selezione, Campi Incrociati e Aggiornamento) e quindi non di tutti i tipi permessi da Access. Per l'articolo abbiamo scelto un'impaginazione in forma di tabella Word per mostrare contemporaneamente sia il problema che la sua soluzione e per riportare una breve nota esplicativa. Mostreremo sempre il listato SQL della query soluzione, in qualche caso la videata dell'Editor delle Query (l'ambiente QbE di Access), in altri casi la vista dei dati, mentre, nei casi più significativi, mostreremo ambedue le videate.

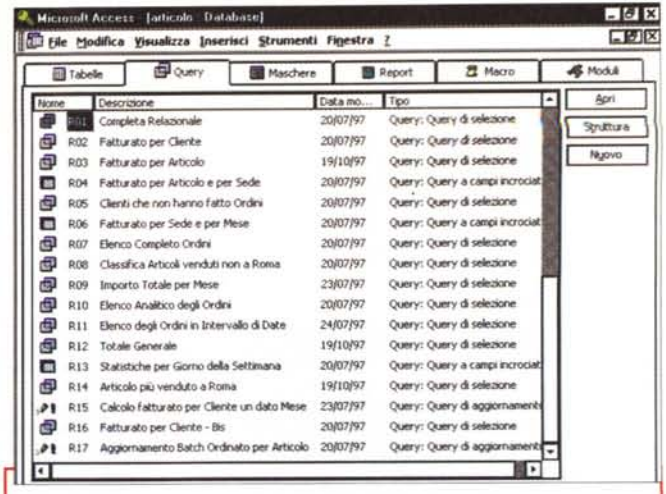


Figura 5 - Elenco delle seconde 17 query - Il caso quadritabelle.
Tutte le problematiche viste nelle prime 17 query possono essere riproposte anche nel caso pluritabelle. E' ovvio che le successive 17 query affronteranno soprattutto problematiche relazionali, che nascono proprio dal fatto che ci troviamo a lavorare con quattro tabelle. Tra queste devono essere stati definiti i rapporti relazionali e deve essere stato imposto il controllo dell'integrità referenziale, che garantisce l'allineamento reciproco tra le tabelle. Ad esempio non ci può essere un ordine assegnato ad un cliente che non c'è, non si può vendere un articolo che non esiste ed altre ovvietà del genere, che vengono controllate direttamente da Access.

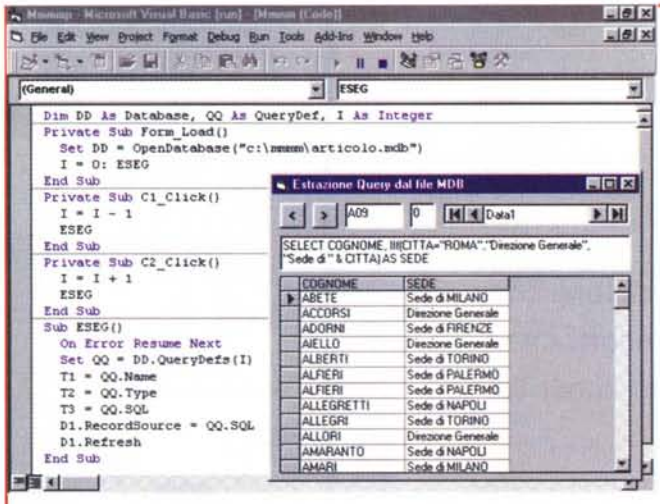


Figura 6 - Programma Visual Basic che legge ed esegue le query del file MDB.
Una query si concretizza in una istruzione SQL (in pratica Access memorizza l'istruzione SQL e non il risultato della query). Nel nostro file ARTICOLO.MDB ci sono quindi ben 34 istruzioni SQL, che possono essere lette "dal di fuori", ad esempio utilizzando la tecnologia DAO. Lo facciamo con il Visual Basic realizzando un semplice programma che scorre gli oggetti QueryDefs presenti nel database, ne legge la parte SQL e la riesegue, assegnandola come proprietà RecordSource ad un oggetto DataControl, che usa, anche esso, l'SQL per la definizione dell'insieme dei dati che gestisce.

no. Ad esempio è possibile costruire un **Campo Calcolato** che utilizza un **parametro** oppure che serve per creare un **raggruppamento**.

Prepariamo il database

Per eseguire fedelmente i nostri esercizi occorre che venga ricostruito perfettamente lo stesso database che abbiamo usato noi.

Prima la tabella "single", che si chiama **PERSONE** e che (nel nostro caso) contiene 500 record e che serve per i primi 17 esercizi.

Poi le quattro tabelle, per i secondi 17 esercizi, le cui strutture sono riportate nella figura 3, poi le tre relazioni che le uniscono, che vediamo graficizzate in figura 2.

Quindi riempiamo le tabelle di dati corretti e significativi. Quella "single" con 500 record significativi (ma potete inserirne anche di meno oppure adattare una tabella che già avete).

Per quanto riguarda il caso con quattro tabelle, noi abbiamo inserito 8 clienti e 6 articoli nelle due tabelle anagrafiche. Poi abbiamo inserito 20 ordini, facendo in modo che un paio di clienti non avessero fatto nessun ordine. Poi, ad unire gli ordini con gli articoli, abbiamo inserito 45 righe. Siamo stati attenti a che ogni ordine avesse almeno una riga, considerando il fatto che, da un punto di vista puramente relazionale, sarebbe invece possibile che un ordine non avesse righe, così come è possibile che un cliente non abbia ordini.

Insomma è necessario che il database sia a "prova di bomba" per dare significatività agli esercizi e per evitare errori dovuti non alle query ma alla scorrettezza dei dati di prova.

