Prima di poter iniziare a eseguire gli esercizi di questo capitolo, dedicato alle basi di dati e al linguaggio SQL, è necessario verificare di disporre degli strumenti adatti ed essere sicuri di saperli utilizzare.

Per facilitare l'esecuzione di queste esercitazioni, sia gli esercizi, sia le verifiche sono realizzabili con SQLite, attraverso il programma '**`sqlite3`**'. Pertanto gli esercizi prevedono l'uso di basi di dati personali, ognuna contenuta tutta in un file.

Le verifiche associate a queste esercitazioni portano a produrre dei fogli stampati, che gli studenti devono avere la cura di controllarle in base a quanto indicato nella traccia delle verifiche stesse, prima della consegna all'insegnante.

Per poter svolgere gli esercizi e le verifiche, ogni studente deve essere in grado di scrivere e modificare file di testo, con un programma adatto (per esempio va bene il programma Gedit). In questi file di testo vanno inserite le istruzioni SQL necessarie allo svolgimento del lavoro; per evitare confusione, i file che contengono codice SQL vengono nominati con l'estensione '`.sql`'.

Per eseguire le istruzioni SQL contenute in un file, si usa il programma '**`sqlite3`**' nel modo seguente:

```
$ sqlite3 file_db < file_sql [Invio]
```

In questo caso, le istruzioni contenute nel file ***file_sql***, vengono applicate alla base di dati contenuta nel file ***file_db***.

Per essere certi di sapere usare gli strumenti occorre fare una prova. Si realizzi il file di testo denominato '`prova-istruzioni.sql`', contenente quanto segue, sostituendo le metavariabili ***cognome***, ***nome***, ***classe*** e ***data*** con qualcosa di appropriato:

```
-- Esercizio di prova di: cognome nome classe
-- Data: data
-- File: prova-istruzioni.sql

CREATE TABLE Prova (Codice  INTEGER,
                    Cognome VARCHAR(60),
                    Nome    VARCHAR(60));
```

Una volta salvato il file con il nome stabilito, lo si esegue nella base di dati contenuta nel file '`prova.db`'. Dal momento che il file '`prova.db`' non esiste, essendo la prima volta che viene utilizzato questo nome, l'esecuzione delle istruzioni comporta automaticamente la creazione della base di dati relativa:

```
$ sqlite3 prova.db < prova-istruzioni.sql [Invio]
```

Se non si vedono segnalazioni di alcun genere, le istruzioni contenute nel file '`prova-istruzioni.sql`' sono state eseguite tutte con successo.

Le istruzioni contenute nel file '`prova-istruzioni.sql`' servono a produrre una ***relazione***, denominata '**`Prova`**', contenente alcuni ***attributi*** ('**`Codice`**', '**`Cognome`**' e '**`Nome`**').

Se il file contenente le istruzioni SQL contiene degli errori, o viene eseguito quando ciò non deve essere fatto, è probabile vedere apparire dei messaggi, che vanno letti attentamente. Per esempio, se venisse eseguito nuovamente il file '`prova-istruzioni.sql`' nella stessa base di dati, si otterrebbe una segnalazione che avvisa del fatto che la relazione '**`Prova`**' esiste già (e non può essere creata nuovamente):

```
$ sqlite3 prova.db < prova-istruzioni.sql [Invio]

  CREATE TABLE Prova (Codice  INTEGER,
                      Cognome VARCHAR(60),
                      Nome    VARCHAR(60));
  SQL error: table Prova already exists
```

Il programma '**`sqlite3`**' può essere usato anche interattivamente. Per farlo, si avvia senza indicare il file contenente le istruzioni SQL. Si proceda in questo modo:

```
$ sqlite3 prova.db [Invio]
```

A questo punto appare l'invito del programma '**sqlite3**', che indica la sua attesa di comandi o di istruzioni SQL, digitati direttamente:

```
sqlite>
```

Con il comando '**.schema**' (si osservi il fatto che il comando inizia con un punto) è possibile visualizzare l'elenco delle relazioni esistenti, in forma di istruzione SQL. Si proceda inserendo questo comando:

```
sqlite> .schema [Invio]

  CREATE TABLE Prova (Codice  INTEGER,
                      Cognome VARCHAR(60),
                      Nome    VARCHAR(60));
```

Si proceda inserendo l'istruzione necessaria a eliminare la relazione '**Prova**' creata poco prima:

```
sqlite> DROP TABLE Prova; [Invio]
```

Si conclude con il funzionamento di '**sqlite3**' con il comando '**.quit**':

```
sqlite> .quit [Invio]
```

Prima di passare alle sezioni successive, vanno eliminati i file '`prova-istruzioni.sql`' e '`prova.db`' che non servono più.

In questi esercizi vengono creati i file seguenti, elencati in ordine alfabetico, con il riferimento alla sezione in cui sono utilizzati:

79.1 Creazione ed eliminazione delle relazioni

Una ***relazione*** è un insieme di ***tuple***, suddivise in ***attributi***, tutte nello stesso modo. Una relazione si rappresenta normalmente in forma tabellare, dove le tuple sono costituite dalle righe e gli attributi dalle colonne.

Figura 79.4. Relazione «Articoli (Articolo, Descrizione, UM, Listino, ScortaMin)».

Articolo	Descrizione	UM	Listino	ScortaMin
1	Dischetti da 9 cm 1440 Kibyte	pz	0,20	500
2	Dischetti da 9 cm 1440 Kibyte colorati	pz	0,25	500
101	CD-R 16x	pz	0,50	500
102	CD-R 52x	pz	1,00	500
201	CD-RW 4x	pz	1,00	200
202	CD-RW 8x	pz	1,50	200
301	DVD-R 8x	pz	1,00	200
302	DVD-R 16x	pz	2,00	200
401	DVD+R 8x	pz	1,00	200
402	DVD+R 16x	pz	2,00	200
501	DVD-RW 8x	pz	2,00	200
601	DVD+RW 8x	pz	2,00	200

I valori che si possono inserire nelle celle della tabella dipendono dal ***dominio*** dell'attributo relativo. Per esempio, l'attributo '**Listino**' (corrispondente alla quarta colonna) della relazione '**Articoli**', può contenere solo valori numerici positivi, con un massimo di due decimali.

79.1.1 Creazione di una relazione

Con l'ausilio di un programma per la scrittura e la modifica di file di testo puro, si crei il file '`prova-creazione-articoli.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-creazione-articoli.sql

CREATE TABLE Articoli (Articolo    INTEGER        NOT NULL,
                       Descrizione CHAR(60)       NOT NULL,
                       UM          CHAR(7)        DEFAULT 'pz',
                       Listino     NUMERIC(14,2)  DEFAULT 0,
                       ScortaMin   NUMERIC(12,3)  DEFAULT 0,
                       PRIMARY KEY (Articolo));

INSERT INTO Articoli VALUES
    (1, 'Dischetti da 9 cm 1440 Kibyte', 'pz', 0.2, 500);
INSERT INTO Articoli VALUES
    (2, 'Dischetti da 9 cm 1440 Kibyte colorati', 'pz', 0.25, 500);
```

L'istruzione '**CREATE TABLE**' permette la creazione della relazione '**Articoli**', stabilendo dei vincoli, per cui gli attributi '**Articolo**' e '**Descrizione**' non possono contenere un valore nullo; inoltre viene stabilito il valore predefinito per gli altri attributi. Si stabilisce anche che l'attributo '**Articolo**' deve essere una chiave primaria, comportando la necessità che non appaiano tuple con lo stesso codice articolo.

Le istruzioni '**INSERT**', inseriscono le prime due tuple della relazione. A questo proposito, si osservi che i dati numerici, come il prezzo di listino e il livello della scorta minima, si indicano così come sono, con l'accortezza di usare **il punto per la separazione dei decimali**, mentre **le stringhe** (le informazioni testuali) **vanno delimitate da apici singoli**.

Si controlli di avere scritto il file ‘`prova-creazione-articoli.sql`’ in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con li nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < prova-creazione-articoli.sql [Invio]
```

Se il programma mostra dei messaggi, si tratta di errori, che devono essere verificati attentamente, prima di proseguire.

Si ricorda che il file ‘`mag.db`’, contenente la base di dati, viene creato automaticamente se non dovesse già essere presente.

79.1.2 Eliminazione di una relazione

Per eliminare una relazione si usa l’istruzione ‘**`DROP TABLE`**’, come nell’esempio seguente:

```
DROP TABLE Articoli;
```

Si vuole eliminare la relazione ‘**`Articoli`**’ appena creata nella base di dati contenuta nel file ‘`mag.db`’, ma trattandosi di un’operazione molto semplice, è meglio usare il programma ‘**`sqlite3`**’ in modo interattivo. Si avvii il programma ‘**`sqlite3`**’ e si eseguano i comandi successivi, come descritto qui di seguito, utilizzando anche il comando ‘**`.schema`**’ per avere l’elenco delle relazioni esistenti, prima di cancellare effettivamente quella stabilita:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions

sqlite> .schema [Invio]

  CREATE TABLE Articoli (Articolo    INTEGER        NOT NULL,
                         Descrizione CHAR(60)       NOT NULL,
                         UM          CHAR(7)        DEFAULT 'pz',
                         Listino     NUMERIC(14,2)  DEFAULT 0,
                         ScortaMin   NUMERIC(12,3)  DEFAULT 0,
                         PRIMARY KEY (Articolo));

sqlite> DROP TABLE Articoli; [Invio]

sqlite> .quit [Invio]
```

Si ricorda che se, a seguito dell’inserimento dell’istruzione ‘**`DROP TABLE`**’, il programma mostra dei messaggi, si tratta di errori che devono essere verificati attentamente, prima di proseguire.

79.1.3 Verifica sulla creazione e popolazione della relazione «Articoli»

Per poter svolgere questa verifica, gli studenti devono essere in grado di realizzare un file di testo contenente codice SQL, con le istruzioni necessarie alla creazione di una relazione e con quelle che permettono l’inserimento delle tuple. Inoltre, devono essere in grado di utilizzare il programma ‘**`sqlite3`**’ in modo interattivo, per visualizzare l’elenco delle relazioni esistenti nella base di dati e per eliminare una relazione.

Si riprenda il file ‘`prova-creazione-articoli.sql`’ e lo si salvi con il nome ‘`creazione-articoli.sql`’. Il file ‘`creazione-articoli.sql`’ va poi modificato aggiungendo le istruzioni necessarie a completare l’inserimento degli articoli che sono visibili nella figura 79.4.

Una volta completato e salvato il file ‘`creazione-articoli.sql`’, se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < creazione-articoli.sql [Invio]
```

Se si ottengono degli errori, si deve eliminare la relazione ‘**`Articoli`**’ dalla base di dati contenuta nel file ‘`mag.db`’, utilizzando il programma ‘**`sqlite3`**’ in modo interattivo, quindi, dopo le correzioni, si deve riprovare.

Una volta eseguita l’operazione con successo, si stampi il file ‘`creazione-articoli.sql`’ e lo si consegni per la correzione all’insegnante.

Nella valutazione viene controllata la correttezza del contenuto del file e la coerenza estetica nella scrittura delle istruzioni SQL.

79.1.4 Verifica sulla creazione e popolazione della relazione «Causali»

Prima di svolgere questa verifica, è necessario avere svolto quella precedente, della quale valgono anche gli stessi requisiti.

Si crei il file ‘`creazione-causali.sql`’, inserendo le istruzioni necessarie a creare la relazione ‘**`Causali`**’, con il contenuto che si vede nella figura 79.9, tenendo conto che:

1. l’attributo ‘**`Causale`**’ è di tipo ‘**`INTEGER`**’, non ammette il valore nullo e costituisce la chiave primaria;
2. l’attributo ‘**`Descrizione`**’ è di tipo ‘**`CHAR`**’ a 60 caratteri e non ammette il valore nullo;
3. l’attributo ‘**`Variazione`**’ è di tipo ‘**`NUMERIC`**’, a una sola cifra, senza decimali, con un valore predefinito pari a zero.

Figura 79.9. Relazione **`Causali (Causale, Descrizione, Variazione)`**.

Causale	Descrizione	Variazione
1	Carico per acquisto	+1
2	Scarico per vendita	-1
3	Reso da cliente	+1
4	Reso a fornitore	-1
5	Rettifica aumento acquisto	+1
6	Rettifica aumento vendite	-1
7	Rettifica diminuzione vendite	+1
8	Rettifica diminuzione acquisti	-1
9	Carico da produzione	+1
10	Scarico a produzione	-1
11	Carico da altro magazzino	+1
12	Scarico ad altro magazzino	-1
13	Saldo iniziale	+1

Figura 79.10. Scheletro del file ‘`creazione-causali.sql`’, da completare.

```
-- Creazione della relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: creazione-causali.sql

CREATE TABLE Causali …

INSERT INTO Causali …
…
```

Una volta completato e salvato il file ‘`creazione-causali.sql`’, se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < creazione-causali.sql [Invio]
```

Se si ottengono degli errori, si deve eliminare la relazione ‘**`Causali`**’ dalla base di dati contenuta nel file ‘`prova.db`’, utilizzando il programma ‘**`sqlite3`**’ in modo interattivo, quindi, dopo le correzioni, si deve riprovare.

Una volta eseguita l’operazione con successo, si stampi il file ‘`creazione-causali.sql`’ e lo si consegni per la correzione all’insegnante.

Nella valutazione viene controllata la correttezza del contenuto del file e la coerenza estetica nella scrittura delle istruzioni SQL.

79.1.5 Verifica sulla creazione e popolazione della relazione «Fornitori»

Prima di svolgere questa verifica, è necessario avere svolto quelle precedenti, delle quali valgono anche gli stessi requisiti.

Si crei il file ‘`creazione-fornitori.sql`’, inserendo le istruzioni necessarie a creare la relazione ‘**`Fornitori`**’, con il contenuto che si vede nella figura 79.11, tenendo conto che:

1. l’attributo ‘**`Fornitore`**’ è di tipo ‘**`INTEGER`**’, non ammette il valore nullo e costituisce la chiave primaria;
2. l’attributo ‘**`RagioneSociale`**’ è di tipo ‘**`VARCHAR`**’ a 120 caratteri e non ammette il valore nullo;
3. l’attributo ‘**`Paese`**’ è di tipo ‘**`CHAR`**’ a 30 caratteri e il suo valore predefinito è ‘**`ITALIA`**’;
4. l’attributo ‘**`Indirizzo`**’ è di tipo ‘**`VARCHAR`**’ a 120 caratteri e non ammette il valore nullo;
5. l’attributo ‘**`CAP`**’ è di tipo ‘**`CHAR`**’ a 10 caratteri;
6. l’attributo ‘**`Citta`**’ è di tipo ‘**`VARCHAR`**’ a 120 caratteri e non ammette il valore nullo;
7. l’attributo ‘**`Prov`**’ è di tipo ‘**`CHAR`**’ a 2 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
8. l’attributo ‘**`Telefono`**’ è di tipo ‘**`CHAR`**’ a 20 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
9. l’attributo ‘**`Fax`**’ è di tipo ‘**`CHAR`**’ a 20 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
10. l’attributo ‘**`CFPI`**’ (codice fiscale o partita IVA) è di tipo ‘**`CHAR`**’ a 30 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla.

Figura 79.11. Relazione **`Fornitori (Fornitore, RagioneSociale, Paese, Indirizzo, CAP, Citta, Prov, Telefono, Fax, CFPI)`**.

Fornitore	Ragione-Sociale	Paese	Indirizzo	CAP	Citta	Prov	Telefono	Fax	CFPI
1	Tizio Tizi	ITALIA	via Tazio, 11	31100	Treviso	TV	0422,111111	0422,222222	12345678901
2	Caio Cai	ITALIA	via Caino, 22	31033	Castelfranco Veneto	TV	0423,222222	0423,333333	23456789012
3	Sempronio Semproni	ITALIA	via Salina, 33	31057	Silea	TV	0422,333333	0422,444444	34567890123

Figura 79.12. Scheletro del file ‘`creazione-fornitori.sql`’, da completare.

```
-- Creazione della relazione "Fornitori"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: creazione-fornitori.sql

CREATE TABLE Fornitori …

INSERT INTO Fornitori …
…
```

Una volta completato e salvato il file ‘`creazione-fornitori.sql`’, se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < creazione-fornitori.sql [Invio]
```

Se si ottengono degli errori, si deve eliminare la relazione ‘**`Fornitori`**’ dalla base di dati contenuta nel file ‘`mag.db`’, utilizzando il programma ‘**`sqlite3`**’ in modo interattivo, quindi, dopo le correzioni, si deve riprovare.

Una volta eseguita l’operazione con successo, si stampi il file ‘`creazione-fornitori.sql`’ e lo si consegni per la correzione all’insegnante.

Nella valutazione viene controllata la correttezza del contenuto del file e la coerenza estetica nella scrittura delle istruzioni SQL.

79.1.6 Verifica sulla creazione e popolazione della relazione «Clienti»

Prima di svolgere questa verifica, è necessario avere svolto quelle precedenti, delle quali valgono anche gli stessi requisiti.

Si crei il file ‘`creazione-clienti.sql`’, inserendo le istruzioni necessarie a creare la relazione ‘**`Clienti`**’, con il contenuto che si vede nella figura 79.13, tenendo conto che:

1. l’attributo ‘**`Cliente`**’ è di tipo ‘**`INTEGER`**’, non ammette il valore nullo e costituisce la chiave primaria;
2. l’attributo ‘**`RagioneSociale`**’ è di tipo ‘**`VARCHAR`**’ a 120 caratteri e non ammette il valore nullo;
3. l’attributo ‘**`Paese`**’ è di tipo ‘**`CHAR`**’ a 30 caratteri e il suo valore predefinito è ‘**`ITALIA`**’;
4. l’attributo ‘**`Indirizzo`**’ è di tipo ‘**`VARCHAR`**’ a 120 caratteri e non ammette il valore nullo;
5. l’attributo ‘**`CAP`**’ è di tipo ‘**`CHAR`**’ a 10 caratteri;
6. l’attributo ‘**`Citta`**’ è di tipo ‘**`VARCHAR`**’ a 120 caratteri e non ammette il valore nullo;
7. l’attributo ‘**`Prov`**’ è di tipo ‘**`CHAR`**’ a 2 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
8. l’attributo ‘**`Telefono`**’ è di tipo ‘**`CHAR`**’ a 20 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
9. l’attributo ‘**`Fax`**’ è di tipo ‘**`CHAR`**’ a 20 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
10. l’attributo ‘**`CFPI`**’ (codice fiscale o partita IVA) è di tipo ‘**`CHAR`**’ a 30 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla.

Figura 79.13. Relazione **`Clienti (Cliente, RagioneSociale, Paese, Indirizzo, CAP, Citta, Prov, Telefono, Fax, CFPI)`**.

Cliente	Ragione-Sociale	Paese	Indirizzo	CAP	Citta	Prov	Telefono	Fax	CFPI
1	Mevio Mevi	ITALIA	via Mare, 11	31050	Morgano	TV	0422,444444	0422,555555	45678901234
2	Filano Filani	ITALIA	via Farfalle, 22	31032	Feltre	BL	0439,555555	0439,666666	56789012345
3	Martino Martini	ITALIA	via Marte, 33	31010	Mareno di Piave	TV	0438,666666	0438,777777	67890123456

Figura 79.14. Scheletro del file ‘`creazione-clienti.sql`’, da completare.

```
-- Creazione della relazione "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: creazione-clienti.sql

CREATE TABLE Clienti …

INSERT INTO Clienti …
…
```

Una volta completato e salvato il file ‘`creazione-clienti.sql`’, se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < creazione-clienti.sql [Invio]
```

Se si ottengono degli errori, si deve eliminare la relazione ‘**`Clienti`**’ dalla base di dati contenuta nel file ‘`mag.db`’, utilizzando il programma ‘**`sqlite3`**’ in modo interattivo, quindi, dopo le correzioni, si deve riprovare.

Una volta eseguita l’operazione con successo, si stampi il file ‘`creazione-clienti.sql`’ e lo si consegni per la correzione all’insegnante.

Nella valutazione viene controllata la correttezza del contenuto del file e la coerenza estetica nella scrittura delle istruzioni SQL.

79.1.7 Verifica sulla creazione e popolazione della relazione «Movimenti»

Prima di svolgere questa verifica, è necessario avere svolto quelle precedenti, delle quali valgono anche gli stessi requisiti.

Si crei il file ‘`creazione-movimenti.sql`’, inserendo le istruzioni necessarie a creare la relazione ‘**`Movimenti`**’, con il contenuto che si vede nella figura 79.15, tenendo conto che:

1. l’attributo ‘**`Movimento`**’ è di tipo ‘**`INTEGER`**’, non ammette il valore nullo e costituisce la chiave primaria;
2. l’attributo ‘**`Articolo`**’ è di tipo ‘**`INTEGER`**’ e non ammette il valore nullo;

3. l'attributo '**Causale**' è di tipo '**INTEGER**' e non ammette il valore nullo;
4. l'attributo '**Data**' è di tipo '**DATE**' e non ammette il valore nullo;
5. l'attributo '**Cliente**' è di tipo '**INTEGER**';
6. l'attributo '**Fornitore**' è di tipo '**INTEGER**';
7. l'attributo '**Quantita**' è di tipo '**NUMERIC**' a 15 cifre, di cui cinque sono usate per i decimali, e non ammette il valore nullo;
8. l'attributo '**Valore**' è di tipo '**NUMERIC**' a 14 cifre, di cui due sono usate per i decimali, e non ammette il valore nullo.

Figura 79.15. Relazione **Movimenti (Movimento, Articolo, Causale, Data, Cliente, Fornitore, Quantita, Valore)**.

Movimento	Articolo	Causale	Data	Cliente	Fornitore	Quantita	Valore
1	2	1	2012-01-15	NULL	3	10000	100,00
2	2	2	2012-01-16	2	NULL	1000	10,00
3	102	1	2012-01-17	NULL	2	1000	200,00
4	102	2	2012-01-18	1	NULL	100	20,00
5	401	1	2012-01-19	NULL	1	1000	200,00
6	401	2	2012-01-20	3	NULL	200	40,00
7	401	4	2012-01-20	NULL	1	100	20,00
8	102	4	2012-01-20	NULL	2	100	20,00
9	601	1	2012-01-21	NULL	3	2000	1000,00
10	601	2	2012-01-25	1	NULL	1000	500,00

Figura 79.16. Scheletro del file '`creazione-movimenti.sql`', da completare.

```
-- Creazione della relazione "Movimenti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: creazione-movimenti.sql

CREATE TABLE Movimenti …

INSERT INTO Movimenti …
…
```

Si ricorda che i valori numerici vanno indicati come sono, con l'accortezza di usare il punto per la separazione dei decimali; le date, come le stringhe (i valori testuali) vanno delimitate con apici singoli. In questo caso, quando viene a mancare il valore per il cliente o il fornitore, si inserisce il valore «nullo», che si scrive con la parola chiave '**NULL**', come appare nella figura 79.15, senza usare apici.

Una volta completato e salvato il file '`creazione-movimenti.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < creazione-movimenti.sql [Invio]
```

Se si ottengono degli errori, si deve eliminare la relazione '**Movimenti**' dalla base di dati contenuta nel file '`mag.db`', utilizzando il programma '**sqlite3**' in modo interattivo, quindi, dopo le correzioni, si deve riprovare.

Una volta eseguita l'operazione con successo, si stampi il file '`creazione-movimenti.sql`' e lo si consegni per la correzione all'insegnante.

Nella valutazione viene controllata la correttezza del contenuto del file e la coerenza estetica nella scrittura delle istruzioni SQL.

79.1.8 Conclusione

Prima di passare alla sezione successiva, si deve realizzare un file contenente le istruzioni con cui creare e popolare le relazioni descritte in questa. In pratica, si tratta di copiare il contenuto dei file '`creazione-articoli.sql`', '`creazione-causali.sql`', '`creazione-fornitori.sql`', '`creazione-clienti.sql`' e '`creazione-movimenti.sql`', in un file completo, denominato '`magazzino.sql`'.

Una volta realizzato il file '`magazzino.sql`', si deve cancellare il file '`mag.db`' e ricreare a partire dalle istruzioni contenute nel file '`magazzino.sql`':

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

Se vengono segnalati degli errori, occorre correggere il file '`magazzino.sql`', cancellare nuovamente il file '`mag.db`', quindi si deve ripetere l'operazione. La base di dati contenuta nel file '`mag.db`', viene usata ancora e non si può proseguire se non si riesce a ricrearla correttamente.

All'inizio della sezione è stato creato il file '`prova-creazione-articoli.sql`' che a questo punto non serve più e va eliminato.

79.2 Interrogazione semplice di una relazione

Attraverso l'istruzione '**SELECT**' è possibile estrarre il contenuto di una o più relazioni simultaneamente. In questa sezione si mostrano alcune situazioni riferite a una sola relazione.

79.2.1 Interrogazione completa

Si ottiene l'elenco completo di una relazione utilizzando l'istruzione seguente:

```
SELECT * FROM nome_relazione
```

Si eseguano i passaggi seguenti, per ottenere la visualizzazione del contenuto complessivo della relazione '**Articoli**' e della relazione '**Causali**', così come dovrebbero essere contenute nella base di dati del file '**mag.db**':

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions

sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM Articoli; [Invio]

  Articolo    Descrizione                 UM          Listino     ScortaMin
  ----------  --------------------------  ----------  ----------  ----------
  1           Dischetti da 9 cm 1440 Kibyte  pz          0.2         500
  2           Dischetti da 9 cm 1440 Kibyte  pz          0.25        500
  101         CD-R 16x                    pz          0.5         500
  102         CD-R 52x                    pz          1           500
  201         CD-RW 4x                    pz          1           200
  202         CD-RW 8x                    pz          1.5         200
  301         DVD-R 8x                    pz          1           200
  302         DVD-R 16x                   pz          2           200
  401         DVD+R 8x                    pz          1           200
  402         DVD+R 16x                   pz          2           200
  501         DVD-RW 8x                   pz          2           200
  601         DVD+RW 8x                   pz          2           200

sqlite> SELECT * FROM Causali; [Invio]

  Causale     Descrizione          Variazione
  ----------  -------------------  ----------
  1           Carico per acquisto  1
  2           Scarico per vendita  -1
  3           Reso da cliente      1
  4           Reso a fornitore     -1
  5           Rettifica aumento a  1
  6           Rettifica aumento v  -1
  7           Rettifica diminuzio  1
  8           Rettifica diminuzio  -1
  9           Carico da produzion  1
  10          Scarico a produzion  -1
  11          Carico da altro mag  1
  12          Scarico ad altro ma  -1
  13          Saldo iniziale       1

sqlite> .quit [Invio]
```

Si osservi che i comandi '**.headers on**' e '**.mode column**' servono a ottenere un elenco incolonnato con le intestazioni, altrimenti, il risultato sarebbe poco gradevole esteticamente.

79.2.2 Interrogazione con selezione di alcuni attributi

Si ottiene l'elenco di tutte le tuple di una relazione, limitatamente a un certo gruppo di attributi, mettendo, al posto dell'asterisco, i nomi degli attributi desiderati:

```
SELECT attributo [, attributo] … FROM nome_relazione
```

Si eseguano i passaggi seguenti, per ottenere la visualizzazione del contenuto di tutte le tuple della relazione '**Articoli**', limitatamente

agli attributi '**Articolo**', '**Descrizione**' e '**Listino**', così come dovrebbero essere contenute nella base di dati del file '**mag.db**':

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT Articolo, Descrizione, Listino [Invio]

      >        FROM Articoli; [Invio]

  Articolo    Descrizione                    Listino
  ----------  -----------------------------  ----------
  1           Dischetti da 9 cm 1440 Kibyte  0.2
  2           Dischetti da 9 cm 1440 Kibyte  0.25
  101         CD-R 16x                       0.5
  102         CD-R 52x                       1
  201         CD-RW 4x                       1
  202         CD-RW 8x                       1.5
  301         DVD-R 8x                       1
  302         DVD-R 16x                      2
  401         DVD+R 8x                       1
  402         DVD+R 16x                      2
  501         DVD-RW 8x                      2
  601         DVD+RW 8x                      2
```

Intuitivamente, si comprende che si può anche cambiare l'ordine di visualizzazione degli attributi:

```
sqlite> SELECT Descrizione, Articolo, Listino [Invio]

      >        FROM Articoli; [Invio]

  Descrizione                    Articolo    Listino
  -----------------------------  ----------  ----------
  Dischetti da 9 cm 1440 Kibyte  1           0.2
  Dischetti da 9 cm 1440 Kibyte  2           0.25
  CD-R 16x                       101         0.5
  CD-R 52x                       102         1
  CD-RW 4x                       201         1
  CD-RW 8x                       202         1.5
  DVD-R 8x                       301         1
  DVD-R 16x                      302         2
  DVD+R 8x                       401         1
  DVD+R 16x                      402         2
  DVD-RW 8x                      501         2
  DVD+RW 8x                      601         2
```

Si conclude il funzionamento interattivo di '**sqlite3**' con il comando '**.quit**':

```
sqlite> .quit [Invio]
```

79.2.3 Stampa del contenuto di una relazione

Per ottenere la stampa del contenuto di una o di più relazioni, conviene scrivere le istruzioni necessarie in un file di testo, come già fatto in precedenza. Si proceda con la creazione del file '`prova-stampa-artico-caus.sql`', con il contenuto seguente, che ricalca quanto già mostrato nelle sezioni precedenti:

```
-- Stampa del contenuto delle relazioni "Articoli" e "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-stampa-artico-caus.sql

.headers on
.mode column

SELECT * FROM Articoli;
SELECT * FROM Causali;
```

Per verificare il funzionamento delle istruzioni contenute nel file '`stampa-artico-caus.sql`', si può utilizzare il comando seguente, che interviene nella base di dati contenuta nel file '`mag.db`', limitandosi a visualizzare il risultato:

```
$ sqlite3 mag.db < prova-stampa-artico-caus.sql [Invio]
```


Si dovrebbe ottenere il listato seguente:

```
Articolo    Descrizione                    UM          Listino     ScortaMin
----------  -----------------------------  ----------  ----------  ----------
1           Dischetti da 9 cm 1440 Kibyte  pz          0.2         500
2           Dischetti da 9 cm 1440 Kibyte  pz          0.25        500
101         CD-R 16x                       pz          0.5         500
102         CD-R 52x                       pz          1           500
201         CD-RW 4x                       pz          1           200
202         CD-RW 8x                       pz          1.5         200
301         DVD-R 8x                       pz          1           200
302         DVD-R 16x                      pz          2           200
401         DVD+R 8x                       pz          1           200
402         DVD+R 16x                      pz          2           200
501         DVD-RW 8x                      pz          2           200
601         DVD+RW 8x                      pz          2           200
Causale     Descrizione          Variazione
----------  -------------------  ----------
1           Carico per acquisto  1
2           Scarico per vendita  -1
3           Reso da cliente      1
4           Reso a fornitore     -1
5           Rettifica aumento a  1
6           Rettifica aumento v  -1
7           Rettifica diminuzio  1
8           Rettifica diminuzio  -1
9           Carico da produzion  1
10          Scarico a produzion  -1
11          Carico da altro mag  1
12          Scarico ad altro ma  -1
13          Saldo iniziale       1
```

Per ottenere il risultato stampato su carta, basta modificare leggermente il comando:

```
$ sqlite3 mag.db < prova-stampa-artico-caus.sql ↩
↪| lpr [Invio]
```

79.2.4 Verifica sull'interrogazione della relazione «Articoli»

Si prepari il file '`interr-artico-01.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple della relazione '**Articoli**', ordinando gli attributi in questo modo: '**Descrizione**', '**Articolo**', '**UM**', '**ScortaMin**' e '**Listino**'.

Figura 79.25. Scheletro del file '`interr-artico-01.sql`', da completare.

```
-- Interrogazione della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-artico-01.sql

.headers on
.mode column

SELECT …
```

Una volta completato e salvato il file '`interr-artico-01.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-artico-01.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Descrizione                    Articolo    UM          ScortaMin   Listino
-----------------------------  ----------  ----------  ----------  ----------
Dischetti da 9 cm 1440 Kibyte  1           pz          500         0.2
Dischetti da 9 cm 1440 Kibyte  2           pz          500         0.25
CD-R 16x                       101         pz          500         0.5
CD-R 52x                       102         pz          500         1
CD-RW 4x                       201         pz          200         1
CD-RW 8x                       202         pz          200         1.5
DVD-R 8x                       301         pz          200         1
DVD-R 16x                      302         pz          200         2
DVD+R 8x                       401         pz          200         1
DVD+R 16x                      402         pz          200         2
DVD-RW 8x                      501         pz          200         2
DVD+RW 8x                      601         pz          200         2
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-artico-01.sql | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interr-artico-01.sql`'.

79.2.5 Verifica sull'interrogazione delle relazioni «Fornitori» e «Clienti»

Si prepari il file '`interr-forn-clie-01.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple delle relazioni '**`Fornitori`**' e '**`Clienti`**', limitatamente agli attributi: '**`Fornitore`**' (nel caso della relazione '**`Fornitori`**') o '**`Cliente`**' (nel caso della relazione '**`Clienti`**'), '**`RagioneSociale`**', '**`Telefono`**' e '**`Fax`**'.

Figura 79.27. Scheletro del file '`interr-forn-clie-01.sql`', da completare.

```
-- Interrogazione delle relazioni "Fornitori" e "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-forn-clie-01.sql

.headers on
.mode column

SELECT …
SELECT …
```

Una volta completato e salvato il file '`interr-forn-clie-01.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-forn-clie-01.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Fornitore   RagioneSociale  Telefono     Fax
----------  --------------  -----------  -----------
1           Tizio Tizi      0422,111111  0422,222222
2           Caio Cai        0423,222222  0423,333333
3           Sempronio Semp  0422,333333  0422,444444
Cliente     RagioneSociale  Telefono     Fax
----------  --------------  -----------  -----------
1           Mevio Mevi      0422,444444  0422,555555
2           Filano Filani   0439,555555  0439,666666
3           Martino Martin  0438,666666  0438,777777
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-forn-clie-01.sql ↩
↪        | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interr-forn-clie-01.sql`'.

79.2.6 Conclusione

Il file '`prova-stampa-artico-caus.sql`', non serve più nelle sezioni successive, pertanto va eliminato.

79.3 Interrogazione ordinata di una relazione

Attraverso l'istruzione '**`SELECT`**', aggiungendo l'opzione '**`ORDERED BY`**', è possibile specificare gli attributi secondo i quali ordinare il risultato. In mancanza dell'indicazione di questa opzione, l'elenco delle tuple si ottiene secondo un ordine «casuale», che solitamente coincide con la sequenza di inserimento.