Le 34 query (e le altre... 66 per arrivare a 100)

Poiché sono previsti 34 esercizi con una cinquantina di videate ci siamo dovuti preoccupare anche di impaginarli sul nostro PC, quindi le pagine che seguono non risulteranno stampate con lo stile standard di MC. L'elenco delle query, le 17 più 17, lo trovate nelle figure 4 e 5.

Nella figura 6 invece vediamo come possano essere lette le 34 query anche dal di fuori di Access, nel nostro caso direttamente da MS Visual Basic, che è, ovviamente, anche in grado di rieseguirle.

C'è da dire che però alcune delle query "girano" solo in Access, sono quelle che usano delle function di Access che non sono standard nell'SQL.

Un'ultima cosa prima di... lasciarvi lavorare. Le 34 query non solo non esauriscono le possibilità di Access ma non coprono neanche tutte le tipologie di query possibili... in natura.

A01 - Elenco delle Persone non di Roma

```
SELECT COGNOME, CITTA FROM PERSONE WHERE CITTA<>"ROMA"
ORDER BY COGNOME;
```

Cominciamo con una semplice **Query di Selezione** con due soli campi. Il campo **COGNOME**, sul quale si esegue **l'Ordinamento**, e il campo **CITTA** su cui si imposta un **Criterio**. Si può utilizzare la clausola **Not** o l'operatore **<>** con i quali si possono negare altri criteri. Spesso è più facile esprimere un Criterio in positivo e poi negarlo.

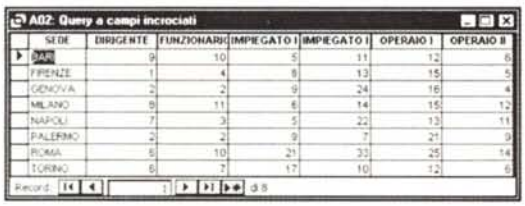
A02 - Campi Incrociati tra Sedi e Qualifica

```
TRANSFORM COUNT(COGNOME) AS TOTALE SELECT CITTA FROM
PERSONE GROUP BY CITTA PIVOT QUALIFICA;
```

Proseguiamo con una query di altro tipo, **Campi Incrociati**, nella quale vanno indicati, in generale, tre elementi:

Intestazione delle Righe (anche più di uno)	**CITTA**
Intestazione delle Colonne (uno solo)	**QUALIFICA**
Valore (uno solo)	**COGNOME**

Sui primi due campi va eseguito un **Raggruppamento**, mentre sul terzo va eseguito un **Calcolo**, in genere un **Conteggio**, in caso di campo **Alfanumerico**, una **Somma**, in caso di campo **Numerico**.



A03 - Elenco Cognomi in ordine di Lunghezza

```
SELECT COGNOME FROM PERSONE ORDER BY LEN(COGNOME) DESC;
```

Viene usato un **Campo Calcolato**, che non viene mostrato (basta togliere il visto sulla riga Mostra), per mettere in ordine il risultato della query. Il calcolo è l'applicazione della funzione **Len**, che calcola la lunghezza in caratteri di una stringa, al campo **COGNOME**. Si tratta di una funzione che estrae da un campo di tipo stringa un valore di tipo numero.

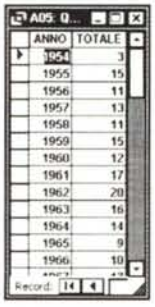
A04 - Elenco Persone con Anno di Assunzione

```
SELECT MATRICOLA, COGNOME, YEAR(DATA) AS ANNO FROM
PERSONE;
```

Non ci interessa visualizzare il campo **Data**, ma solo il valore del suo anno. Anche questa query quindi prevede l'utilizzo di un campo calcolato.
L'anno viene ottenuto applicando la funzione **Year** al campo **DATA**.
Si tratta di una funzione che estrae da un campo di tipo data un valore di tipo numero.

A05 - Conteggio dei Cognomi per Anno di Assunzione

```
SELECT Year(DATA) AS ANNO, Count(COGNOME) AS TOTALE FROM
PERSONE GROUP BY Year(DATA);
```



Creiamo un **Raggruppamento**. Occorre premere, nell'ambiente **QbE**, il pulsante con la lettera **Sigma**, che inserisce la riga **Formula** nella griglia del QbE. Di ogni colonna va definito il comportamento nei confronti del raggruppamento.
Si tratta di un'evoluzione della query precedente: una volta calcolato l'**Anno**, lo si può usare come campo di Raggruppamento per contare le frequenze di persone assunte per ogni anno.

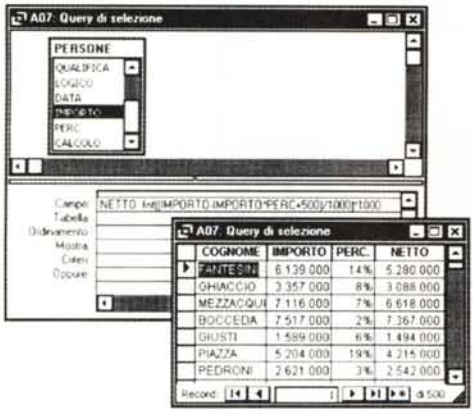
A06 - Elenco cognomi senza doppia T o doppia S

```
SELECT COGNOME FROM PERSONE WHERE COGNOME Not (Like
"*TT*" Or Like «*SS*»)
```

Altro esempio di negazione di un Criterio complesso, più facilmente impostabile in "positivo" che non in negativo.
Vogliamo escludere dalla lista i cognomi in cui ci sia una doppia T o una doppia S.
Da notare la necessità di utilizzare le parentesi per forzare la prevalenza dell'Or sul Not.