79.3.1 Interrogazione ordinata

A titolo di esempio, si vuole ottenere l'elenco delle tuple della relazione '**`Articoli`**', in ordine di descrizione. Si può utilizzare il programma '**`sqlite3`**' in modo interattivo:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions

sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM Articoli ↩
↪               ORDER BY Descrizione; [Invio]
```



```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
101         CD-R 16x     pz          0.5         500
102         CD-R 52x     pz          1           500
201         CD-RW 4x     pz          1           200
202         CD-RW 8x     pz          1.5         200
402         DVD+R 16x    pz          2           200
401         DVD+R 8x     pz          1           200
601         DVD+RW 8x    pz          2           200
302         DVD-R 16x    pz          2           200
301         DVD-R 8x     pz          1           200
501         DVD-RW 8x    pz          2           200
1           Dischetti d  pz          0.2         500
2           Dischetti d  pz          0.25        500
```

Con la relazione '**`Articoli`**', potrebbe essere interessante un ordinamento per listino, ma in questo caso si aggiunge anche la descrizione, quando il prezzo di listino risulta uguale:

```
sqlite> SELECT * FROM Articoli ORDER BY Listino, Descrizione;
[Invio]
```

```
Articolo    Descrizione                  UM          Listino     ScortaMin
----------  ---------------------------  ----------  ----------  ----------
1           Dischetti da 9 cm 1440 Kibyte  pz          0.2         500
2           Dischetti da 9 cm 1440 Kibyte  pz          0.25        500
101         CD-R 16x                     pz          0.5         500
102         CD-R 52x                     pz          1           500
201         CD-RW 4x                     pz          1           200
401         DVD+R 8x                     pz          1           200
301         DVD-R 8x                     pz          1           200
202         CD-RW 8x                     pz          1.5         200
402         DVD+R 16x                    pz          2           200
601         DVD+RW 8x                    pz          2           200
302         DVD-R 16x                    pz          2           200
501         DVD-RW 8x                    pz          2           200
```

```
sqlite> .quit [Invio]
```

Si osservi che l'ordinamento dipende dal tipo di informazione che l'attributo relativo può contenere. Per esempio, nel caso della relazione '**`Articoli`**', il riordino per descrizione avviene in modo lessicografico, mentre il riordino per listino avviene in base al valore numerico.

79.3.2 Verifica sull'interrogazione ordinata della relazione «Articoli»

Si prepari il file '`interr-artico-02.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple della relazione '**`Articoli`**', ordinate in base al livello di scorta minima e di descrizione; inoltre, si vogliono ottenere solo alcuni attributi, secondo la sequenza: '**`ScortaMin`**', '**`Descrizione`**', '**`Articolo`**'.

Figura 79.32. Scheletro del file '`interr-artico-02.sql`', da completare.

```
-- Interrogazione della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-artico-02.sql

.headers on
.mode column

SELECT …
```

Una volta completato e salvato il file '`interr-artico-02.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-artico-02.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
ScortaMin   Descrizione  Articolo
----------  -----------  ----------
200         CD-RW 4x     201
200         CD-RW 8x     202
200         DVD+R 16x    402
200         DVD+R 8x     401
200         DVD+RW 8x    601
200         DVD-R 16x    302
200         DVD-R 8x     301
200         DVD-RW 8x    501
```

```
500         CD-R 16x     101
500         CD-R 52x     102
500         Dischetti d  1
500         Dischetti d  2
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-artico-02.sql | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file ‘`interr-artico-02.sql`’.

79.3.3 Verifica sull’interrogazione ordinata della relazione «Clienti»

Si prepari il file ‘`interr-clie-01.sql`’, seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l’elenco di tutte le tuple della relazione ‘**`Clienti`**’, ordinate in base alla denominazione della ragione sociale, limitatamente agli attributi ‘**`Cliente`**’ e ‘**`RagioneSociale`**’.

Figura 79.34. Scheletro del file ‘`interr-clie-01.sql`’, da completare.

```
-- Interrogazione della relazione "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-clie-01.sql

.headers on
.mode column

SELECT …
```

Una volta completato e salvato il file ‘`interr-clie-01.sql`’, se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-clie-01.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Cliente     RagioneSociale
----------  --------------
2           Filano Filani
3           Martino Martin
1           Mevio Mevi
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-clie-01.sql | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file ‘`interr-clie-01.sql`’.

79.3.4 Verifica sull’interrogazione ordinata della relazione «Causali»

Si prepari il file ‘`interr-caus-01.sql`’, seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l’elenco di tutte le tuple della relazione ‘**`Causali`**’, ordinate in base al fatto che si tratti di movimenti in diminuzione o in aumento (l’attributo ‘**`Variazione`**’).

Figura 79.36. Scheletro del file ‘`interr-caus-01.sql`’, da completare.

```
-- Interrogazione della relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-caus-01.sql

.headers on
.mode column

SELECT …
```

Una volta completato e salvato il file ‘`interr-caus-01.sql`’, se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-clie-01.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Causale     Descrizione                Variazione
----------  -------------------------  ----------
12          Scarico ad altro magazzino  -1
10          Scarico a produzione        -1
8           Rettifica diminuzione acqu  -1
6           Rettifica aumento vendite   -1
4           Reso a fornitore            -1
2           Scarico per vendita         -1
13          Saldo iniziale              1
11          Carico da altro magazzino   1
9           Carico da produzione        1
7           Rettifica diminuzione vend  1
5           Rettifica aumento acquisto  1
3           Reso da cliente             1
1           Carico per acquisto         1
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-caus-01.sql | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file ‘`interr-caus-01.sql`’.

79.4 Interrogazione selettiva di una relazione

Attraverso l’istruzione ‘**`SELECT`**’, aggiungendo l’opzione ‘**`WHERE`**’, è possibile specificare una condizione per la selezione delle tuple desiderate. In mancanza dell’indicazione di questa opzione, l’elenco delle tuple è sempre completo. La parola chiave ‘**`WHERE`**’ precede un’espressione logica, che viene valutata per ogni tupla: se l’espressione risulta valida (*Vero*), allora la tupla viene presa in considerazione.

In queste lezioni non viene descritto in modo dettagliato come scrivere delle espressioni logiche; tuttavia, vengono raccolte qui delle tabelle riassuntive per la loro realizzazione. Possono essere usate in modo intuitivo, ma nelle verifiche non si richiede altro che utilizzare o modificare leggermente degli esempi già mostrati.

Tabella 79.38. Operatori di confronto.

Operatore e operandi	Descrizione
op1 `=` *op2*	*Vero* se gli operandi si equivalgono.
op1 `<>` *op2*	*Vero* se gli operandi sono differenti.
op1 `<` *op2*	*Vero* se il primo operando è minore del secondo.
op1 `>` *op2*	*Vero* se il primo operando è maggiore del secondo.
op1 `<=` *op2*	*Vero* se il primo operando è minore o uguale al secondo.
op1 `>=` *op2*	*Vero* se il primo operando è maggiore o uguale al secondo.

Tabella 79.39. Operatori logici.

Operatore e operandi	Descrizione
`NOT` *op*	Inverte il risultato logico dell’operando.
op1 `AND` *op2*	*Vero* se entrambi gli operandi restituiscono il valore *Vero*.
op1 `OR` *op2*	*Vero* se almeno uno degli operandi restituisce il valore *Vero*.

Tabella 79.40. Espressioni sulle stringhe di caratteri.

Espressioni e modelli	Descrizione
stringa `LIKE` *modello*	Restituisce *Vero* se il modello corrisponde alla stringa. Si osservi che SQLite non accetta la forma ‘**`IS LIKE`**’.
stringa `NOT LIKE` *modello*	Restituisce *Vero* se il modello non corrisponde alla stringa. Si osservi che SQLite non accetta la forma ‘**`IS NOT LIKE`**’.
`_`	Rappresenta un carattere qualsiasi.
`%`	Rappresenta una sequenza indeterminata di caratteri.

Tabella 79.41. Espressioni di verifica dei valori indeterminati.

Operatori	Descrizione
espressione `IS NULL`	Restituisce *Vero* se l'espressione genera un risultato indeterminato.
espressione `IS NOT NULL`	Restituisce *Vero* se l'espressione non genera un risultato indeterminato.

Tabella 79.42. Espressioni per la verifica dell'appartenenza di un valore a un intervallo o a un elenco.

Operatori e operandi	Descrizione
op1 `IN (` ***elenco*** `)`	*Vero* se il primo operando è contenuto nell'elenco.
op1 `NOT IN (` ***elenco*** `)`	*Vero* se il primo operando non è contenuto nell'elenco.
op1 `BETWEEN` ***op2*** `AND` ***op3***	*Vero* se il primo operando è compreso tra il secondo e il terzo.
op1 `NOT BETWEEN` ***op2*** `AND` ***op3***	*Vero* se il primo operando non è compreso nell'intervallo.

79.4.1 Interrogazione selettiva

A titolo di esempio, si vuole ottenere l'elenco delle tuple della relazione '**`Articoli`**', selezionando solo quelle che riportano un prezzo di listino maggiore o uguale a 1,00 €. Si può utilizzare il programma '**`sqlite3`**' in modo interattivo:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
sqlite> .headers on [Invio]
sqlite> .mode column [Invio]
sqlite> SELECT * FROM Articoli WHERE Listino >= 1; [Invio]

  Articolo    Descrizione  UM          Listino     ScortaMin
  ----------  -----------  ----------  ----------  ----------
  102         CD-R 52x     pz          1           500
  201         CD-RW 4x     pz          1           200
  202         CD-RW 8x     pz          1.5         200
  301         DVD-R 8x     pz          1           200
  302         DVD-R 16x    pz          2           200
  401         DVD+R 8x     pz          1           200
  402         DVD+R 16x    pz          2           200
  501         DVD-RW 8x    pz          2           200
  601         DVD+RW 8x    pz          2           200
```

La condizione di selezione potrebbe essere più articolata; per esempio si potrebbe decidere di selezionare gli articoli che hanno un prezzo di listino maggiore o uguale a 1,00 € e che hanno una descrizione che inizia con «DVD»:

```
sqlite> SELECT * FROM Articoli [Invio]
   ...>          WHERE Listino >= 1 [Invio]
   ...>            AND Descrizione LIKE 'DVD%'; [Invio]

  Articolo    Descrizione  UM          Listino     ScortaMin
  ----------  -----------  ----------  ----------  ----------
  301         DVD-R 8x     pz          1           200
  302         DVD-R 16x    pz          2           200
  401         DVD+R 8x     pz          1           200
  402         DVD+R 16x    pz          2           200
  501         DVD-RW 8x    pz          2           200
  601         DVD+RW 8x    pz          2           200
```

Come sempre, si conclude il funzionamento interattivo di '**`sqlite3`**' con il comando '**`.quit`**':

```
sqlite> .quit [Invio]
```

79.4.2 Verifica sull'interrogazione selettiva della relazione «Articoli»

Si prepari il file '`interr-artico-03.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco delle tuple della relazione '**`Articoli`**', corrispondenti a dei «DVD», che abbiano un prezzo minore o uguale a 1,00 € (l'operatore da usare per rappresentare il confronto «minore o uguale» è '**`<=`**').

Figura 79.46. Scheletro del file '`interr-artico-03.sql`', da completare.

```
-- Interrogazione della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-artico-03.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file '`interr-artico-03.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-artico-03.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
301         DVD-R 8x     pz          1           200
401         DVD+R 8x     pz          1           200
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-artico-03.sql | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interr-artico-03.sql`'.

79.4.3 Verifica sull'interrogazione selettiva e ordinata della relazione «Articoli»

Si prepari il file '`interr-artico-04.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco ordinato per livello di scorta minima delle tuple della relazione '**`Articoli`**', che corrispondono a dei «CD».

Figura 79.48. Scheletro del file '`interr-artico-04.sql`', da completare.

```
-- Interrogazione della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-artico-04.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
       ORDER BY …
```

Una volta completato e salvato il file '`interr-artico-04.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-artico-04.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
202         CD-RW 8x     pz          1.5         200
201         CD-RW 4x     pz          1           200
102         CD-R 52x     pz          1           500
101         CD-R 16x     pz          0.5         500
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-artico-04.sql | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interr-artico-04.sql`'.

79.4.4 Verifica sull'interrogazione selettiva della relazione «Causali»

Si prepari il file '`interr-caus-02.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco delle tuple della relazione '**`Causali`**' che comportano un aumento (contabile) della quantità di un articolo in magazzino. Le causali che rappresentano un aumento della quantità sono quelle che, nell'attributo '**`Variazione`**' hanno il valore 1 (ovvero +1); pertanto, per selezionare le tuple in questione, è sufficiente verificare che questo valore sia esattamente pari a uno (utilizzando l'operatore '**`=`**').

Figura 79.50. Scheletro del file '`interr-caus-02.sql`', da completare.

```
-- Interrogazione della relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-caus-02.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file '`interr-caus-02.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-caus-02.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Causale     Descrizione          Variazione
----------  -------------------  ----------
1           Carico per acquisto  1
3           Reso da cliente      1
5           Rettifica aumento a  1
7           Rettifica diminuzio  1
9           Carico da produzion  1
11          Carico da altro mag  1
13          Saldo iniziale       1
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-caus-02.sql | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interr-caus-02.sql`'.

79.5 Interrogazioni simultanee di più relazioni

Quando si realizzano delle relazioni, spesso si considerano dei collegamenti tra queste, per evitare di ripetere le stesse informazioni in relazioni differenti. La relazione '**`Movimenti`**', creata all'inizio di queste lezioni, contiene diversi attributi che, in pratica, fanno riferimento a tuple di altre relazioni.

Figura 79.52. La relazione '**`Movimenti`**', già apparsa nella figura 79.15.

Movimento	Articolo	Causale	Data	Cliente	Fornitore	Quantita	Valore
1	2	1	2012-01-15	NULL	3	10000	100,00
2	2	2	2012-01-16	2	NULL	1000	10,00
3	102	1	2012-01-17	NULL	2	1000	200,00
4	102	2	2012-01-18	1	NULL	100	20,00
5	401	1	2012-01-19	NULL	1	1000	200,00
6	401	2	2012-01-20	3	NULL	200	40,00
7	401	4	2012-01-20	NULL	1	100	20,00
8	102	4	2012-01-20	NULL	2	100	20,00
9	601	1	2012-01-21	NULL	3	2000	1000,00
10	601	2	2012-01-25	1	NULL	1000	500,00

Intuitivamente si comprende che i dati usati per creare il collegamento con un'altra relazione, devono essere sufficienti a individuare le tuple in modo univoco. Quindi, sulla base di questa univocità, si possono collegare effettivamente i dati attraverso delle interrogazioni che coinvolgono tutte le relazioni interessate, per generare un listato con le informazioni desiderate.

79.5.1 Interrogazione simultanea delle relazioni «Movimenti» e «Articoli»

Con l'ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file '`prova-interr-movi-arti.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Interrogazione delle relazioni "Movimenti" e "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-interr-movi-arti.sql

.headers on
.mode column

SELECT Movimenti.Data, Articoli.Descrizione,
       Movimenti.Causale, Movimenti.Quantita
       FROM Movimenti, Articoli
       WHERE Movimenti.Articolo = Articoli.Articolo;
```

Come si può vedere, per evitare ambiguità, i nomi degli attributi sono preceduti dal nome della relazione a cui appartengono, separati da un punto.

Si controlli di avere scritto il file '`prova-interr-movi-arti.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < prova-interr-movi-arti.sql [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Data        Descrizione                           Causale     Quantita
----------  ------------------------------------  ----------  ----------
2012-01-15  Dischetti da 9 cm 1440 Kibyte colorati  1          10000
2012-01-16  Dischetti da 9 cm 1440 Kibyte colorati  2          1000
2012-01-17  CD-R 52x                               1          1000
2012-01-18  CD-R 52x                               2          100
2012-01-19  DVD+R 8x                              1          1000
2012-01-20  DVD+R 8x                              2          200
2012-01-20  DVD+R 8x                              4          100
2012-01-20  CD-R 52x                             4          100
2012-01-21  DVD+RW 8x                            1          2000
2012-01-25  DVD+RW 8x                            2          1000
```

79.5.2 Interrogazione simultanea delle relazioni «Movimenti», «Articoli» e «Causali»

Si riprenda il file '`prova-interr-movi-arti.sql`' e lo si modifichi in modo da avere il contenuto seguente:

```
-- Interrogazione delle relazioni "Movimenti", "Articoli"
--   e "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-interr-movi-arti.sql

.headers on
.mode column

SELECT Movimenti.Data, Articoli.Descrizione,
       Causali.Descrizione
       FROM Movimenti, Articoli, Causali
       WHERE Movimenti.Articolo = Articoli.Articolo
         AND Movimenti.Causale = Causali.Causale;
```

Si controlli di avere modificato il file '`prova-interr-movi-arti.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < prova-interr-movi-arti.sql [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Data        Descrizione                           Descrizione
----------  ------------------------------------  -------------------
2012-01-15  Dischetti da 9 cm 1440 Kibyte colorati  Carico per acquisto
2012-01-16  Dischetti da 9 cm 1440 Kibyte colorati  Scarico per vendita
2012-01-17  CD-R 52x                               Carico per acquisto
2012-01-18  CD-R 52x                               Scarico per vendita
2012-01-19  DVD+R 8x                              Carico per acquisto
2012-01-20  DVD+R 8x                              Scarico per vendita
```

```
2012-01-20  DVD+R 8x                    Reso a fornitore
2012-01-20  CD-R 52x                    Reso a fornitore
2012-01-21  DVD+RW 8x                   Carico per acquisto
2012-01-25  DVD+RW 8x                   Scarico per vendita
```

79.5.3 Verifica sull'interrogazione simultanea delle relazioni «Movimenti» e «Causali»

Si prepari il file '`interr-movi-caus-01.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple per le quali si possa stabilire un abbinamento in base al codice della causale. Precisamente, si vuole ottenere l'attributo '**`Articolo`**' dalla relazione '**`Movimenti`**'; l'attributo '**`Descrizione`**' dalla relazione '**`Causali`**'; l'attributo '**`Data`**' dalla relazione '**`Movimenti`**'.

Figura 79.57. Scheletro del file '`interr-movi-caus-01.sql`', da completare.

```
-- Interrogazione delle relazioni "Movimenti" e "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-movi-caus-01.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file '`interr-movi-caus-01.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-movi-caus-01.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Descrizione          Data
----------  -------------------  ----------
2           Carico per acquisto  2012-01-15
2           Scarico per vendita  2012-01-16
102         Carico per acquisto  2012-01-17
102         Scarico per vendita  2012-01-18
401         Carico per acquisto  2012-01-19
401         Scarico per vendita  2012-01-20
401         Reso a fornitore     2012-01-20
102         Reso a fornitore     2012-01-20
601         Carico per acquisto  2012-01-21
601         Scarico per vendita  2012-01-25
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-movi-caus-01.sql | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interr-movi-caus-01.sql`'.

79.5.4 Verifica sull'interrogazione simultanea delle relazioni «Movimenti», «Causali» e «Clienti»

Si prepari il file '`interr-movi-caus-clienti-01.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple per le quali si possa stabilire un abbinamento in base al codice della causale e in base al codice del cliente. Precisamente, si vuole ottenere l'attributo '**`Articolo`**' dalla relazione '**`Movimenti`**'; l'attributo '**`Descrizione`**' dalla relazione '**`Causali`**'; l'attributo '**`Data`**' dalla relazione '**`Movimenti`**'; l'attributo '**`RagioneSociale`**' dalla relazione '**`Clienti`**'.

Figura 79.59. Scheletro del file '`interr-movi-caus-clienti-01.sql`', da completare.

```
-- Interrogazione delle relazioni "Movimenti", "Causali"
-- e "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-movi-caus-clienti-01.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file '`interr-movi-caus-clienti-01.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-movi-caus-clienti-01.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Descrizione          Data        RagioneSociale
----------  -------------------  ----------  --------------
2           Scarico per vendita  2012-01-16  Filano Filani
102         Scarico per vendita  2012-01-18  Mevio Mevi
401         Scarico per vendita  2012-01-20  Martino Martin
601         Scarico per vendita  2012-01-25  Mevio Mevi
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-movi-caus-clienti-01.sql ↩
↪       | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interr-movi-caus-clienti-01.sql`'.

79.5.5 Verifica sull'interrogazione ordinata e simultanea delle relazioni «Movimenti», «Causali» e «Clienti»

Si prepari il file '`interr-movi-caus-clienti-02.sql`', che deve avere gli stessi requisiti della verifica precedente, facendo in modo, però, che il risultato dell'interrogazione avvenga in modo ordinato, in base alla ragione sociale dei clienti.

Figura 79.61. Scheletro del file '`interr-movi-caus-clienti-02.sql`', da completare.

```
-- Interrogazione delle relazioni "Movimenti", "Causali"
-- e "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-movi-caus-clienti-02.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
       ORDER BY …
```

Una volta completato e salvato il file '`interr-movi-caus-clienti-02.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-movi-caus-clienti-02.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Descrizione          Data        RagioneSociale
----------  -------------------  ----------  --------------
2           Scarico per vendita  2012-01-16  Filano Filani
401         Scarico per vendita  2012-01-20  Martino Martin
601         Scarico per vendita  2012-01-25  Mevio Mevi
102         Scarico per vendita  2012-01-18  Mevio Mevi
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-movi-caus-clienti-02.sql ↩
↪       | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file ‘`interr-movi-caus-clienti-02.sql`’.

79.6 Interrogazioni simultanee di più relazioni e alias

Quando si interrogano simultaneamente più relazioni, può succedere che il risultato che si ottiene contenga degli attributi di relazioni differenti, ma con lo stesso nome, oppure potrebbe non essere abbastanza esplicito il suo contenuto. Nell’istruzione ‘**`SELECT`**’ con cui si esegue l’interrogazione, è possibile dichiarare dei nomi alternativi agli attributi, secondo le modalità descritte in questa sezione.

79.6.1 Interrogazione simultanea delle relazioni «Movimenti», «Articoli» e «Causali»

Si riprenda il file ‘`prova-interr-movi-arti.sql`’ e lo si modifichi in modo da avere il contenuto seguente:

```
-- Interrogazione delle relazioni "Movimenti", "Articoli"
-- e "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-interr-movi-arti.sql

.headers on
.mode column

SELECT Movimenti.Data,
       Articoli.Descrizione AS Articolo,
       Causali.Descrizione AS Causale
       FROM Movimenti, Articoli, Causali
       WHERE Movimenti.Articolo = Articoli.Articolo
         AND Movimenti.Causale = Causali.Causale;
```

Si controlli di avere modificato il file ‘`prova-interr-movi-arti.sql`’ in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < prova-interr-movi-arti.sql [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Data        Articolo                                Causale
----------  --------------------------------------  -------------------
2012-01-15  Dischetti da 9 cm 1440 Kibyte colorati  Carico per acquisto
2012-01-16  Dischetti da 9 cm 1440 Kibyte colorati  Scarico per vendita
2012-01-17  CD-R 52x                                Carico per acquisto
2012-01-18  CD-R 52x                                Scarico per vendita
2012-01-19  DVD+R 8x                                Carico per acquisto
2012-01-20  DVD+R 8x                                Scarico per vendita
2012-01-20  DVD+R 8x                                Reso a fornitore
2012-01-20  CD-R 52x                                Reso a fornitore
2012-01-21  DVD+RW 8x                               Carico per acquisto
2012-01-25  DVD+RW 8x                               Scarico per vendita
```

Come si può osservare, l’attributo ‘**`Descrizione`**’ della relazione ‘**`Articoli`**’ appare con il nome ‘**`Articolo`**’, mentre l’attributo ‘**`Descrizione`**’ della relazione ‘**`Causali`**’ appare con il nome ‘**`Causale`**’.

79.6.2 Verifica sull’interrogazione simultanea delle relazioni «Movimenti» e «Causali»

Si prepari il file ‘`interr-movi-caus-02.sql`’, seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l’elenco di tutte le tuple per le quali si possa stabilire un abbinamento in base al codice della causale. Precisamente, si vuole ottenere l’attributo ‘**`Articolo`**’ dalla relazione ‘**`Movimenti`**’; l’attributo ‘**`Descrizione`**’ dalla relazione ‘**`Causali`**’; l’attributo ‘**`Data`**’ dalla relazione ‘**`Movimenti`**’. Inoltre, si vuole che l’attributo ‘**`Descrizione`**’ della relazione ‘**`Causali`**’, appaia con il nome ‘**`Causale`**’.

Figura 79.65. Scheletro del file ‘`interr-movi-caus-02.sql`’, da completare.

```
-- Interrogazione delle relazioni "Movimenti" e "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-movi-caus-02.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file ‘`interr-movi-caus-02.sql`’, se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-movi-caus-02.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Causale              Data
----------  -------------------  ----------
2           Carico per acquisto  2012-01-15
2           Scarico per vendita  2012-01-16
102         Carico per acquisto  2012-01-17
102         Scarico per vendita  2012-01-18
401         Carico per acquisto  2012-01-19
401         Scarico per vendita  2012-01-20
401         Reso a fornitore     2012-01-20
102         Reso a fornitore     2012-01-20
601         Carico per acquisto  2012-01-21
601         Scarico per vendita  2012-01-25
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-movi-caus-02.sql | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file ‘`interr-movi-caus-02.sql`’.

79.6.3 Verifica sull’interrogazione simultanea delle relazioni «Movimenti», «Causali» e «Clienti»

Si prepari il file ‘`interr-movi-caus-clienti-03.sql`’, seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l’elenco di tutte le tuple per le quali si possa stabilire un abbinamento in base al codice della causale e in base al codice del cliente. Precisamente, si vuole ottenere l’attributo ‘**`Articolo`**’ dalla relazione ‘**`Movimenti`**’; l’attributo ‘**`Descrizione`**’ dalla relazione ‘**`Causali`**’; l’attributo ‘**`Data`**’ dalla relazione ‘**`Movimenti`**’; l’attributo ‘**`RagioneSociale`**’ dalla relazione ‘**`Clienti`**’. L’attributo ‘**`Descrizione`**’ della relazione ‘**`Causali`**’ deve apparire con il nome ‘**`Causale`**’ e l’attributo ‘**`RagioneSociale`**’ della relazione ‘**`Clienti`**’ deve apparire con il nome ‘**`Cliente`**’.

Figura 79.67. Scheletro del file ‘`interr-movi-caus-clienti-03.sql`’, da completare.

```
-- Interrogazione delle relazioni "Movimenti", "Causali"
-- e "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interr-movi-caus-clienti-03.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file ‘`interr-movi-caus-clienti-03.sql`’, se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < interr-movi-caus-clienti-03.sql [Invio]
```

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Causale              Data        Cliente
----------  -------------------  ----------  --------------
```

```
 2          Scarico per vendita  2012-01-16  Filano Filani
 102        Scarico per vendita  2012-01-18  Mevio Mevi
 401        Scarico per vendita  2012-01-20  Martino Martin
 601        Scarico per vendita  2012-01-25  Mevio Mevi
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interr-movi-caus-clienti-03.sql ↩
↪       | lpr [Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interr-movi-caus-clienti-03.sql`'.

79.6.4 Conclusione

Il file '`prova-interr-movi-arti.sql`' non serve più e va cancellato.

79.7 Viste

È possibile trasformare l'interrogazione di una o più relazioni in una ***vista***, la quale diventa in pratica una relazione virtuale.

79.7.1 Creazione della vista «Listino»

Con l'ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file '`prova-vista-listino.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione della vista "Listino"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-vista-listino.sql

CREATE VIEW Listino AS
            SELECT Articolo AS Codice,
                   Descrizione AS Articolo,
                   Listino AS EUR
                   FROM Articoli;
```

In questo modo, si crea la vista '**`Listino`**', composta dagli attributi '**`Codice`**', '**`Articolo`**' e '**`EUR`**', utilizzando, rispettivamente, gli attributi '**`Articolo`**', '**`Descrizione`**' e '**`Listino`**' dalla relazione '**`Articoli`**'.

Si controlli di avere scritto il file '`prova-vista-listino.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < prova-vista-listino.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la creazione della vista '**`Listino`**' ha avuto successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si deve prima cancellare la vista, quindi si può ritentare l'inserimento del comando (ammesso che il file '`prova-vista-listino.sql`' sia stato corretto di conseguenza). I passaggi per eliminare la vista, in modo interattivo, sono quelli seguenti:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions
sqlite> DROP VIEW Listino; [Invio]

sqlite> .quit [Invio]
```

Quando si è consapevoli di avere creato correttamente la vista '**`Listino`**', la si può interrogare come se fosse una relazione normale. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions
sqlite> .headers on [Invio]
```



```
sqlite> .mode column [Invio]

sqlite> SELECT * FROM Listino; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
 Codice      Articolo                      EUR
 ----------  ----------------------------  ----------
 1           Dischetti da 9 cm 1440 Kibyte  0.2
 2           Dischetti da 9 cm 1440 Kibyte  0.25
 101         CD-R 16x                      0.5
 102         CD-R 52x                      1
 201         CD-RW 4x                      1
 202         CD-RW 8x                      1.5
 301         DVD-R 8x                      1
 302         DVD-R 16x                     2
 401         DVD+R 8x                      1
 402         DVD+R 16x                     2
 501         DVD-RW 8x                     2
 601         DVD+RW 8x                     2
```

Come sempre, si conclude il funzionamento interattivo di '**`sqlite3`**' con il comando '**`.quit`**':

```
sqlite> .quit [Invio]
```

79.7.2 Creazione della vista «Resi»

Con l'ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file '`prova-vista-resi.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione della vista "Resi" (resi a fornitori)
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-vista-resi.sql

CREATE VIEW Resi AS
            SELECT Articoli.Descrizione     AS Articolo,
                   Movimenti.Data           AS Data,
                   Fornitori.RagioneSociale AS Fornitore,
                   Movimenti.Quantita       AS Reso,
                   Movimenti.Valore         AS Valore
                   FROM Articoli, Movimenti, Fornitori
                   WHERE Movimenti.Causale   = 4
                     AND Movimenti.Articolo
                         = Articoli.Articolo
                     AND Movimenti.Fornitore
                         = Fornitori.Fornitore;
```

In questo modo, si crea la vista '**`Resi`**', utilizzando le relazioni '**`Articoli`**', '**`Movimenti`**' e '**`Fornitori`**', limitando la selezione delle tuple della relazione '**`Movimenti`**' a quelle che riguardano un reso a fornitore, in quanto la causale corrisponde al numero quattro.

Si controlli di avere scritto il file '`prova-vista-resi.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < prova-vista-resi.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la creazione della vista '**`Resi`**' ha avuto successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si deve prima cancellare la vista, quindi si può ritentare l'inserimento del comando (ammesso che il file '`prova-vista-resi.sql`' sia stato corretto di conseguenza). I passaggi per eliminare la vista, in modo interattivo, sono quelli seguenti:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions
sqlite> DROP VIEW Resi; [Invio]

sqlite> .quit [Invio]
```

Quando si è consapevoli di avere creato correttamente la vista '**Resi**', la si può interrogare come se fosse una relazione normale. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions
sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM Resi; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
 Articolo    Data        Fornitore   Reso        Valore
 ----------  ----------  ----------  ----------  ----------
 CD-R 52x    2012-01-20  Caio Cai    100         20
 DVD+R 8x    2012-01-20  Tizio Tizi  100         20
```

Come sempre, si conclude il funzionamento interattivo di '**sqlite3**' con il comando '**.quit**':

```
sqlite> .quit [Invio]
```

79.7.3 Verifica sulla creazione della vista «Acquisti»

Si prepari il file '`vista-acquisti.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere un elenco dei movimenti di magazzino che riguardano i carichi per acquisto (causale uno). La vista deve essere composta dagli attributi seguenti:

1. '**Articolo**', corrispondente alla descrizione dell'articolo acquistato;
2. '**Data**', corrispondente alla data di acquisto;
3. '**Fornitore**', corrispondente alla ragione sociale del fornitore dal quale l'articolo è stato acquistato;
4. '**Acquistato**', corrispondente alla quantità acquistata;
5. '**Valore**', corrispondente al valore complessivo caricato (pari all'attributo con lo stesso nome della relazione '**Movimenti**').

Figura 79.77. Scheletro del file '`vista-acquisti.sql`', da completare.

```
-- Creazione della vista "Acquisti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-acquisti.sql

CREATE VIEW …
       SELECT …
              FROM …
              WHERE …
```

Una volta completato e salvato il file '`vista-acquisti.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < vista-acquisti.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la creazione della vista '**Acquisti**' ha avuto successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si deve prima cancellare la vista, quindi si può ritentare l'inserimento del comando (ammesso che il file '`vista-acquisti.sql`' sia stato corretto di conseguenza).

Quando si è consapevoli di avere creato correttamente la vista '**Acquisti**', la si può interrogare come se fosse una relazione normale. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions
sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM Acquisti; [Invio]
```


Si dovrebbe ottenere il listato seguente:

```
 Articolo                            Data        Fornitore          Acquistato  Valore
 ----------------------------------  ----------  -----------------  ----------  ----------
 Dischetti da 9 cm 1440 Kibyte colorati  2012-01-15  Sempronio Semproni  10000       100
 CD-R 52x                            2012-01-17  Caio Cai           1000        200
 DVD+R 8x                            2012-01-19  Tizio Tizi         1000        200
 DVD+RW 8x                           2012-01-21  Sempronio Semproni  2000        1000
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file '`vista-acquisti.sql`'.

79.7.4 Verifica sulla creazione della vista «Vendite»

Si prepari il file '`vista-vendite.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere un elenco dei movimenti di magazzino che riguardano gli scarichi per vendita (causale due). La vista deve essere composta dagli attributi seguenti:

1. '**Articolo**', corrispondente alla descrizione dell'articolo venduto;
2. '**Data**', corrispondente alla data di vendita;
3. '**Cliente**', corrispondente alla ragione sociale del cliente al quale l'articolo è stato venduto;
4. '**Venduto**', corrispondente alla quantità venduta;
5. '**Valore**', corrispondente al valore complessivo scaricato (pari all'attributo con lo stesso nome della relazione '**Movimenti**').

Figura 79.80. Scheletro del file '`vista-vendite.sql`', da completare.

```
-- Creazione della vista "Vendite"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-vendite.sql

CREATE VIEW …
       SELECT …
              FROM …
              WHERE …
```

Una volta completato e salvato il file '`vista-vendite.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < vista-vendite.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la creazione della vista '**Vendite**' ha avuto successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si deve prima cancellare la vista, quindi si può ritentare l'inserimento del comando (ammesso che il file '`vista-vendite.sql`' sia stato corretto di conseguenza).

Quando si è consapevoli di avere creato correttamente la vista '**Vendite**', la si può interrogare come se fosse una relazione normale. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions
sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM Vendite; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
 Articolo                            Data        Cliente        Venduto     Valore
 ----------------------------------  ----------  -------------  ----------  ----------
 Dischetti da 9 cm 1440 Kibyte colorati  2012-01-16  Filano Filani  1000        10
 CD-R 52x                            2012-01-18  Mevio Mevi     100         20
 DVD+R 8x                            2012-01-20  Martino Marti  200         20
 DVD+RW 8x                           2012-01-25  Mevio Mevi     1000        500
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file '`vista-vendite.sql`'.

79.7.5 Conclusione

Prima di proseguire, si deve riprendere il file ‘`magazzino.sql`’ e vi si devono aggiungere le istruzioni per la creazione delle viste ‘**`Acquisti`**’ e ‘**`Vendite`**’, come contenuto nei file ‘`vista-acquisti.sql`’ e ‘`vista-vendite.sql`’.

Una volta aggiornato il file ‘`magazzino.sql`’ come descritto, si deve cancellare il file ‘`mag.db`’ e ricreare a partire dalle istruzioni contenute nel file ‘`magazzino.sql`’:

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

Se vengono segnalati degli errori, occorre correggere il file ‘`magazzino.sql`’, cancellare nuovamente il file ‘`mag.db`’, quindi si deve ripetere l’operazione. La base di dati contenuta nel file ‘`mag.db`’, viene usata ancora e non si può proseguire se non si riesce a ricrearla correttamente.

In precedenza sono stati creati i file ‘`prova-vista-listino.sql`’ e ‘`prova-vista-resi.sql`’, che a questo punto non servono più e vanno cancellati.

79.8 Modifica del contenuto delle tuple

Una volta inserita una tupla in una relazione, si può modificare il suo contenuto con l’istruzione ‘**`UPDATE`**’, la quale si applica a tutte le tuple che soddisfano una certa condizione.

79.8.1 Modifica di una causale di magazzino

Con l’ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file ‘`prova-modifica-causali.sql`’, contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Modifica della relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-modifica-causali.sql

UPDATE Causali
       SET Descrizione = 'car x acq'
       WHERE Causale = 1;
```

In questo modo, si vuole modificare la tupla della relazione ‘**`Causali`**’, con il codice causale uno, in modo che la descrizione risulti molto più breve.