A07 - Calcolo Arrotondato alle 1.000 lire

```
SELECT COGNOME, IMPORTO, PERC, INT((IMPORTO-
IMPORTO*PERC+500)/1000)*1000 AS NETTO FROM PERSONE;
```



Campo calcolato che utilizza i due campi **IMPORTO** e **PERC**.

Il calcolo viene reso più complesso dal fatto che viene utilizzato un classico algoritmo per ottenere l'arrotondamento del risultato alle 1.000 lire, basato sull'uso della funzione **Int**.

A08 - Ordinamento Forzato

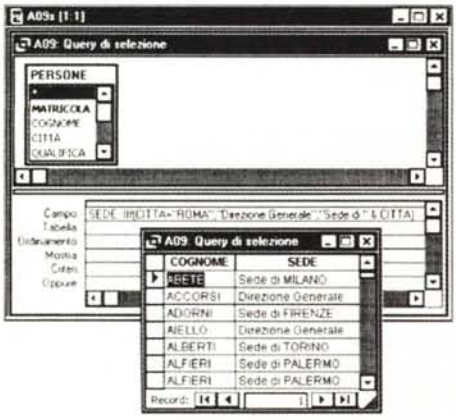
```
SELECT CITTA, COGNOME FROM PERSONE
ORDER BY IIf(CITTA="ROMA","A","B"), CITTA, COGNOME;
```

Vogliamo ordinare i cognomi per **CITTA**, inserendo però i cognomi di ROMA al primo posto nell'elenco. Utilizziamo un **Campo Calcolato** che fornisce A, in caso di cognome di Roma, altrimenti B. Il calcolo si basa sulla funzione IIF, la cui sintassi:

```
IIF(test;risposta se vero;risposta se falso)
```

A09 - Elenco con Direzione e Sede

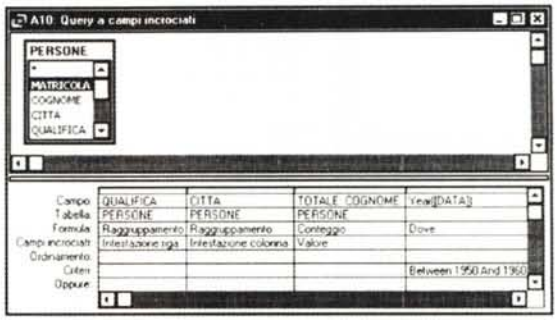
```
SELECT COGNOME,
IIf(CITTA="ROMA","Direzione Generale","Sede di "&CITTA)
AS SEDE FROM PERSONE ORDER BY COGNOME;
```



Variante dell'esercizio precedente. In questo caso il calcolo deve produrre una nuova colonna che visualizza due stringhe differenti a seconda che il cognome sia di Roma o meno.

A10 - Incrocio con clausola DOVE

```
TRANSFORM COUNT(COGNOME) AS TOTALE SELECT QUALIFICA FROM
PERSONE WHERE YEAR(DATA) Between 1950 And 1960 GROUP BY
QUALIFICA PIVOT CITTA;
```



Query di tipo **Campi Incrociati** tra i campi **QUALIFICA** e **CITTA**. Il valore è il conteggio del **COGNOME**. Viene introdotto un criterio sulla **DATA** (clausola **Between** per selezionare un intervallo di date) che dà fastidio al raggruppamento. L'unico modo per venirne fuori è quello di utilizzare la clausola **Dove**.

A11 - Aggiornamento Batch del Campo Calcolo

```
UPDATE PERSONE
SET CALCOLO = Int((IMPORTO-IMPORTO*perc+500)/1000)*1000;
```

Query di Aggiornamento. Si tratta di una **Action Query**, che produce quindi modifiche nei dati.
Vogliamo modificare il campo **CALCOLO** inserendo un valore pari all'applicazione del campo **PERC** al campo **IMPORTO**, il tutto arrotondato alle 1.000 lire.

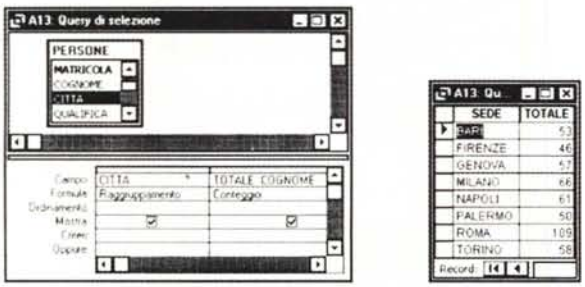
A12 - Aggiornamento condizionale del Campo Calcolo

```
UPDATE PERSONE
SET CALCOLO = IIf(CITTA="ROMA",5000000,CALCOLO+1000000);
```

Seconda query di tipo **Aggiornamento**. Si tratta ancora di una **Action Query**, che produce quindi modifiche nei dati. Aggiorniamo il campo **CALCOLO** con il valore **5.000.000** in caso di persone di ROMA, altrimenti incrementiamo il valore precedente di **1.000.000.**

A13 - Conteggio per Città

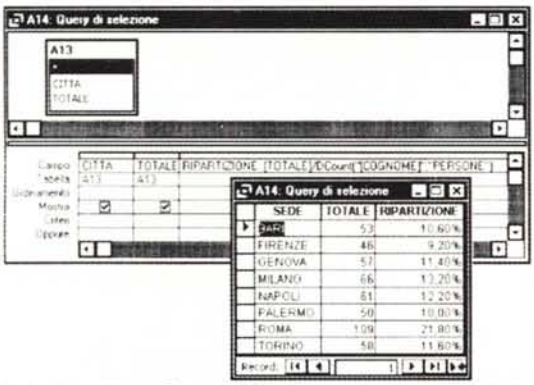
```
SELECT CITTA, Count(COGNOME) AS TOTALE FROM PERSONE
GROUP BY CITTA;
```



Semplice raggruppamento sul campo **CITTA** per ottenere il conteggio del campo **COGNOME**. Questa query serve come base di partenza per la successiva query.
In Access è possibile realizzare delle **CATENE** di query.
Va lanciata direttamente l'ultima della catena che provvede ad eseguire tutte le query precedenti.