Si controlli di avere scritto il file ‘`prova-modifica-causali.sql`’ in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < prova-modifica-causali.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la modifica della tupla dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all’errore, dovrebbe essere sufficiente modificare il file ‘`prova-modifica-causali.sql`’ e riprovare.

Quando si è consapevoli di avere modificato correttamente la tupla in questione, si può interrogare la relazione per verificare i cambiamenti apportati. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
sqlite> .headers on [Invio]
sqlite> .mode column [Invio]
sqlite> SELECT * FROM Causali; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
  Causale     Descrizione  Variazione
  ----------  -----------  ----------
  1           car x acq    1
  2           Scarico per  -1
  3           Reso da cli  1
  4           Reso a forn  -1
  5           Rettifica a  1
  6           Rettifica a  -1
  7           Rettifica d  1
  8           Rettifica d  -1
  9           Carico da p  1
  10          Scarico a p  -1
  11          Carico da a  1
  12          Scarico ad   -1
  13          Saldo inizi  1
```

Come sempre, si conclude il funzionamento interattivo di ‘**`sqlite3`**’ con il comando ‘**`.quit`**’:

```
sqlite> .quit [Invio]
```

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file ‘`mag.db`’ e poi ricrearlo con il comando seguente:

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

79.8.2 Modifica di diverse causali di magazzino

Si riprenda il file ‘`prova-modifica-causali.sql`’ e lo si modifichi secondo la forma seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Modifica della relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-modifica-causali.sql

UPDATE Causali
       SET Descrizione = UPPER (Descrizione)
       WHERE Variazione = 1;

UPDATE Causali
       SET Descrizione = LOWER (Descrizione)
       WHERE Variazione = -1;
```

In questo modo, si vuole modificare ogni tupla della relazione ‘**`Causali`**’ che corrisponde a un aumento di quantità in magazzino (in quanto nell’attributo ‘**`Variazione`**’ ha il valore +1), in modo da avere una descrizione con tutte lettere maiuscole. Nel contempo, si vuole che le descrizione associate a movimenti in diminuzione, siano scritte utilizzando soltanto caratteri minuscoli.

Si controlli di avere scritto il file ‘`prova-modifica-causali.sql`’ in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < prova-modifica-causali.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la modifica delle tuple dovrebbe essere stata eseguita con successo, altrimenti, è stato commesso un errore. Per rimediare all’errore, dovrebbe essere sufficiente modificare il file ‘`prova-modifica-causali.sql`’ e riprovare.

Quando si è consapevoli di avere modificato correttamente le tuple, si può interrogare la relazione per verificare i cambiamenti apportati. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
sqlite> .headers on [Invio]
sqlite> .mode column [Invio]
sqlite> SELECT * FROM Causali; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Causale     Descrizione          Variazione
----------  -------------------  ----------
1           CARICO PER ACQUISTO  1
2           scarico per vendita  -1
3           RESO DA CLIENTE      1
4           reso a fornitore     -1
5           RETTIFICA AUMENTO A  1
6           rettifica aumento v  -1
7           RETTIFICA DIMINUZIO  1
8           rettifica diminuzio  -1
9           CARICO DA PRODUZION  1
10          scarico a produzion  -1
11          CARICO DA ALTRO MAG  1
12          scarico ad altro ma  -1
13          SALDO INIZIALE       1
```

Come sempre, si conclude il funzionamento interattivo di '**`sqlite3`**' con il comando '**`.quit`**':

```
sqlite> .quit [Invio]
```

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file '`mag.db`' e poi ricrearlo con il comando seguente:

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

79.8.3 Verifica sulla modifica della relazione «Articoli»

Si prepari il file '`modifica-articoli.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole cambiare la descrizione del primo e del secondo articolo, in modo da avere rispettivamente: «Floppy 1.4» e «Floppy 1.4 C». Per ottenere questo risultato è necessario utilizzare due volte l'istruzione '**`UPDATE`**'.

Figura 79.89. Scheletro del file '`modifica-articoli.sql`', da completare.

```
-- Modifica della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: modifica-articoli.sql

UPDATE Articoli
       SET …
       WHERE Articolo = 1;

UPDATE Articoli
       SET …
       WHERE Articolo = 2;
```

Una volta completato e salvato il file '`modifica-articoli.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < modifica-articoli.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la modifica delle tuple dovrebbe essere avvenuta con successo, altrimenti è stato commesso un errore. Per rimediare all'errore dovrebbe essere sufficiente correggere il file '`modifica-articoli.sql`' e riprovare. Quando si ritiene di avere eseguito l'operazione correttamente, si può interrogare la relazione '**`Articoli`**' per verificarne il risultato. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions
sqlite> .headers on [Invio]
sqlite> .mode column [Invio]
sqlite> SELECT * FROM Articoli; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
1           Floppy 1.4   pz          0.2         500
2           Floppy 1.4   pz          0.25        500
101         CD-R 16x     pz          0.5         500
102         CD-R 52x     pz          1           500
```



```
201         CD-RW 4x     pz          1           200
202         CD-RW 8x     pz          1.5         200
301         DVD-R 8x     pz          1           200
302         DVD-R 16x    pz          2           200
401         DVD+R 8x     pz          1           200
402         DVD+R 16x    pz          2           200
501         DVD-RW 8x    pz          2           200
601         DVD+RW 8x    pz          2           200
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file '`modifica-articoli.sql`'.

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file '`mag.db`' e poi ricrearlo con il comando seguente:

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

79.8.4 Verifica sulla modifica delle relazioni «Clienti» e «Fornitori»

Si prepari il file '`modifica-clienti-fornitori.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole cambiare la ragione sociale delle relazioni '**`Clienti`**' e '**`Fornitori`**', in modo che sia costituita da caratteri maiuscoli. Pertanto, la sostituzione riguarda tutte le tuple in entrambe le relazioni.

Figura 79.92. Scheletro del file '`modifica-clienti-fornitori.sql`', da completare.

```
-- Modifica delle relazioni "Clienti" e "Fornitori"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: modifica-clienti-fornitori.sql

UPDATE Clienti
       SET …

UPDATE Fornitori
       SET …
```

Una volta completato e salvato il file '`modifica-clienti-fornitori.sql`', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < modifica-clienti-fornitori.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la modifica delle tuple dovrebbe essere avvenuta con successo, altrimenti è stato commesso un errore. Per rimediare all'errore dovrebbe essere sufficiente correggere il file '`modifica-articoli.sql`' e riprovare. Quando si ritiene di avere eseguito l'operazione correttamente, si possono interrogare le due relazioni per verificarne il contenuto. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions
sqlite> .headers on [Invio]
sqlite> .mode column [Invio]
sqlite> SELECT * FROM Clienti; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Cliente RagioneSociale Paese  Indirizzo    CAP   Citta     Prov Telefono    Fax         CFPI
------- -------------- ------ ------------ ----- --------- ---- ----------- ----------- -----------
1       MEVIO VEVI     ITALIA via Mare, 11 31050 Morgano   TV   0422,444444 0422,555555 45678901234
2       FILANO FILANI  ITALIA via Farfalle 31032 Feltre    BL   0439,555555 0439,666666 56789012345
3       MARTINO MARTIN ITALIA via Marte, 3 31010 Mareno di TV   0438,666666 0438,777777 67890123456
```

```
sqlite> SELECT * FROM Fornitori; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Fornitore RagioneSociale Paese  Indirizzo     CAP   Citta      Prov Telefono    Fax         CFPI
--------- -------------- ------ ------------- ----- ---------- ---- ----------- ----------- -----------
1         TIZIO TIZI     ITALIA via Tazio, 11 31100 Treviso    TV   0422,111111 0422,222222 12345678901
2         CAIO CAI       ITALIA via Caino, 22 31033 Castelfran TV   0423,222222 0423,333333 23456789012
3         SEMPRONIO SEMP ITALIA via Salina, 3 31057 Silea      TV   0422,333333 0422,444444 34567890123
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file '`modifica-clienti-fornitori.sql`'.

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file ‘`mag.db`’ e poi ricrearlo con il comando seguente:

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

79.8.5 Conclusione

Il file ‘`prova-modifica-causali.sql`’ non serve più e va cancellato.

79.9 Eliminazione delle tuple

La cancellazione delle tuple avviene attraverso l’istruzione ‘**`DELETE FROM`**’, con un procedimento simile a quello della modifica, in quanto va specificata la condizione di cancellazione, altrimenti si ottiene l’eliminazione di tutte le tuple della relazione.

79.9.1 Cancellazione di una causale di magazzino

Con l’ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file ‘`prova-cancella-causali.sql`’, contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Cancellazione nella relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-cancella-causali.sql

DELETE FROM Causali
       WHERE Causale = 1;
```

In questo modo, si vuole eliminare la tupla della relazione ‘**`Causali`**’, con il codice causale uno (quella che ha la descrizione «Carico per acquisto»).

Si controlli di avere scritto il file ‘`prova-cancella-causali.sql`’ in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < prova-cancella-causali.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la cancellazione della tupla dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all’errore, dovrebbe essere sufficiente modificare il file ‘`prova-cancella-causali.sql`’ e riprovare. Quando si ritiene di avere cancellato la tupla in questione, si può interrogare la relazione per verificarne lo stato. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions
sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM Causali; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Causale     Descrizione          Variazione
----------  -------------------  ----------
2           Scarico per vendita  -1
3           Reso da cliente      1
4           Reso a fornitore     -1
5           Rettifica aumento a  1
6           Rettifica aumento v  -1
7           Rettifica diminuzio  1
8           Rettifica diminuzio  -1
9           Carico da produzion  1
10          Scarico a produzion  -1
11          Carico da altro mag  1
12          Scarico ad altro ma  -1
13          Saldo iniziale       1
```

Come sempre, si conclude il funzionamento interattivo di ‘**`sqlite3`**’ con il comando ‘**`.quit`**’:

```
sqlite> .quit [Invio]
```

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file ‘`mag.db`’ e poi ricrearlo con il comando seguente:

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

79.9.2 Cancellazione di diverse causali di magazzino

Si riprenda il file ‘`prova-cancella-causali.sql`’ e lo si modifichi secondo la forma seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Cancellazione nella relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-cancella-causali.sql

DELETE FROM Causali
       WHERE Variazione = -1;
```

In questo modo, si vogliono eliminare le tuple corrispondenti a una riduzione della quantità in magazzino, (in quanto nell’attributo ‘**`Variazione`**’ ha il valore `-1`).

Si controlli di avere scritto il file ‘`prova-cancella-causali.sql`’ in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < prova-cancella-causali.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la cancellazione dovrebbe avere avuto successo, altrimenti, è stato commesso un errore. Per rimediare all’errore, dovrebbe essere sufficiente modificare il file ‘`prova-cancella-causali.sql`’ e riprovare. Quando si ritiene di avere eseguito l’operazione con successo, si può interrogare la relazione per verificare i cambiamenti apportati. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions
sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM Causali; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Causale     Descrizione          Variazione
----------  -------------------  ----------
1           Carico per acquisto  1
3           Reso da cliente      1
5           Rettifica aumento a  1
7           Rettifica diminuzio  1
9           Carico da produzion  1
11          Carico da altro mag  1
13          Saldo iniziale       1
```

Come sempre, si conclude il funzionamento interattivo di ‘**`sqlite3`**’ con il comando ‘**`.quit`**’:

```
sqlite> .quit [Invio]
```

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file ‘`mag.db`’ e poi ricrearlo con il comando seguente:

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

79.9.3 Verifica sulla cancellazione di alcuni articoli

Si prepari il file ‘`cancella-articoli.sql`’, seguendo lo scheletro seguente, tenendo conto che si vogliono eliminare i dischetti (i primi due).

Figura 79.102. Scheletro del file ‘`cancella-articoli.sql`’, da completare.

```
-- Cancellazione di alcune tuple della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: cancella-articoli.sql

DELETE FROM Articoli
       WHERE …

DELETE FROM Articoli
       WHERE …
```

Una volta completato e salvato il file ‘`cancella-articoli.sql`’, se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < cancella-articoli.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la cancellazione delle tuple dovrebbe essere avvenuta con successo, altrimenti è stato commesso un errore. Per rimediare all’errore dovrebbe essere sufficiente correggere il file ‘`modifica-articoli.sql`’ e riprovare. Quando si ritiene di avere eseguito l’operazione correttamente, si può interrogare la relazione ‘**`Articoli`**’ per verificarne il risultato. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM Articoli; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
101         CD-R 16x     pz          0.5         500
102         CD-R 52x     pz          1           500
201         CD-RW 4x     pz          1           200
202         CD-RW 8x     pz          1.5         200
301         DVD-R 8x     pz          1           200
302         DVD-R 16x    pz          2           200
401         DVD+R 8x     pz          1           200
402         DVD+R 16x    pz          2           200
501         DVD-RW 8x    pz          2           200
601         DVD+RW 8x    pz          2           200
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file ‘`cancella-articoli.sql`’.

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file ‘`mag.db`’ e poi ricrearlo con il comando seguente:

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

79.9.4 Conclusione

Il file ‘`prova-cancella-causali.sql`’ non serve più e va cancellato.

79.10 Grilletti per il controllo del dominio degli attributi

Nel momento in cui si inseriscono o si modificano i valori per una tupla di una certa relazione, può essere importante fare in modo di rifiutare i valori impossibili, in quanto non facenti parte del dominio previsto per gli attributi della stessa. Di solito, questo tipo di controllo può essere dichiarato in fase di creazione della relazione; tuttavia, un DBMS limitato potrebbe ignorare tali dichiarazioni.

I ***grilletti*** sono delle funzioni che «scattano», in quanto vengono eseguite, quando si verificano certi eventi. Attraverso i grilletti è possibile impedire l’inserimento di valori errati all’interno degli attributi e questo è l’obiettivo della sezione.

79.10.1 Creazione dei grilletti «Causali_ins» e «Causali_upd»

Con l’ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file ‘`grilletti-causali.sql`’, contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione dei grilletti "Causali_ins" e "Causali_upd"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-causali.sql

CREATE TRIGGER Causali_ins
               BEFORE INSERT ON Causali
               FOR EACH ROW
               BEGIN
                    SELECT CASE
                    WHEN (NEW.Variazione > 1)
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere superiore a 1!')
                    WHEN (NEW.Variazione < -1)
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere inferiore a -1!')
                    WHEN (NEW.Variazione = 0)
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere pari a 0!')
                    END;
                END;

CREATE TRIGGER Causali_upd
               BEFORE UPDATE ON Causali
               FOR EACH ROW
               BEGIN
                    SELECT CASE
                    WHEN (NEW.Variazione > 1)
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere superiore a 1!')
                    WHEN (NEW.Variazione < -1)
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere inferiore a -1!')
                    WHEN (NEW.Variazione = 0)
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere pari a 0!')
                    END;
                END;
```

In questo modo, si creano i grilletti ‘**`Causali_ins`**’ e ‘**`Causali_upd`**’, con lo scopo di avvisare in caso di inserimento di un valore impossibile nell’attributo ‘**`Variazione`**’ della relazione ‘**`Causali`**’ (sia nel caso di inserimento di una tupla nuova, sia quando si cerca di modificare quell’attributo in una tupla già esistente). Si osservi che, all’interno dei messaggi di errore, l’apostrofo è stato raddoppiato, per evitare che possa essere interpretato come la conclusione della stringa.

Si controlli di avere scritto il file ‘`grilletti-causali.sql`’ in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < grilletti-causali.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all’errore, si devono prima cancellare i grilletti, quindi si può ritentare l’inserimento del comando (ammesso che il file ‘`grilletti-causali.sql`’ sia stato corretto di conseguenza). I passaggi per eliminare i grilletti, in modo interattivo, sono quelli seguenti:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
sqlite> DROP TRIGGER Causali_ins; [Invio]

sqlite> DROP TRIGGER Causali_upd; [Invio]

sqlite> .quit [Invio]
```

Quando si ritiene di avere creato correttamente i grilletti, si può tentare l’inserimento o la modifica di tuple con valori errati nella relazione ‘**`Causali`**’, per verificare se queste vengono rifiutate come dovrebbero. Si proceda con i passaggi seguenti, utilizzando ‘**`sqlite3`**’ in modo interattivo:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
```

```
sqlite> INSERT INTO Causali VALUES (100, 'Doppio carico', +2);
[Invio]

  INSERT INTO Causali VALUES (100, 'Doppio carico', +2);
  SQL error: L'attributo "Variazione" non può essere ↩
 ↪superiore a 1!

sqlite> INSERT INTO Causali VALUES (101, 'Doppio scarico',
-2); [Invio]

  INSERT INTO Causali VALUES (101, 'Doppio scarico', -2);
  SQL error: L'attributo "Variazione" non può essere ↩
 ↪inferiore a -1!

sqlite> INSERT INTO Causali VALUES (102, 'Movimento nullo',
0); [Invio]

  INSERT INTO Causali VALUES (102, 'Movimento nullo', 0);
  SQL error: L'attributo "Variazione" non può essere pari a 0!

sqlite> UPDATE Causali SET Variazione = +2 WHERE Causale = 1;
[Invio]

  UPDATE Causali SET Variazione = +2 WHERE Causale = 1;
  SQL error: L'attributo "Variazione" non può essere ↩
 ↪superiore a 1!

sqlite> UPDATE Causali SET Variazione = -2 WHERE Causale = 2;
[Invio]

  UPDATE Causali SET Variazione = -2 WHERE Causale = 2;
  SQL error: L'attributo "Variazione" non può essere ↩
 ↪inferiore a -1!

sqlite> UPDATE Causali SET Variazione = 0 WHERE Causale = 3;
[Invio]

  UPDATE Causali SET Variazione = 0 WHERE Causale = 3;
  SQL error: L'attributo "Variazione" non può essere pari a 0!

sqlite> .quit [Invio]
```

79.10.2 Creazione del grilletto «Articoli_ins» e «Articoli_upd»

Con l'ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file '`grilletti-articoli.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione dei grilletti "Articoli_ins" e "Articoli_upd"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-articoli.sql

CREATE TRIGGER Articoli_ins
              BEFORE INSERT ON Articoli
              FOR EACH ROW
              BEGIN
                   SELECT CASE
                   WHEN (NEW.Listino <= 0)
                   THEN
                       RAISE (ABORT, 'Il prezzo non può essere inferiore o uguale a zero!')
                   WHEN (NEW.ScortaMin < 0)
                   THEN
                       RAISE (ABORT, 'La scorta minima non può essere inferiore a zero!')
                   END;
               END;

CREATE TRIGGER Articoli_upd
              BEFORE UPDATE ON Articoli
              FOR EACH ROW
              BEGIN
                   SELECT CASE
                   WHEN (NEW.Listino <= 0)
                   THEN
                       RAISE (ABORT, 'Il prezzo non può essere inferiore o uguale a zero!')
                   WHEN (NEW.ScortaMin < 0)
                   THEN
                       RAISE (ABORT, 'La scorta minima non può essere inferiore a zero!')
                   END;
               END;
```

In questo modo, si creano i grilletti '**`Articoli_ins`**' e '**`Articoli_upd`**', con lo scopo di impedire l'inserimento di valori impossibili per il prezzo di listino e per la scorta minima (sia con le istruzioni '**`INSERT`**', sia con '**`UPDATE`**').

Si controlli di avere scritto il file '`grilletti-articoli.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

```
$ sqlite3 mag.db < grilletti-articoli.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si devono prima cancellare i grilletti, quindi si può ritentare l'inserimento del comando (ammesso che il file '`grilletti-articoli.sql`' sia stato corretto di conseguenza). I passaggi per eliminare i grilletti, in modo interattivo, sono quelli seguenti:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions

sqlite> DROP TRIGGER Articoli_ins; [Invio]

sqlite> DROP TRIGGER Articoli_upd; [Invio]

sqlite> .quit [Invio]
```

Quando si ritiene di avere creato correttamente i grilletti in questione, si può tentare l'inserimento di tuple con valori errati nella relazione '**`Articoli`**', per verificare se queste vengono rifiutate come dovrebbero. Si proceda con i passaggi seguenti, utilizzando '**`sqlite3`**' in modo interattivo:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions

sqlite> INSERT INTO Articoli [Invio]

   ...>     VALUES (660, 'DVD gratis','pz', 0, 200); [Invio]

  INSERT INTO Articoli VALUES (660, 'DVD gratis','pz', 0, 200);
  SQL error: Il prezzo non può essere inferiore o uguale a zero!

sqlite> INSERT INTO Articoli [Invio]

   ...>     VALUES (661, 'DVD ti paghiamo noi','pz', -2.00,
200); [Invio]

  INSERT INTO Articoli VALUES (661, 'DVD ti paghiamo noi','pz', -2.00, 200);
  SQL error: Il prezzo non può essere inferiore o uguale a zero!

sqlite> INSERT INTO Articoli [Invio]

   ...>     VALUES (662, 'DVD virtuale','pz', 2.00, -200);
[Invio]

  INSERT INTO Articoli VALUES (662, 'DVD virtuale','pz', ↩
 ↪2.00, -200);
  SQL error: La scorta minima non può essere inferiore a zero!

sqlite> UPDATE Articoli SET Listino = 0 WHERE Articolo = 1;
[Invio]

  UPDATE Articoli SET Listino = 0 WHERE Articolo = 1;
  SQL error: Il prezzo non può essere inferiore o uguale a zero!

sqlite> UPDATE Articoli SET Listino = -2.00 WHERE Articolo =
2; [Invio]

  UPDATE Articoli SET Listino = -2.00 WHERE Articolo = 2;
  SQL error: Il prezzo non può essere inferiore o uguale a zero!

sqlite> UPDATE Articoli SET ScortaMin = -200 WHERE Articolo =
101; [Invio]

  UPDATE Articoli SET ScortaMin = -200 WHERE Articolo = 101;
  SQL error: La scorta minima non può essere inferiore a zero!

sqlite> .quit [Invio]
```

79.10.3 Verifica sulla creazione dei grilletti «Movimenti_ins» e «Movimenti_upd»

Si prepari il file '`grilletti-movimenti.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole impedire l'inserimento nella relazione '**`Movimenti`**' di quantità inferiori o uguali a zero e di valori inferiori a zero.

Figura 79.123. Scheletro del file ‘grilletto-movimenti.sql’, da completare.

```
-- Creazione dei grilletti "Movimenti_ins" e "Movimenti_upd"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-movimenti.sql

CREATE TRIGGER Movimenti_ins
               BEFORE INSERT …
               FOR EACH ROW
               BEGIN
                    …
                    …
                    …
               END;

CREATE TRIGGER Movimenti_upd
               BEFORE UPDATE …
               FOR EACH ROW
               BEGIN
                    …
                    …
                    …
               END;
```

Una volta completato e salvato il file ‘grilletti-movimenti’, se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < grilletti-movimenti.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all’errore, si devono prima cancellare i grilletti, quindi si può ritentare l’inserimento del comando (ammesso che il file ‘grilletti-movimenti.sql’ sia stato corretto di conseguenza).

Si consegni per la valutazione la stampa del file ‘grilletti-movimenti.sql’.

79.10.4 Conclusione

Prima di passare alla sezione successiva, si deve riprendere il file ‘magazzino.sql’ e vi si devono aggiungere le istruzioni per la creazione dei grilletti ‘**Causali_ins**’, ‘**Causali_upd**’, ‘**Articoli_ins**’, ‘**Articoli_upd**’, ‘**Movimenti_ins**’ e ‘**Movimenti_upd**’, come contenuto nei file ‘grilletti-causali.sql’, ‘grilletti-articoli.sql’ e ‘grilletti-movimenti.sql’.

Si osservi che la dichiarazione dei grilletti va collocata immediatamente dopo la creazione della relazione a cui fanno riferimento e immediatamente prima delle istruzioni che inseriscono delle tuple.

Una volta aggiornato il file ‘magazzino.sql’ come descritto, si deve cancellare il file ‘mag.db’ e ricreare a partire dalle istruzioni contenute nel file ‘magazzino.sql’:

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

Se vengono segnalati degli errori, occorre correggere il file ‘magazzino.sql’, cancellare nuovamente il file ‘mag.db’, quindi si deve ripetere l’operazione. La base di dati contenuta nel file ‘mag.db’, viene usata nella sezione successiva e non si può proseguire se non si riesce a ricrearla correttamente.

79.11 Grilletti per il controllo della validità esterna

Nel momento in cui si inseriscono, modificano o eliminano dei valori per una certa relazione, può essere importante fare in modo di rifiutare le azioni che non sono valide, in base al contenuto di altre relazioni. Di solito, questo tipo di controllo può essere dichiarato in fase di creazione della relazione; tuttavia, un DBMS limitato potrebbe ignorare tali dichiarazioni.

Qui si mostra l’uso dei grilletti per imporre dei vincoli di validità dipendenti dal contenuto di altre relazioni.

79.11.1 Controllo del codice articolo tra la relazione «Movimenti» e la relazione «Articoli»

In precedenza sono stati creati due grilletti, denominati ‘**Movimenti_ins**’ e ‘**Movimenti_upd**’, con lo scopo di impedire l’inserimento (o la modifica) di valori impossibili per la quantità e per il valore del movimento. Questi due grilletti vengono ripresi ed estesi, allo scopo di impedire che possano essere inseriti movimenti riferiti ad articoli inesistenti, in quanto non ancora dichiarati nella relazione ‘**Articoli**’; inoltre ne viene aggiunto un altro, per impedire che un articolo possa essere eliminato dalla relazione ‘**Articoli**’, se questo risulta essere ancora utilizzato nella relazione ‘**Movimenti**’.

Pertanto, si crei il file ‘grilletti-movimenti-articoli.sql’, contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione dei grilletti "Movimenti_ins", "Movimenti_upd" e "Articoli_del"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-movimenti-articoli.sql

CREATE TRIGGER Movimenti_ins
       BEFORE INSERT ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            END;
       END;

CREATE TRIGGER Movimenti_upd
       BEFORE UPDATE ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            END;
       END;

CREATE TRIGGER Articoli_del
       BEFORE DELETE ON Articoli
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN ((SELECT Articolo FROM Movimenti WHERE Articolo = OLD.Articolo) IS NOT NULL)
            THEN
                RAISE (ABORT, 'L''articolo non può essere rimosso, perché è utilizzato nella relazione Movimenti!')
            END;
       END;
```

Una volta completato e salvato il file ‘grilletti-movimenti-articoli’, se ne deve controllare il funzionamento con la base di dati, ma prima vanno rimossi i grilletti ‘**Movimenti_ins**’ e ‘**Movimenti_upd**’, che qui vengono ricreati. Basta eseguire i passaggi seguenti:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions

sqlite> DROP TRIGGER Articoli_ins; [Invio]

sqlite> DROP TRIGGER Articoli_upd; [Invio]

sqlite> .quit [Invio]
```

Quando i grilletti preesistenti sono stati rimossi, si può eseguire il file ‘grilletti-movimenti-articoli.sql’ nella base di dati:

```
$ sqlite3 mag.db < grilletti-movimenti-articoli.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all’errore, si devono prima cancellare i grilletti (questa volta sono tre: ‘**Movimenti_ins**’, ‘**Movimenti_upd**’ e ‘**Articoli_del**’), quindi si può ritentare l’inserimento del comando (ammesso che il file ‘grilletti-movimenti-articoli.sql’ sia stato corretto di conseguenza).

Per verificare che i vincoli dichiarati funzionino come previsto, si

può provare a inserire un movimento che fa riferimento a un articolo inesistente; quindi, si può provare a cancellare un articolo che risulta invece movimentato:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
sqlite> INSERT INTO Movimenti [Invio]
   ...>     VALUES (11, 777, 2, '2012-01-25', [Invio]
   ...>     1, NULL, 1000, 500.00); [Invio]

  INSERT INTO Movimenti VALUES (11, 777, 2, '2012-01-25', 1, NULL, 1000, 500.00);
  SQL error: Il codice articolo non è presente nella relazione Articoli!
sqlite> UPDATE Movimenti SET Articolo = 777 [Invio]
   ...>     WHERE Movimento = 2; [Invio]

  UPDATE Movimenti SET Articolo = 777 WHERE Movimento = 2;
  SQL error: Il codice articolo non è presente nella ↩
↪relazione Articoli!
sqlite> DELETE FROM Articoli WHERE Articolo = 2; [Invio]

  DELETE FROM Articoli WHERE Articolo = 2;
  SQL error: L'articolo non può essere rimosso, ↩
↪perché è utilizzato nella relazione Movimenti!
sqlite> .quit [Invio]
```

79.11.2 Controllo del codice cliente tra la relazione «Movimenti» e la relazione «Clienti»

Vengono qui ripresi i grilletti '**Movimenti_ins**' e '**Movimenti_upd**', aggiungendo il grilletto '**Clienti_del**', con lo scopo di impedire che possano essere inseriti movimenti riferiti a clienti inesistenti (in quanto non ancora dichiarati nella relazione '**Clienti**') e di impedire la cancellazione di un cliente quando questo risulta essere ancora utilizzato nella relazione '**Movimenti**'.

Pertanto, si crei il file '`grilletti-movimenti-clienti.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione dei grilletti "Movimenti_ins", "Movimenti_upd" e "Clienti_del"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-movimenti-clienti.sql

CREATE TRIGGER Movimenti_ins
      BEFORE INSERT ON Movimenti
      FOR EACH ROW
      BEGIN
          SELECT CASE
          WHEN (NEW.Quantita <= 0)
          THEN
              RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
          WHEN (NEW.Valore < 0)
          THEN
              RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
          WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
          THEN
              RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
          WHEN ((NEW.Cliente IS NOT NULL)
                AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
          THEN
              RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
          END;
      END;

CREATE TRIGGER Movimenti_upd
      BEFORE UPDATE ON Movimenti
      FOR EACH ROW
      BEGIN
          SELECT CASE
          WHEN (NEW.Quantita <= 0)
          THEN
              RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
          WHEN (NEW.Valore < 0)
          THEN
              RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
          WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
          THEN
              RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
          WHEN ((NEW.Cliente IS NOT NULL)
                AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
          THEN
              RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
          END;
      END;

CREATE TRIGGER Clienti_del
      BEFORE DELETE ON Clienti
      FOR EACH ROW
      BEGIN
          SELECT CASE
          WHEN ((SELECT Cliente FROM Movimenti WHERE Cliente = OLD.Cliente) IS NOT NULL)
          THEN
              RAISE (ABORT, 'Il cliente non può essere rimosso, perché è utilizzato nella relazione Movimenti!')
          END;
      END;
```


A differenza dell'esempio che appare nella sezione precedente, l'attributo '**Cliente**' della relazione '**Movimenti**' può contenere il valore nullo ('**NULL**'). Per questa ragione, il grilletto verifica prima che il valore inserito non sia nullo, poi che il codice cliente esista nella relazione '**Clienti**'.

Una volta completato e salvato il file '`grilletti-movimenti-clienti`', se ne deve controllare il funzionamento con la base di dati, ma prima vanno rimossi i grilletti '**Movimenti_ins**' e '**Movimenti_upd**', che qui vengono ricreati. Basta eseguire i passaggi seguenti:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
sqlite> DROP TRIGGER Articoli_ins; [Invio]
sqlite> DROP TRIGGER Articoli_upd; [Invio]
sqlite> .quit [Invio]
```

Quando i grilletti preesistenti, associati alla relazione '**Movimenti**', sono stati rimossi, si può eseguire il file '`grilletti-movimenti-clienti.sql`' nella base di dati:

```
$ sqlite3 mag.db < grilletti-movimenti-clienti.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si devono prima cancellare i grilletti (questa volta sono tre: '**Movimenti_ins**', '**Movimenti_upd**' e '**Clienti_del**'), quindi si può ritentare l'inserimento del comando (ammesso che il file '`grilletti-movimenti-clienti.sql`' sia stato corretto di conseguenza).

Per verificare che i vincoli dichiarati funzionino come previsto, si può provare a inserire un movimento che fa riferimento a un cliente inesistente; quindi, si può provare a cancellare un articolo che risulta invece movimentato:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
sqlite> INSERT INTO Movimenti [Invio]
   ...>     VALUES (11, 101, 2, '2012-01-25', [Invio]
   ...>     999, NULL, 1000, 500.00); [Invio]

  INSERT INTO Movimenti VALUES (11, 101, 2, '2012-01-25', ↩
↪999, NULL, 1000, 500.00);
  SQL error: Il codice cliente non è presente nella ↩
↪relazione Clienti!
sqlite> UPDATE Movimenti SET Cliente = 999 WHERE Movimento =
2; [Invio]

  UPDATE Movimenti SET Cliente = 999 WHERE Movimento = 2;
  SQL error: Il codice cliente non è presente nella ↩
↪relazione Clienti!
sqlite> DELETE FROM Clienti WHERE Cliente = 2; [Invio]

  DELETE FROM Clienti WHERE Cliente = 2;
  SQL error: Il cliente non può essere rimosso, perché ↩
↪è utilizzato nella relazione Movimenti!
sqlite> .quit [Invio]
```

79.11.3 Verifica sulla creazione dei grilletti «Movimenti_ins», «Movimenti_upd» e «Causali_del»

Si prepari il file '`grilletti-movimenti-causali.sql`', modificando il file '`grilletti-movimenti-clienti.sql`', in modo da riutilizzare quanto già scritto nei grilletti '**Movimenti_ins**' e '**Movimenti_upd**'. Si segua lo scheletro seguente, tenendo conto che si vuole impedire l'inserimento nella relazione '**Movimenti**' di causali inesistenti e che si vuole impedire la cancellazione di una

causale, dalla relazione '**Causali**', se questa risulta utilizzata nella relazione '**Movimenti**' (in pratica, per questa funzione ulteriore, si deve aggiungere il grilletto '**Causali_del**').