A14 - Ripartizione Percentuale per Città

```
SELECT A13.CITTA, A13.TOTALE, [TOTALE]/Dcount
("[COGNOME]","PERSONE") AS RIPARTIZIONE FROM A13;
```



Il risultato di una Query di Selezione può essere assimilato a tutti gli effetti ad una Tabella (una sorta di **Tabella Virtuale**), che può essere riutilizzata per eseguire una successiva query che in tal modo diventa più semplice da preparare.
Per ottenere il conteggio di **COGNOME**, necessario per eseguire il calcolo della percentuale, usiamo la funzione **Dcount**:

```
Dcount("campo";"tabella";"condizione")
```

Nella quale non abbiamo inserito la condizione per ottenere il totale di tutti i valori.

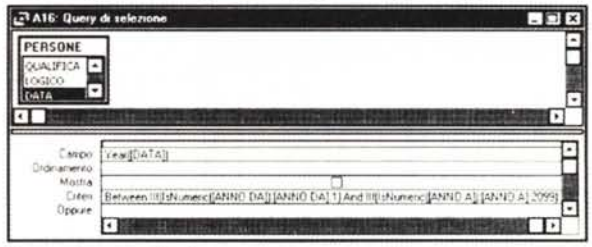
A15 - Ricerca per Cognome (anche parziale)

```
SELECT COGNOME, * FROM PERSONE WHERE COGNOME Like "*" &
[NOMINATIVO] & "*";
```

Query che utilizza un **Parametro**.
Per selezionare i dati partendo anche da una porzione di **COGNOME** abbiamo concatenato due caratteri jolly (*) prima e dopo il parametro.
Ad esempio, digitando ROSS, la query visualizza ROSSI ma anche ROSSINI.

A16 - Persone in Intervallo di Anni

```
SELECT CITTA, Year(DATA) AS ANNO FROM PERSONE WHERE
YEAR(DATA) Between IIf(IsNumeric([ANNO DA]),[ANNO DA],1)
And IIf(IsNumeric([ANNO A]),[ANNO A],2099);
```

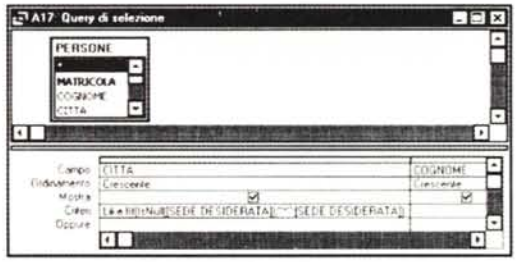


Parametro complicato per operare una selezione sugli anni.
Vengono richiesti due parametri **ANNO DA** e **ANNO A**.
In caso di risposta vuota (oppure non numerica) viene preso un valore basso come ANNO DA ed uno alto come ANNO A.
E' quindi possibile selezionare:

da ANNO DA ad ANNO A	tutti
da ANNO DA in poi	fino ad ANNO A

A17 - Richiesta Sede o Tutte le Sedi

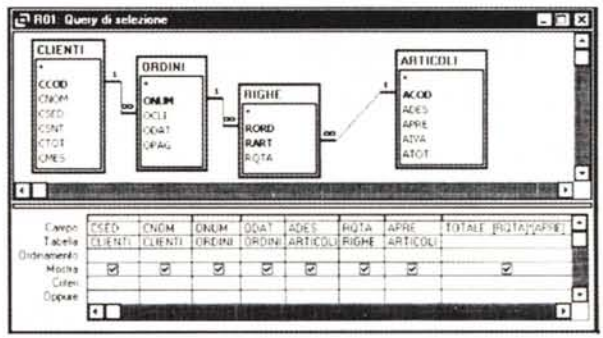
```
SELECT CITTA, COGNOME FROM PERSONE WHERE CITTA Like
IIf(IsNull([SEDE DESIDERATA]),"*",[SEDE DESIDERATA])
ORDER BY CITTA, COGNOME;
```



In questo caso di **Query Parametrizzata** non si usa il jolly.
La query, se non viene digitato nessun parametro (funzione **IsNull()**), imposta come criterio di selezione l'asterisco.

R01 - Completa Relazionale

```
SELECT CSED, CNOM, ONUM, ODAT ADES, RQTA, APRE,
[RQTA]*[APRE] AS TOTALE FROM ARTICOLI INNER JOIN
((CLIENTI INNER JOIN ORDINI ON CLIENTI.CCOD =
ORDINI.OCLI) INNER JOIN RIGHE ON ORDINI.ONUM =
RIGHE.RORD) ON ARTICOLI.ACOD = RIGHE.RART;
```

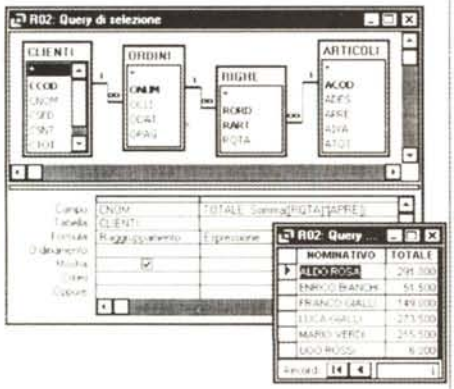


Prima query sul secondo caso studio, basato su **Quattro Tabelle**.
Preleviamo campi dalle varie tabelle, che sono ben collegate tra di loro dalle Relazioni. Inseriamo anche un Campo Calcolato per ricavare il totale della riga di vendita con la formula

```
APRE*RQTA (prezzo per quantità)
```

R02 - Fatturato per Cliente

```
SELECT CNOM, Sum([RQTA]*[APRE]) AS TOTALE FROM (CLIENTI
INNER JOIN ORDINI ON CLIENTI.CCOD =ORDINI.OCLI) INNER JOIN
(ARTICOLI INNER JOIN RIGHE ON ARTICOLI.ACOD = RIGHE.RART)
ON ORDINI.ONUM = RIGHE.RORD GROUP BY CLIENTI.CNOM;
```

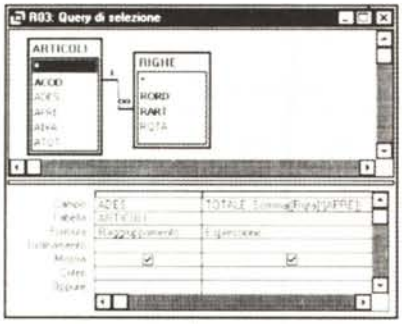


Il Fatturato per Cliente si ottiene calcolando il prodotto della quantità **RQTA** per il prezzo **APRE** e raggruppando per cliente **CNOM**. Anche se non si inseriscono campi relativi agli Ordini la tabella Ordini va comunque caricata perché serve alla Relazione.

Una volta create correttamente le **Relazioni** non c'è nessuna differenza tra lo scegliere campi della stessa tabella e campi di varie tabelle. Nell'esercizio sono le relazioni che individuano gli **Ordini** fatti da ciascun **Cliente**, conseguentemente le **Righe** degli ordini, conseguentemente gli **Articoli**, venduti in quelle righe.

R03 - Fatturato per Articolo

```
SELECT ADES, Sum([Rqta]*[APRE]) AS TOTALE FROM ARTICOLI
INNER JOIN RIGHE ON ARTICOLI.ACOD = RIGHE.RART GROUP BY
ARTICOLI.ADES;
```

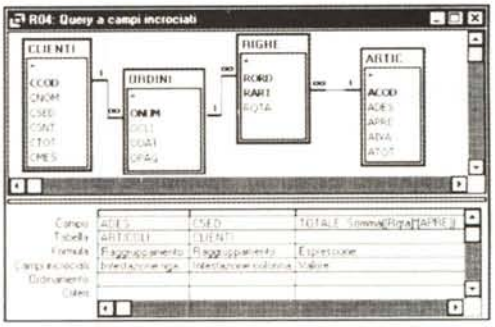


Anche le query che creano dei Raggruppamenti possono lavorare su più tabelle relazionate tra di loro. Se vogliamo calcolare il totale fatturato (od ordinato) per ciascun articolo occorrono solo due tabelle **Articoli** e **Righe**.

R04 - Fatturato per Articolo e per Sede

```
TRANSFORM Sum([RQTA]*[APRE]) AS TOTALE SELECT ADES FROM
(CLIENTI INNER JOIN ORDINI ON CLIENTI.CCOD = ORDINI.OCLI)
INNER JOIN (ARTICOLI INNER JOIN RIGHE ON ARTICOLI.ACOD =
RIGHE.RART) ON ORDINI.ONUM = RIGHE.RORD GROUP BY ADES
PIVOT CSED;
```

Complichiamo l'esercizio precedente aggiungendo il campo **CSED** (dalla tabella Clienti).
Vogliamo un'analisi **Campi Incrociati**, delle vendite per articolo **ADES** e per città **CSED**.



R05 - Clienti che non hanno fatto Ordini

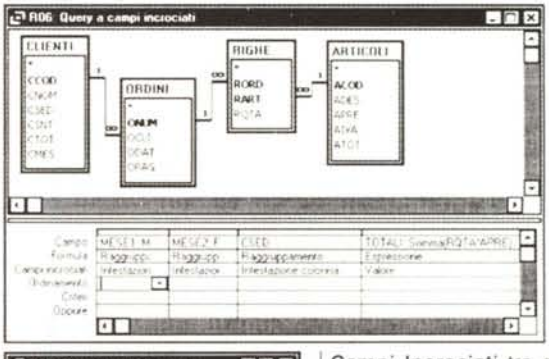
```
SELECT CNOM FROM CLIENTI LEFT JOIN ORDINI ON CLIENTI.CCOD
= ORDINI.OCLI WHERE ONUM Is Null;
```

Cominciamo a vedere le tre varianti delle **Join**.
In caso tabelle unite da **Relazione**, Access normalmente mostra solo i record collegati, ad esempio se selezionamo le tabelle Clienti ed Ordini mostra solo i clienti che hanno ordini. Per vedere anche i clienti che non hanno ordini occorre variare il tipo di **Join**.

Un doppio click sulla linea che unisce le due tabelle fa apparire la Box con le tre opzioni.
Vogliamo individuare i **Clienti** che non hanno fatto **Ordini**. Si sceglie la variante 2, che fa vedere comunque i clienti (anche senza ordini).
Poi per non vedere i clienti che hanno comunque ordini basta inserire un criterio **Is Null** su qualsiasi campo (non vuoto) della tabella ordini.
Significa: mostra tutti i clienti che non hanno corrispondenze nei campi della tabella Ordini.