Figura 79.136. Scheletro del file '`grilletto-movimenti-causali.sql`', da completare.

```
-- Creazione dei grilletti "Movimenti_ins", "Movimenti_upd" e "Causali_del"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-movimenti-causali.sql

CREATE TRIGGER Movimenti_ins
       BEFORE INSERT ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            WHEN ((NEW.Cliente IS NOT NULL)
                  AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
            THEN
                RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
            WHEN …
            THEN
                …
            END;
       END;

CREATE TRIGGER Movimenti_upd
       BEFORE UPDATE ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            WHEN ((NEW.Cliente IS NOT NULL)
                  AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
            THEN
                RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
            WHEN …
            THEN
                …
            END;
       END;

CREATE TRIGGER Causali_del
       BEFORE DELETE ON Causali
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN …
            THEN
                …
            END;
       END;
```

Una volta completato e salvato il file '`grilletti-movimenti-causali`', se ne deve controllare il funzionamento con la base di dati, ma prima vanno rimossi i grilletti '**Movimenti_ins**' e '**Movimenti_upd**', che qui vengono ricreati. Basta eseguire i passaggi seguenti:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
sqlite> DROP TRIGGER Articoli_ins; [Invio]

sqlite> DROP TRIGGER Articoli_upd; [Invio]

sqlite> .quit [Invio]
```

Quando i grilletti preesistenti, associati alla relazione '**Movimenti**', sono stati rimossi, si può eseguire il file '`grilletti-movimenti-causali.sql`' nella base di dati:

```
$ sqlite3 mag.db < grilletti-movimenti-causali.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si devono prima cancellare i grilletti (tutti), quindi si può ritentare l'inserimento del comando (ammesso che il file '`grilletti-movimenti-causali.sql`' sia stato corretto di conseguenza).

Si consegni per la valutazione la stampa del file '`grilletti-movimenti-causali.sql`'.

79.11.4 Verifica sulla creazione dei grilletti «Movimenti_ins», «Movimenti_upd» e «Fornitori_del»

Si prepari il file '`grilletti-movimenti-fornitori.sql`', modificando il file '`grilletti-movimenti-causali.sql`', in modo da riutilizzare quanto già scritto nei grilletti '**Movimenti_ins**' e '**Movimenti_upd**'. Si segua lo scheletro seguente, tenendo conto che si vuole impedire l'inserimento nella relazione '**Movimenti**' di fornitori inesistenti e che si vuole impedire la cancellazione di un fornitore, dalla relazione '**Fornitori**', se questo risulta utilizzato nella relazione '**Movimenti**' (in pratica, per questa funzione ulteriore, si deve aggiungere il grilletto '**Fornitori_del**').

Si osservi che nella relazione '**Movimenti**', l'attributo '**Fornitore**' può avere un valore nullo.

Figura 79.138. Scheletro del file '`grilletto-movimenti-fornitori.sql`', da completare.

```
-- Creazione dei grilletti "Movimenti_ins", "Movimenti_upd" e "Fornitori_del"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-movimenti-fornitori.sql

CREATE TRIGGER Movimenti_ins
       BEFORE INSERT ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            WHEN ((NEW.Cliente IS NOT NULL)
                  AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
            THEN
                RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
            WHEN …
            THEN
                …
            WHEN …
                 AND …
            THEN
                …
            END;
       END;

CREATE TRIGGER Movimenti_upd
       BEFORE UPDATE ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            WHEN ((NEW.Cliente IS NOT NULL)
                  AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
            THEN
                RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
            WHEN …
            THEN
                …
            WHEN …
                 AND …
            THEN
                …
            END;
       END;

CREATE TRIGGER Fornitori_del
       BEFORE DELETE ON Fornitori
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN …
            THEN
                …
            END;
       END;
```

Una volta completato e salvato il file '`grilletti-movimenti-fornitori`', se ne deve controllare il funzionamento con la base di dati, ma prima vanno rimossi i grilletti '**Movimenti_ins**' e '**Movimenti_upd**', che qui vengono ricreati. Basta eseguire i passaggi seguenti:

```
$ sqlite3 mag.db [Invio]
```

```
SQLite version …
Enter ".help" for instructions
sqlite> DROP TRIGGER Articoli_ins; [Invio]

sqlite> DROP TRIGGER Articoli_upd; [Invio]

sqlite> .quit [Invio]
```

Quando i grilletti preesistenti, associati alla relazione '**Movimenti**', sono stati rimossi, si può eseguire il file '`grilletti-movimenti-fornitori.sql`' nella base di dati:

```
$ sqlite3 mag.db < grilletti-movimenti-fornitori.sql [Invio]
```

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si devono prima cancellare i grilletti (tutti), quindi si può ritentare l'inserimento del comando (ammesso che il file '`grilletti-movimenti-fornitori.sql`' sia stato corretto di conseguenza).

Si consegni per la valutazione la stampa del file '`grilletti-movimenti-fornitori.sql`'.

79.11.5 Conclusione

Prima di passare alla sezione successiva, si deve riprendere il file '`magazzino.sql`' e vi si devono sostituire le istruzioni per la creazione dei grilletti '**Movimenti_ins**' e '**Movimenti_upd**', come contenuto nel file '`grilletti-movimenti-fornitori.sql`'; inoltre vanno aggiunti i grilletti '**Articoli_del**', '**Causali_del**', '**Clienti_del**' e '**Fornitori_del**', come sono stati realizzati in questa sezione.

Si osservi che la dichiarazione dei grilletti va collocata dopo la creazione della relazione a cui fanno riferimento e prima delle istruzioni che inseriscono delle tuple nella stessa relazione.

Una volta aggiornato il file '`magazzino.sql`' come descritto, si deve cancellare il file '`mag.db`' e ricreare a partire dalle istruzioni contenute nel file '`magazzino.sql`':

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

Se vengono segnalati degli errori, occorre correggere il file '`magazzino.sql`', cancellare nuovamente il file '`mag.db`', quindi si deve ripetere l'operazione. La base di dati contenuta nel file '`mag.db`', viene usata ancora e non si può proseguire se non si riesce a ricrearla correttamente.

79.12 Selezione di attributi virtuali, ottenuti da un'espressione

Il linguaggio SQL consente di costruire delle espressioni elementari, attraverso operatori matematici e funzioni comuni; l'interrogazione di una relazione può essere realizzata anche attraverso l'uso di espressioni.

Tabella 79.140. Operatori aritmetici comuni.

Operatore e operandi	Descrizione
`-`***op***	Inverte il segno dell'operando.
op1 `+` ***op2***	Somma i due operandi.
op1 `-` ***op2***	Sottrae dal primo il secondo operando.
op1 `*` ***op2***	Moltiplica i due operandi.
op1 `/` ***op2***	Divide il primo operando per il secondo.
op1 `%` ***op2***	Modulo: il resto della divisione tra il primo e il secondo operando.

Tabella 79.141. Alcune funzioni riconosciute da SQLite.

Funzione	Descrizione
`ABS(`***n***`)`	Restituisce il valore assoluto di ***n***
`LENGTH(`***stringa***`)`	Restituisce la lunghezza in caratteri della stringa indicata come argomento.
`LOWER(`***stringa***`)` `UPPER(`***stringa***`)`	La prima funzione restituisce la stringa indicata come argomento, con lettere minuscole; la seconda con lettere maiuscole.
`MIN(`***x***`,`***y***`[,…])` `MAX(`***x***`,`***y***`[,…])`	La prima funzione restituisce il valore minimo tra quelli indicati come argomento; la seconda, invece, restituisce il valore massimo.
`ROUND(`***n***`[,`***n***`])`	Restituisce il valore di ***n*** arrotondato a ***m*** decimali. Se ***m*** viene omesso, si intende pari a zero.
`SUBSTR(`***stringa***`,`***n***`,`***m***`)`	Estrae la stringa che inizia dalla posizione ***n***, lunga ***m*** caratteri.

79.12.1 Interrogazione della relazione «Movimenti» in modo da ottenere il valore unitario

Nella relazione '**Movimenti**' appare un attributo denominato '**Valore**'. Si tratta del valore dell'articolo, determinato in base al **costo di acquisto** (da non confondere con il prezzo di listino), con il quale si determina il valore delle merci in magazzino. Pre ogni tupla della relazione, si vuole ottenere il valore unitario, che si calcola dividendo il valore per la quantità movimentata corrispondente.

Si crei il file '`prova-interrogazione-movimenti-vu.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Interrogazione della relazione "Movimenti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-interrogazione-movimenti-vu.sql

.mode columns
.headers on

SELECT Articolo,
       Causale,
       Data,
       Quantita,
       (Valore/Quantita) AS ValoreUnitario
       FROM Movimenti;
```

Si osservi che, nell'ultima colonna del listato che si vuole ottenere, viene indicata l'espressione '**(Valore/Quantita)**', associata a un alias, in modo da mostrare una descrizione appropriata.

Una volta completato e salvato il file '`prova-interrogazione-movimenti-vu.sql`', se ne deve controllare il funzionamento con la base di dati:

```
$ sqlite3 mag.db < prova-interrogazione-movimenti-vu.sql [Invio]
```

Si dovrebbe ottenere un listato simile a quello seguente:

```
Articolo    Causale     Data        Quantita    ValoreUnitario
----------  ----------  ----------  ----------  --------------
2           1           2012-01-15  10000       0.01
2           2           2012-01-16  1000        0.01
102         1           2012-01-17  1000        0.2
102         2           2012-01-18  100         0.2
401         1           2012-01-19  1000        0.2
401         2           2012-01-20  200         0.2
401         4           2012-01-20  100         0.2
102         4           2012-01-20  100         0.2
601         1           2012-01-21  2000        0.5
601         2           2012-01-25  1000        0.5
```

Se invece si ottengono degli errori, dovrebbe essere sufficiente correggere il file '`prova-interrogazione-movimenti-vu.sql`' e poi riprovare.

79.12.2 Vista della relazione «Movimenti» in modo da ottenere il valore unitario

Così come è possibile scrivere un'interrogazione a una relazione indicando delle espressioni, se ne può realizzare una vista, così da semplificare gli accessi a queste informazioni generate attraverso dei calcoli.

Si crei il file '`vista-movimenti-extra.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Vista "MovimentiExtra"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-movimenti-extra.sql

CREATE VIEW MovimentiExtra AS
    SELECT Movimento,
           Articolo,
           Causale,
           Data,
           Cliente,
           Fornitore,
           Quantita,
           Valore,
           (Valore/Quantita) AS ValoreUnitario
           FROM Movimenti;
```

La vista '**`MovimentiExtra`**' che si ottiene in questo modo, include tutti gli attributi della relazione '**`Movimenti`**', aggiungendo l'attributo virtuale '**`ValoreUnitario`**', ottenuto dividendo il valore complessivo per la quantità movimentata.

Una volta completato e salvato il file '`vista-movimenti-extra.sql`', se ne deve controllare il funzionamento con la base di dati:

```
$ sqlite3 mag.db < vista-movimenti-extra.sql [Invio]
```

Se non vengono generati dei messaggi, l'operazione dovrebbe essere stata completata con successo, altrimenti, se la vista è stata creata, ma in modo errato, è necessario eliminarla, quindi si può correggere il file '`vista-movimenti-extra.sql`' e riprovare. Per eliminare la vista creata in modo errato, si può utilizzare il programma '**`sqlite3`**' in modo interattivo, come già mostrato in altri capitoli (istruzione '**`DROP VIEW`**').

Quando si ritiene di avere creato la vista in modo corretto, è bene verificare di avere ottenuto il risultato desiderato:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions
sqlite> .headers on [Invio]
sqlite> .mode columns [Invio]
sqlite> SELECT * FROM MovimentiExtra [Invio]

 Movimento Articolo Causale Data       Cliente Fornitore Quantita Valore ValoreUnitario
 --------- -------- ------- ---------- ------- --------- -------- ------ --------------
 1         2        1       2012-01-15         3         10000    100    0.01
 2         2        2       2012-01-16 2                 1000     10     0.01
 3         102      1       2012-01-17         2         1000     200    0.2
 4         102      2       2012-01-18 1                 100      20     0.2
 5         401      1       2012-01-19         1         1000     200    0.2
 6         401      2       2012-01-20 3                 200      20     0.2
 7         401      4       2012-01-20         1         100      20     0.2
 8         102      4       2012-01-20         2         100      20     0.2
 9         601      1       2012-01-21         3         2000     1000   0.5
 10        601      2       2012-01-25 1                 1000     500    0.5

sqlite> .quit [Invio]
```

79.12.3 Verifica sulla creazione della vista «MovimentiExtra»

In questa verifica si deve riprendere il file '`vista-movimenti-extra.sql`', per modificarlo, in modo da aggiungere un attributo virtuale ulteriore, contenente la quantità in forma algebrica: valori positivi per i carichi e valori negativi per gli scarichi. Dal momento che l'informazione se trattasi di carico o scarico è contenuta nella relazione '**`Causali`**', anche questa va utilizzata nella costruzione della vista.

Si modifichi il file '`vista-movimenti-extra.sql`', seguendo lo scheletro che viene proposto, per far sì che la vista '**`MovimentiExtra`**' contenga gli attributi seguenti:

1. '**`Movimento`**', corrispondente al numero di sequenza assegnato a ogni movimento nella relazione '**`Movimenti`**';
2. '**`Articolo`**', corrispondente al codice articolo della relazione '**`Movimenti`**';
3. '**`Causale`**', corrispondente al codice causale della relazione '**`Movimenti`**';
4. '**`Data`**', corrispondente alla data del movimento nella relazione '**`Movimenti`**';
5. '**`Cliente`**', corrispondente al codice cliente della relazione '**`Movimenti`**';
6. '**`Fornitore`**', corrispondente al codice fornitore della relazione '**`Movimenti`**';
7. '**`Quantità`**', corrispondente alla quantità movimentata nella relazione '**`Movimenti`**';
8. '**`Valore`**', corrispondente al valore del movimento, nella relazione '**`Movimenti`**';
9. '**`ValoreUnitario`**', corrispondente al valore unitario del movimento, ottenuto dividendo il valore complessivo per la quantità (dalla relazione '**`Movimenti`**');
10. '**`QuantitaAlgebrica`**', corrispondente alla quantità movimentata, con segno, ottenuta moltiplicando l'attributo '**`Variazione`**' della relazione '**`Causali`**' all'attributo '**`Quantita`**' della relazione '**`Movimenti`**'.

Figura 79.147. Scheletro del file '`vista-movimenti-extra.sql`', da completare.

```
-- Vista "MovimentiExtra"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-movimenti-extra.sql

CREATE VIEW MovimentiExtra AS
    SELECT Movimenti.Movimento          AS Movimento,
           Movimenti.Articolo           AS Articolo,
           Movimenti.Causale            AS Causale,
           Movimenti.Data               AS Data,
           Movimenti.Cliente            AS Cliente,
           Movimenti.Fornitore          AS Fornitore,
           Movimenti.Quantita           AS Quantita,
           Movimenti.Valore             AS Valore,
           (Movimenti.Valore/Movimenti.Quantita)
                                        AS ValoreUnitario,
           …
           FROM …
           WHERE Movimenti.Causale = Causali.causale;
```

Prima di poter eseguire questo file con la base di dati, occorre eliminare la vista '**`MovimentiExtra`**', che già dovrebbe esistere. Si ricorda che per eliminare una vista si utilizza l'istruzione '**`DROP VIEW`**' e che conviene intervenire con il programma '**`sqlite3`**' in modo interattivo.

Per eseguire il file '`vista-movimenti-extra.sql`', si agisce come sempre:

```
$ sqlite3 mag.db < vista-movimenti-extra.sql [Invio]
```

Se la creazione della vista produce degli errori, occorre eliminare nuovamente la vista e, dopo la correzione del file '`vista-movimenti-extra.sql`', si può ritentare.

Quando si è consapevoli di avere creato correttamente la vista '**`MovimentiExtra`**', la si può interrogare come se fosse una relazione normale:

```
$ sqlite3 mag.db [Invio]
```

```
  SQLite version …
  Enter ".help" for instructions
sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM MovimentiExtra; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Movimento Articolo Causale Data       Cliente Fornitore Quantita Valore ValoreUnitario QuantitaAlgebrica
--------- ------- ------- ---------- ------- --------- -------- ------ -------------- -----------------
1         2       1       2012-01-15         3         10000    100    0.01           10000
2         2       2       2012-01-16 2               1000     10     0.01           -1000
3         102     1       2012-01-17         2       1000     200    0.2            1000
4         102     2       2012-01-18 1               100      20     0.2            -100
5         401     1       2012-01-19         1       1000     200    0.2            1000
6         401     2       2012-01-20 3               200      20     0.2            -200
7         401     4       2012-01-20         1       100      20     0.2            -100
8         102     4       2012-01-20         2       100      20     0.2            -100
9         601     1       2012-01-21         3       2000     1000   0.5            2000
10        601     2       2012-01-25 1               1000     500    0.5            -1000
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file '`vista-movimenti-extra.sql`'.

79.12.4 Conclusione

Prima di passare alla sezione successiva, si deve riprendere il file '`magazzino.sql`' e vi si deve aggiungere l'istruzione per la creazione della vista '**`MovimentiExtra`**', come realizzato nella verifica appena conclusa.

Una volta aggiornato il file '`magazzino.sql`' come descritto, si deve cancellare il file '`mag.db`' e ricreare a partire dalle istruzioni contenute nel file '`magazzino.sql`':

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

Se vengono segnalati degli errori, occorre correggere il file '`magazzino.sql`', cancellare nuovamente il file '`mag.db`', quindi si deve ripetere l'operazione. La base di dati contenuta nel file '`mag.db`', viene usata nella sezione successiva e non si può proseguire se non si riesce a ricrearla correttamente.