R06 - Fatturato per Sede e per Mese

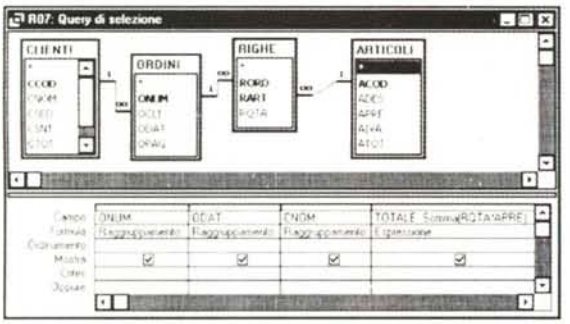
```
TRANSFORM Sum(RQTA*APRE) AS TOTALI SELECT Month(ODAT) AS
MESE1, Format(ODAT,"mmmm") AS MESE2 FROM CLIENTI INNER
JOIN (ARTICOLI INNER JOIN (ORDINI INNER JOIN RIGHE ON
ORDINI.ONUM = RIGHE.RORD) ON ARTICOLI.ACOD = RIGHE.RART)
ON CLIENTI.CCOD = ORDINI.OCLI GROUP BY Month(ODAT),
Format(ODAT,"mmmm") PIVOT CSED;
```



Campi Incrociati tra città **CSED** e mese della Data (**ODAT** campo della tabella Ordini). Per mostrare il mese in chiaro usiamo la funzione FORMAT, per mettere in ordine di mese usiamo la funzione MONTH.

R07 - Elenco Completo Ordini

```
SELECT ONUM, ODAT, CNOM, Sum(RQTA*APRE) AS TOTALE FROM
(CLIENTI INNER JOIN ORDINI ON CLIENTI.CCOD = ORDINI.OCLI)
INNER JOIN (ARTICOLI INNER JOIN RIGHE ON ARTICOLI.ACOD =
RIGHE.RART) ON ORDINI.ONUM = RIGHE.RORD GROUP BY ONUM,
ODAT, CNOM;
```



In questo caso eseguiamo una vista sui dati partendo dagli **Ordini**. Per ogni ordine vediamo campi provenienti dalla tabella **Clienti** (un ordine un cliente) e campi provenienti dalla tabella **Righe** e **Articoli** (un ordine è costituito da molte righe). Eseguiamo anche il calcolo del **Totale** dell'ordine, raggruppando per ordine e sommando il calcolo tra prezzo e quantità.

R08 – Classifica Articoli venduti non a Roma

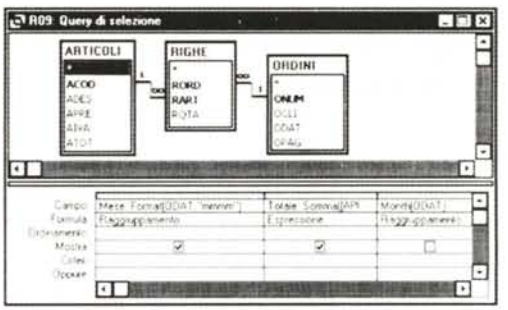
```
SELECT ADES, Sum(RQTA) AS TOTALE FROM CLIENTI INNER JOIN
(ORDINI INNER JOIN (ARTICOLI INNER JOIN RIGHE ON
ARTICOLI.ACOD = RIGHE.RART) ON ORDINI.ONUM = RIGHE.RORD)
ON CLIENTI.CCOD = ORDINI.OCLI WHERE Not (CSED)="ROMA"
GROUP BY ADES ORDER BY Sum(RQTA) DESC;
```

Può essere eseguito qualsiasi tipo di analisi sui dati prelevando campi da qualsiasi tabella del database.
In ogni caso i corretti collegamenti tra i campi delle varie tabelle sono garantiti dalle relazioni.
Vediamo di nuovo l'utilizzo della clausola **DOVE** che entra in gioco quando c'è un criterio che dà fastidio al raggruppamento.

R09 – Importo Totale per Mese

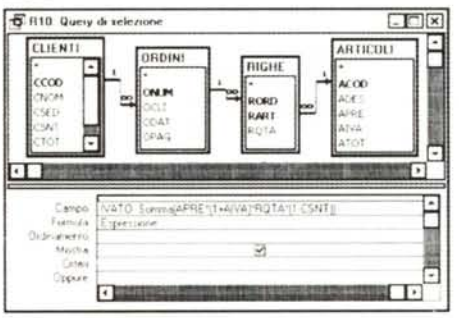
```
SELECT Format(ODAT,"mmmm") AS Mese, Sum([APRE]*[RQTA]) AS
Totale FROM ORDINI INNER JOIN (ARTICOLI INNER JOIN RIGHE
ON ARTICOLI.ACOD = RIGHE.RART) ON ORDINI.ONUM = RIGHE.RORD
GROUP BY Format(ODAT,"mmmm"), Month(ODAT);
```

Altro calcolo analitico ottenuto pescando campi dalle varie tabelle. Viene anche usata la funzione **Format** per tradurre la data nel suo mese. I campi sono tre, il mese in chiaro, il mese in numero, che serve per ordinare, il totale delle righe di vendita.