79.13 Aggregazioni

L'aggregazione è una forma di interrogazione attraverso cui si ottengono risultati riepilogativi del contenuto di una relazione, nel suo complesso o a gruppi di tuple. Per questo si utilizzano delle funzioni speciali al posto dell'espressione che esprime gli attributi del risultato. Queste funzioni restituiscono un solo valore e come tali concorrono a creare un'unica tupla.

Tabella 79.150. Alcune funzioni aggreganti riconosciute da SQLite.

Funzione	Descrizione
`COUNT(`***x***`)`	Restituisce il numero di tuple, nel gruppo, per le quali l'espressione ***x*** restituisce un valore diverso da '**`NULL`**'.
`COUNT(*)`	Restituisce il numero di tuple esistenti nel gruppo.
`AVG(`***x***`)`	Restituisce la media, nel gruppo di tuple, dei valori che ottiene l'espressione ***x***, escludendo '**`NULL`**' e considerando i valori non numerici pari a zero.
`MIN(`***x***`)` `MAX(`***x***`)`	Restituisce il valore minimo o massimo, nel gruppo di tuple, dei valori che ottiene l'espressione ***x***, escludendo '**`NULL`**' e considerando i valori non numerici pari a zero.
`SUM(`***x***`)`	Restituisce la somma, nel gruppo di tuple, dei valori che ottiene l'espressione ***x***, escludendo '**`NULL`**' e considerando i valori non numerici pari a zero.

La forma che assume l'istruzione '**`SELECT`**' quando si usano le aggregazioni e tipicamente quella seguente:

```
SELECT specificazione_dell'attributo_1 [ , …specificazione_dell'attributo_n ]
    FROM specificazione_della_relazione_1 [ , …specificazione_della_relazione_n ]
    [WHERE condizione]
    [GROUP BY attributo_1 [ , … ]]
```

In pratica, le funzioni aggreganti vanno usate nell'elenco che descrive gli attributi. Se non si usa l'opzione '**`GROUP BY`**', il gruppo di tuple di riferimento comprende tutte le tuple della relazione o della congiunzione (di relazioni). Se si specifica l'opzione '**`GROUP BY`**', le tuple vengono raggruppate in base all'uguaglianza degli attributi indicati come argomento di tale opzione. In pratica:

1. la relazione ottenuta dall'istruzione '**`SELECT`**…**`FROM`**' viene filtrata dalla condizione '**`WHERE`**';
2. la relazione risultante viene riordinata in modo da raggruppare le tuple in cui i contenuti degli attributi elencati dopo l'opzione '**`GROUP BY`**' sono uguali;
3. su questi gruppi di tuple vengono valutate le funzioni di aggregazione.

79.13.1 Aggregazioni banali

Per prendere un po' di dimestichezza con le aggregazioni, conviene usare il programma '**`sqlite3`**' in modo interattivo e fare qualche piccolo esperimento:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions
sqlite> .headers on [Invio]

sqlite> .mode column [Invio]
```

Si vogliono contare le tuple della relazione '**`Movimenti`**':

```
sqlite> SELECT COUNT(*) FROM Movimenti; [Invio]

  10
```

Si vogliono contare i movimenti per ogni tipo di articolo:

```
sqlite> SELECT Articolo, COUNT(*) FROM Movimenti GROUP BY
Articolo; [Invio]

  Articolo    COUNT(*)
  ----------  ----------
  2           2
  102         3
  401         3
  601         2
```

Si vuole conoscere la quantità esistente di ogni articolo (si usa la vista ***MovimentiExtra***, che offre l'attributo '**`QuantitaAlgebrica`**'):

```
sqlite> SELECT Articolo, SUM(QuantitaAlgebrica) [Invio]

   ...>     FROM MovimentiExtra GROUP BY Articolo; [Invio]

  Articolo    SUM(QuantitaAlgebrica)
  ----------  ----------------------
  2           9000
  102         800
  401         700
  601         1000
```

Si vuole conoscere la quantità esistente di ogni articolo in magazzino e il valore (il valore viene calcolato a partire da quello medio, moltiplicato per la quantità algebrica):

```
sqlite> SELECT Articolo, SUM(QuantitaAlgebrica), [Invio]

    ...>       SUM(QuantitaAlgebrica*ValoreUnitario) [Invio]

    ...>     FROM MovimentiExtra GROUP BY Articolo; [Invio]
```

```
Articolo    SUM(QuantitaAlgebrica)  SUM(QuantitaAlgebrica*ValoreUnitario)
----------  ----------------------  -------------------------------------
2           9000                    90
102         800                     160
401         700                     140
601         1000                    500
```

Si vuole conoscere la quantità esistente di ogni articolo in magazzino e il costo medio, determinato dividendo il valore complessivo per la quantità esistente:

```
sqlite> SELECT Articolo, [Invio]

sqlite>        SUM(QuantitaAlgebrica), [Invio]

sqlite>        SUM(QuantitaAlgebrica*ValoreUnitario)/↩
↪SUM(QuantitaAlgebrica) [Invio]

sqlite>        FROM MovimentiExtra GROUP BY Articolo; [Invio]

Articolo    SUM(QuantitaAlgebrica)  SUM(QuantitaAlgebrica*ValoreUnitario)/SUM(QuantitaAlgebrica)
----------  ----------------------  ------------------------------------------------------------
2           9000                    0.01
102         800                     0.2
401         700                     0.2
601         1000                    0.5

sqlite> .quit [Invio]
```

79.13.2 Verifica sulla creazione della vista «SituazioneMagazzino»

Si vuole realizzare la vista '**SituazioneMagazzino**' che, in questa verifica, si limiti a mostrare poche informazioni riepilogative sullo stato del magazzino.

Si realizzi il file '`vista-situazione-magazzino-1.sql`', seguendo lo scheletro che viene proposto, per far sì che la vista '**SituazioneMagazzino**' contenga gli attributi seguenti:

1. '**Codice**', corrispondente al codice articolo della relazione '**Movimenti**' o della vista '**MovimentiExtra**';
2. '**Articolo**', corrispondente alla descrizione dell'articolo, come indicato nella relazione '**Articoli**';
3. '**Esistenza**', corrispondente alla somma algebrica dei carichi, come si ottiene dalla vista '**MovimentiExtra**'.

Figura 79.157. Scheletro del file '`vista-situazione-magazzino-1.sql`', da completare.

```
-- Vista "SituazioneMagazzino"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-situazione-magazzino-1.sql

CREATE VIEW SituazioneMagazzino AS
    SELECT MovimentiExtra.Articolo        AS Codice,
           …
           …
           FROM …
           WHERE MovimentiExtra.Articolo = Articoli.Articolo
           GROUP BY MovimentiExtra.Articolo;
```

Per eseguire il file '`vista-situazione-magazzino-1.sql`', si agisce come sempre:

```
$ sqlite3 mag.db < vista-situazione-magazzino-1.sql [Invio]
```

Se la creazione della vista produce degli errori, occorre eliminare la vista e, dopo la correzione del file '`vista-situazione-magazzino-1.sql`', si può ritentare.

Quando si è consapevoli di avere creato correttamente la vista '**SituazioneMagazzino**', la si può interrogare come se fosse una relazione normale:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions

sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM SituazioneMagazzino; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Codice      Articolo                              Esistenza
----------  ------------------------------------  ----------
2           Dischetti da 9 cm 1440 Kibyte colorati  9000
102         CD-R 52x                              800
401         DVD+R 8x                              700
601         DVD+RW 8x                             1000
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file '`vista-situazione-magazzino-1.sql`'.

79.13.3 Verifica sulla creazione della vista «SituazioneMagazzino»

Si vuole estendere la vista '**SituazioneMagazzino**', già realizzata in parte nella verifica precedente; pertanto, in questa verifica si modifica il file '`vista-situazione-magazzino-1.sql`' Salvandolo con il nome '`vista-situazione-magazzino-2`'. Si vogliono ottenere gli attributi seguenti:

1. '**Codice**', corrispondente al codice articolo della relazione '**Movimenti**' o della vista '**MovimentiExtra**';
2. '**Articolo**', corrispondente alla descrizione dell'articolo, come indicato nella relazione '**Articoli**';
3. '**ScortaMin**', corrispondente alla scorta minima, come contenuto nella relazione '**Articoli**';
4. '**Esistenza**', corrispondente alla somma algebrica dei carichi, come si ottiene dalla vista '**MovimentiExtra**';
5. '**Valore**', corrispondente al valore complessivo di ogni articolo (come mostrato negli esempi prima di queste verifiche).

Figura 79.160. Scheletro del file '`vista-situazione-magazzino-2.sql`', da completare.

```
-- Vista "SituazioneMagazzino"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-situazione-magazzino-2.sql

CREATE VIEW SituazioneMagazzino AS
    SELECT MovimentiExtra.Articolo            AS Codice,
           …
           …
           FROM …
           WHERE MovimentiExtra.Articolo = Articoli.Articolo
           GROUP BY MovimentiExtra.Articolo;
```

Prima di poter eseguire questo file con la base di dati, occorre eliminare la vista '**SituazioneMagazzino**', che già dovrebbe esistere. Si ricorda che per eliminare una vista si utilizza l'istruzione '**DROP VIEW**' e che conviene intervenire con il programma '**sqlite3**' in modo interattivo.

Per eseguire il file '`vista-situazione-magazzino-2.sql`', si agisce come sempre:

```
$ sqlite3 mag.db < vista-situazione-magazzino-2.sql [Invio]
```

Se la creazione della vista produce degli errori, occorre eliminare la vista e, dopo la correzione del file '`vista-situazione-magazzino-2.sql`', si può ritentare.

Quando si è consapevoli di avere creato correttamente la vista '**SituazioneMagazzino**', la si può interrogare come se fosse una relazione normale:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions

sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM SituazioneMagazzino; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Codice      Articolo                                ScortaMin   Esistenza   Valore
----------  --------------------------------------  ----------  ----------  ----------
2           Dischetti da 9 cm 1440 Kibyte colorati  500         9000        90
102         CD-R 52x                                500         800         160
401         DVD+R 8x                                200         700         140
601         DVD+RW 8x                               200         1000        500
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file ‘`vista-situazione-magazzino-2.sql`’.

79.13.4 Verifica sulla creazione della vista «SituazioneMagazzino»

Si vuole estendere la vista ‘**`SituazioneMagazzino`**’, già realizzata in parte nella verifica precedente, in modo ottenere anche il costo medio; pertanto, in questa verifica si modifica il file ‘`vista-situazione-magazzino-2.sql`’ salvandolo con il nome ‘`vista-situazione-magazzino-3.sql`’. Si vogliono ottenere gli attributi seguenti:

1. ‘**`Codice`**’, corrispondente al codice articolo della relazione ‘**`Movimenti`**’ o della vista ‘**`MovimentiExtra`**’;
2. ‘**`Articolo`**’, corrispondente alla descrizione dell’articolo, come indicato nella relazione ‘**`Articoli`**’;
3. ‘**`ScortaMin`**’, corrispondente alla scorta minima, come contenuto nella relazione ‘**`Articoli`**’;
4. ‘**`Esistenza`**’, corrispondente alla somma algebrica dei carichi, come si ottiene dalla vista ‘**`MovimentiExtra`**’;
5. ‘**`Valore`**’, corrispondente al valore complessivo di ogni articolo (come mostrato negli esempi prima di queste verifiche);
6. ‘**`CostoMedio`**’, corrispondente al valore complessivo di ogni articolo, diviso la quantità esistente (come mostrato negli esempi prima di queste verifiche).

Figura 79.163. Scheletro del file ‘`vista-situazione-magazzino-3.sql`’, da completare.

```
-- Vista "SituazioneMagazzino"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-situazione-magazzino-3.sql

CREATE VIEW SituazioneMagazzino AS
    SELECT MovimentiExtra.Articolo         AS Codice,
           …
           …
           FROM …
           WHERE MovimentiExtra.Articolo = Articoli.Articolo
           GROUP BY MovimentiExtra.Articolo;
```

Prima di poter eseguire questo file con la base di dati, occorre eliminare la vista ‘**`SituazioneMagazzino`**’, che già dovrebbe esistere. Si ricorda che per eliminare una vista si utilizza l’istruzione ‘**`DROP VIEW`**’ e che conviene intervenire con il programma ‘**`sqlite3`**’ in modo interattivo.

Per eseguire il file ‘`vista-situazione-magazzino-3.sql`’, si agisce come sempre:

```
$ sqlite3 mag.db < vista-situazione-magazzino-3.sql [Invio]
```

Se la creazione della vista produce degli errori, occorre eliminare la vista e, dopo la correzione del file ‘`vista-situazione-magazzino-3.sql`’, si può ritentare.

Quando si è consapevoli di avere creato correttamente la vista ‘**`SituazioneMagazzino`**’, la si può interrogare come se fosse una relazione normale:

```
$ sqlite3 mag.db [Invio]

 SQLite version …
 Enter ".help" for instructions

sqlite> .headers on [Invio]

sqlite> .mode column [Invio]
```



```
sqlite> SELECT * FROM SituazioneMagazzino; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Codice Articolo                                ScortaMin Esistenza Valore CostoMedio
------ --------------------------------------  --------- --------- ------ ----------
2      Dischetti da 9 cm 1440 Kibyte colorati  500       9000      90     0.01
102    CD-R 52x                                500       800       160    0.2
401    DVD+R 8x                                200       700       140    0.2
601    DVD+RW 8x                               200       1000      500    0.5
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file ‘`vista-situazione-magazzino-3.sql`’.

79.13.5 Conclusione

Prima di passare alla sezione successiva, si deve riprendere il file ‘`magazzino.sql`’ e vi si deve aggiungere l’istruzione per la creazione della vista ‘**`SituazioneMagazzino`**’, come realizzato nel’ultima verifica appena conclusa.

Una volta aggiornato il file ‘`magazzino.sql`’ come descritto, si deve cancellare il file ‘`mag.db`’ e ricreare a partire dalle istruzioni contenute nel file ‘`magazzino.sql`’:

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

Se vengono segnalati degli errori, occorre correggere il file ‘`magazzino.sql`’, cancellare nuovamente il file ‘`mag.db`’, quindi si deve ripetere l’operazione. La base di dati contenuta nel file ‘`mag.db`’, viene usata ancora e non si può proseguire se non si riesce a ricrearla correttamente.

79.14 Inserimento automatico del costo medio

A conclusione di queste lezioni sul linguaggio SQL, viene mostrata la soluzione di un problema, senza richiedere altre verifiche.

Quando si inserisce un movimento nella relazione ‘**`Movimenti`**’, l’utente deve indicare il valore del movimento. Si determina facilmente questo valore quando il bene viene acquistato, in quanto corrisponde al costo complessivo (IVA esclusa). Quando l’articolo viene scaricato per perché reso al fornitore, il valore deve essere lo stesso della fattura a cui si riferisce (in proporzione alla quantità resa), ma quando viene scaricato per la vendita, occorre decidere come attribuire questo valore.

Il modo più semplice per definire il valore del bene che viene scaricato per la vendita, o comunque per scopi diversi dal reso, è quello di calcolare il costo medio ponderato per movimento. In pratica, si tratterebbe di consultare la vista ‘**`SituazioneMagazzino`**’, prima di procedere all’inserimento, in modo da conoscere il costo medio unitario, ottenuto in base ai movimenti esistenti.

Quello che si vuol fare qui è di costruire un grilletto che inserisca automaticamente il valore, determinandolo in base al costo medio ponderato per movimento, quando si inserisce un movimento e si omette l’indicazione del valore stesso.

79.14.1 Vista «CostoMedioValido»

La vista ‘**`SituazioneMagazzino`**’ calcola il costo medio tenendo conto di tutte le tuple, anche quelle che contengono un valore indeterminato del movimento (‘**`NULL`**’). Per lo scopo che si vuole raggiungere, è necessario calcolare il costo medio escludendo i valori indeterminati; pertanto, si realizza una vista apposita:

```
-- Vista "CostoMedioValido"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-costo-medio-valido.sql

CREATE VIEW CostoMedioValido AS
    SELECT Articolo,
           SUM(QuantitaAlgebrica*ValoreUnitario)        AS Valore,
           (SUM(QuantitaAlgebrica*ValoreUnitario)/SUM(QuantitaAlgebrica))
                                                        AS CostoMedio
           FROM MovimentiExtra
           WHERE Valore NOT NULL
           GROUP BY Articolo;
```

79.14.2 Grilletto «ValorizzazioneScarichi»

Si può creare un grilletto, che aggiorni automaticamente tutte le tuple della relazione '**`Movimenti`**', che hanno un valore movimentato indeterminato, traendo il costo medio dalla vista '**`CostoMedioValido`**':

```
-- Grilletto "ValorizzazioneScarichi"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletto-valorizzazione-scarichi.sql

CREATE TRIGGER ValorizzazioneScarichi
       AFTER INSERT ON Movimenti
       BEGIN
            UPDATE Movimenti
                   SET Valore =
                       (SELECT CostoMedio * NEW.Quantita FROM CostoMedioValido
                        WHERE Articolo = NEW.articolo)
                   WHERE Valore IS NULL;
       END;
```

Naturalmente, i movimenti che vengono presi in considerazione dal grilletto, sono solo quelli che vengono inseriti **dopo** la sua creazione.

Si osservi, comunque, che occorre anche impedire la sostituzione del valore con qualcosa di indeterminato. In pratica, occorre estendere il grilletto associato alla modifica delle tuple della relazione '**`Movimenti`**', in modo da non accettare valori indeterminati per l'attributo del valore:

```
CREATE TRIGGER Movimenti_upd
       BEFORE UPDATE ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            …
            …
            …
            WHEN (NEW.Valore IS NULL)
            THEN
                RAISE (ABORT, 'In fase di variazione, il valore non può essere indeterminato!')
            END;
       END;
```