R10 - Elenco Analitico degli Ordini

```
SELECT CCOD, CNOM, ONUM, ODAT, Sum(APRE*RQTA) AS LORDO,
Sum(APRE*RQTA*(1-CSNT)) AS NETTO,
Sum(APRE*(1+AIVA)*RQTA*(1-CSNT)) AS IVATO FROM (CLIENTI
LEFT JOIN ORDINI ON CLIENTI.CCOD = ORDINI.OCLI) LEFT JOIN
(ARTICOLI RIGHT JOIN RIGHE ON ARTICOLI.ACOD = RIGHE.RART)
ON ORDINI.ONUM = RIGHE.RORD GROUP BY CCOD, CNOM, ONUM,
ODAT;
```



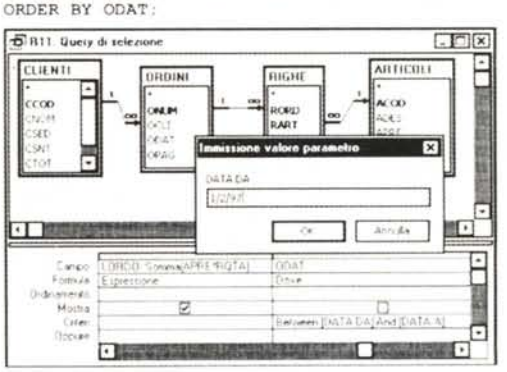
Query che serve per eseguire tutti i calcoli necessari per ciascun ordine. Le colonne desiderate sono:

Lordo totale importo delle righe (**APRE** per **RQTA**)
Netto applicazione a Lordo dello sconto (**CSNT** del cliente)
Ivato applicazione dell'IVA al Netto (**AIVA** dell'articolo).

Da notare il fatto che, in caso di raggruppamento, non è possibile eseguire calcoli in sequenza, ovvero utilizzare in un calcolo una colonna calcolata precedente.

R11 - Elenco degli Ordini in Intervallo di Date

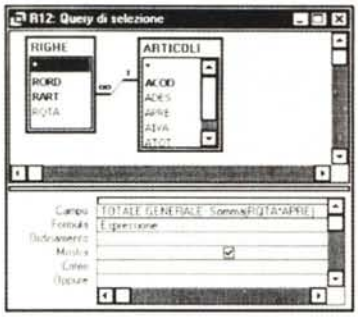
```
SELECT CNOM, ONUM, ODAT, Sum(APRE*RQTA) AS LORDO FROM
(CLIENTI LEFT JOIN ORDINI ON CLIENTI.CCOD = ORDINI.OCLI)
LEFT JOIN (ARTICOLI RIGHT JOIN RIGHE ON ARTICOLI.ACOD =
RIGHE.RART) ON ORDINI.ONUM = RIGHE.RORD WHERE ORDINI.ODAT
Between [DATA DA] And [DATA A] GROUP BY CNOM, ONUM, ODAT
ORDER BY ODAT;
```



Quando c'è un campo di tipo **Data** occorre sempre prevedere delle query parametrizzate che selezionano i dati secondo un certo intervallo. I parametri sono **DATA DA** e **DATA A**.

R12 - Totale Generale

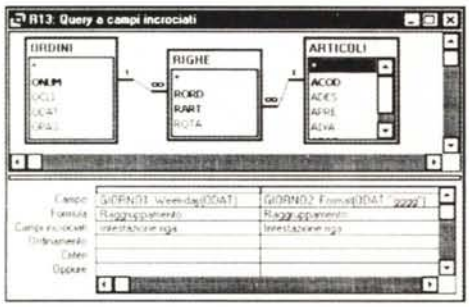
```
SELECT Sum(RQTA*APRE) AS [TOTALE GENERALE] FROM ARTICOLI
INNER JOIN RIGHE ON ARTICOLI.ACOD = RIGHE.RART;
```



Il nostro obiettivo è quello di calcolare il **Totale Generale** di tutti gli Ordini.
Scelte le due tabelle **Righe** ed **Articoli**, ed inserito il solo campo calcolato, **Totale**, tra prezzo **APRE** e quantità **RQTA**, basta raggruppare e sommare il solo campo inserito.
Una sola colonna che produce un solo valore.

R13 - Statistiche per Giorno della Settimana

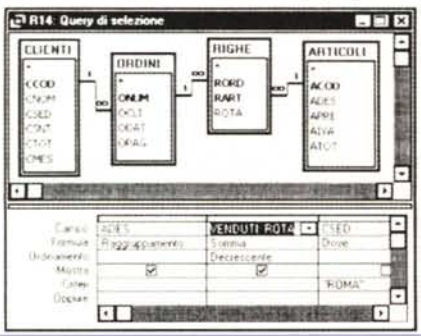
```
TRANSFORM Sum(RQTA) AS TOTALE SELECT Weekday(ODAT) AS
GIORNO1, Format(ODAT,"dddd") AS GIORNO2 FROM ORDINI INNER
JOIN (ARTICOLI INNER JOIN RIGHE ON ARTICOLI.ACOD =
RIGHE.RART) ON ORDINI.ONUM = RIGHE.RORD GROUP BY
Weekday(ODAT), Format(ODAT,"dddd") PIVOT ADES;
```



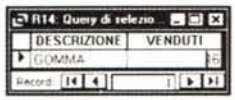
Altro esempio di calcolo evoluto, basato sulla funzione che estrae il giorno della settimana da una Data e sulla funzione che visualizza, in chiaro, lo stesso giorno della settimana.
Le funzioni sono **Weekday**(data) che restituisce un numero e la **Format**(data,"dddd") che restituisce il giorno della settimana come stringa. Usiamo quest'ultima espressione in una query di tipo **Campi Incrociati** che serva per vedere come vengono venduti, nella settimana, i vari articoli.

R14 - Articolo più venduto a Roma

```
SELECT TOP 1 ADES, Sum(RQTA) AS VENDUTI FROM (CLIENTI
INNER JOIN ORDINI ON CLIENTI.CCOD = ORDINI.OCLI) INNER
JOIN (ARTICOLI INNER JOIN RIGHE ON ARTICOLI.ACOD =
RIGHE.RART) ON ORDINI.ONUM = RIGHE.RORD WHERE CSED="ROMA"
GROUP BY ADES ORDER BY Sum(RQTA) DESC;
```



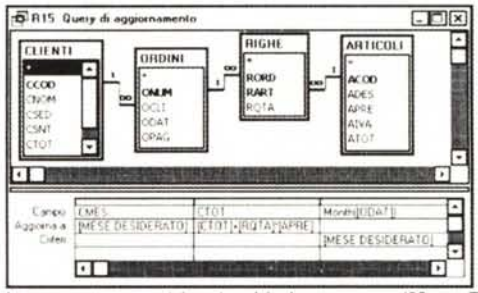
Tra le proprietà della Query va citata quella che si chiama **Primi Valori** e che serve per limitare il numero delle righe in uscita. La relativa clausola SQL è la **TOP**.
Se usata in una query che esegue un ordinamento crea una classifica del primo N.



La clausola TOP non va confusa con un Criterio, non serve a selezionare le righe ma solo a limitare l'output.

R15 - Calcolo Batch del fatturato per Cliente nel Mese

```
UPDATE (CLIENTI INNER JOIN ORDINI ON CLIENTI.CCOD =
ORDINI.OCLI) INNER JOIN (ARTICOLI INNER JOIN RIGHE ON
ARTICOLI.ACOD = RIGHE.RART) ON ORDINI.ONUM = RIGHE.RORD
SET CLIENTI.CMES = [MESE DESIDERATO], CLIENTI.CTOT =
CTOT+RQTA*APRE WHERE Month([ODAT]) = [MESE DESIDERATO];
```



Query di **Aggiornamento** che aggiorna un campo di una tabella con il risultato di un calcolo eseguito su campi di altre tabelle

La query parametrizzata chiede un mese (**Mese Desiderato**), ad esempio 1, e poi aggiorna i due campi **CMES** e **CTOT** della tabella Clienti, il primo con il mese del calcolo e il secondo con il totale delle vendite in quel mese.
Non si possono usare, nelle query di aggiornamento, le funzionalità di raggruppamento, allora occorre utilizzare una formula diretta

```
[CTOT] = [CTOT] + [RQTA] * [APRE]
```

R16 - Fatturato per Cliente – Bis

```
SELECT R01.CNOM, Sum(R01.TOTALE) AS TOT FROM R01 GROUP
BY R01.CNOM;
```

Eseguiamo la stessa query dell'esercizio **R02**, basandoci sulla query **R01** che, in un certo senso, propone in forma piatta tutti i dati che ci servono.
Esiste sempre la possibilità di fare **Catene di Query** che permettono di semplificare, scomponendoli, i vari problemi.

R17 - Aggiornamento Batch Ordinato per Articolo

```
UPDATE ARTICOLI INNER JOIN RIGHE ON ARTICOLI.ACOD =
RIGHE.RART SET ATOT = ATOT+RQTA*APRE;
```

Calcolo Batch, con una **Query di Aggiornamento**, del totale venduto per **Articolo**
Viene alimentato il campo **ATOT** della tabella Articoli con il risultato del prodotto tra **APRE** della tabella Articoli e **RQAT** della tabella Righe.
Anche in questo caso non abbiamo potuto usare le funzionalità di raggruppamento.

LINGOTTO

Torino Lingotto Fiere 4-8 dicembre '97

Cowboys delle consolle e smanettoni intrippati

Nonni digitali e nipoti in ansia di futuro ☞ Internettisti e interinali ☞ **Videoegoisti e spiriti gentili alla ricerca di idee digitali per regali di Natale** ☞ Educatori multimediali e content providers • Fantasmi di Baudelaire e mutanti

Cellularisti vibratili e predatori di carte telefoniche

Imprenditori e Managers "on-line" e impiegati senza scrivania

Pirati innamorati e hackers mercenari • Navigatori approdati e navigatori naufraghi • Dentisti telematici e adoratori di TV satellitare con sguardo al collirio • Agenti intelligenti e sciamani ciberiani • Cittadini in cerca di nuove forme di cittadinanza e tuttologi curiosi • HTMLeisti assatanati e scenaristi ispirati • Multitask-force e cibermartiri della body modification community • Presenzialisti on line e giornalisti inviati nel futuro • Posse digitali e navigatori solitari • Docenti a distanza e discenti in avvicinamento • Trovatori di ragni e cercatori di bachi • Amministratori pubblici in aggiornamento e funzionari senza rete • Notai dal cd-rom facile e agronomi in terra digitale • Patiti del terabyte e genitori in stress di recupero • Pubblicitari convertiti al pixel e progettisti olistici • Videodepressi e videoeccitati • Crackers pestilenziali e webdesigners raffinati • Windowisti di massa e melisti d' elite • Spettatori interattivi e interpassivi •Ingegneri della virtualità e virtuosi ingegneri • Cibernauti idealisti e intermediari d'affari immateriali • Scrittori senza penna ed editori senza carta • Autori in cerca di editori ed editori in cerca d'autore • Netsurfers dal motore immobile e psiconomadi predestinati • Telelavoratori e teledisoccupati • Singles del villaggio globale e comunità virtuali • Utenti unix e utenti unisex

Una sola moltitudine al SalonB.it

- **Mostra mercato**
- **Forum d'aggiornamento**
- **Eventi interattivi**

SalonB.it

Salone del multimedia e dello spettacolo digitale

TEMPO LIBERO • EDUCAZIONE • IMPRESA

www.salonb.it

EXPO cardi

1° raduno internazionale collezionisti carte telefoniche

Organizzazione:
EUPHON, c/o Lingotto
Via Nizza, 294 - 10126 TORINO
Tel. 011. 6644216/26 - Fax 011. 6635095
e-mail: salonbit@euphon.it

Forum/eventi:
POLIEDRA,
Corso Unione Sovietica, 612/3e - 10135 TORINO
Tel. 011. 3912600 - Fax 011. 3912601
e-mail: salonbit@poliedra.it