Music Shop

**Gruppo 10**

Il database serve per gestire una catena nota di negozi nazionale che si occupano della vendita e del noleggio di cd musicali nelle varie città di interesse.

In particolare interessa gestire una catena di filiali che sono in grado di rifornirsi a vicenda. Ogni filiale espone dei cd musicali.

Le filiali devono avere la possibilità di rifornirsi a vicenda e di rifornirsi con il magazzino di riferimento. Occorre anche tener presente che se il magazzino è situato nella stessa provincia della filiale e si avvia una procedura di rifornimento tra filiali, allora la filiale deve dare precedenza al magazzino e si deve rifornire da quest’ultimo.

Dei magazzini si memorizzano: il codice (univoco) e l’indirizzo (composto da: via, cap, città, num civico e provincia).

Ogni filiale è rifornita da un solo magazzino, mentre un magazzino rifornisce più filiali.

Delle filiali si memorizzano: il codice (univoco), il numero di telefono e l’indirizzo.

Ogni filiale possiede del personale: dirigenti e dipendenti. Un dirigente dirige una e una sola filiale, mentre in una filiale possono lavorare più dipendenti. Dei dipendenti e dei dirigenti si memorizzano: costo orario (un dirigente costerà di più di un dipendente), nome, cognome, codice fiscale, telefono e i dati di residenza. Deve essere quindi possibile calcolare il costo totale del personale per ogni filiale.

Una filiale, oltre a venedere e noleggiare cd musicali, può anche sponsorizzare eventi musicali per aumentare la propria notorietà e quella della catena. In particolare una filiale sponsorizza molti eventi e un evento è sponsorizzato da una sola filiale.

Un evento è caratterizzato da: località, data e nome.

Dei cd musicali interessa memorizzare: il nome, la durata, il numero dei brani, il prezzo cad e la disponibilità di copie all’interno della filiale, in caso di assenza della disponibilità si avvierà una procedura di rifornimento del prodotto.

Viene gestito anche il genere musicale dei cd, del genere si memorizza: il nome (univoco), l’anno di popolarità in cui era in voga il genere musicale.

Un cd musicale può aderire a molti generi musicali e ad un genere musicale possono appartenere molti cd musicali.

Interessano gestire anche le band che hanno scritto i cd, delle quali si memorizzano nome e data di fondazione.

Per band e cd musicali occorre gestire il fatto che i nomi di entrambi non sono univoci.

E’ necessario inoltre gestire le case discografiche, in particolare una band può far parte di una casa discografica e una casa discografica segue molte band differenti. La casa discografica è identificata da: nome (univoco) e anno di fondazione.

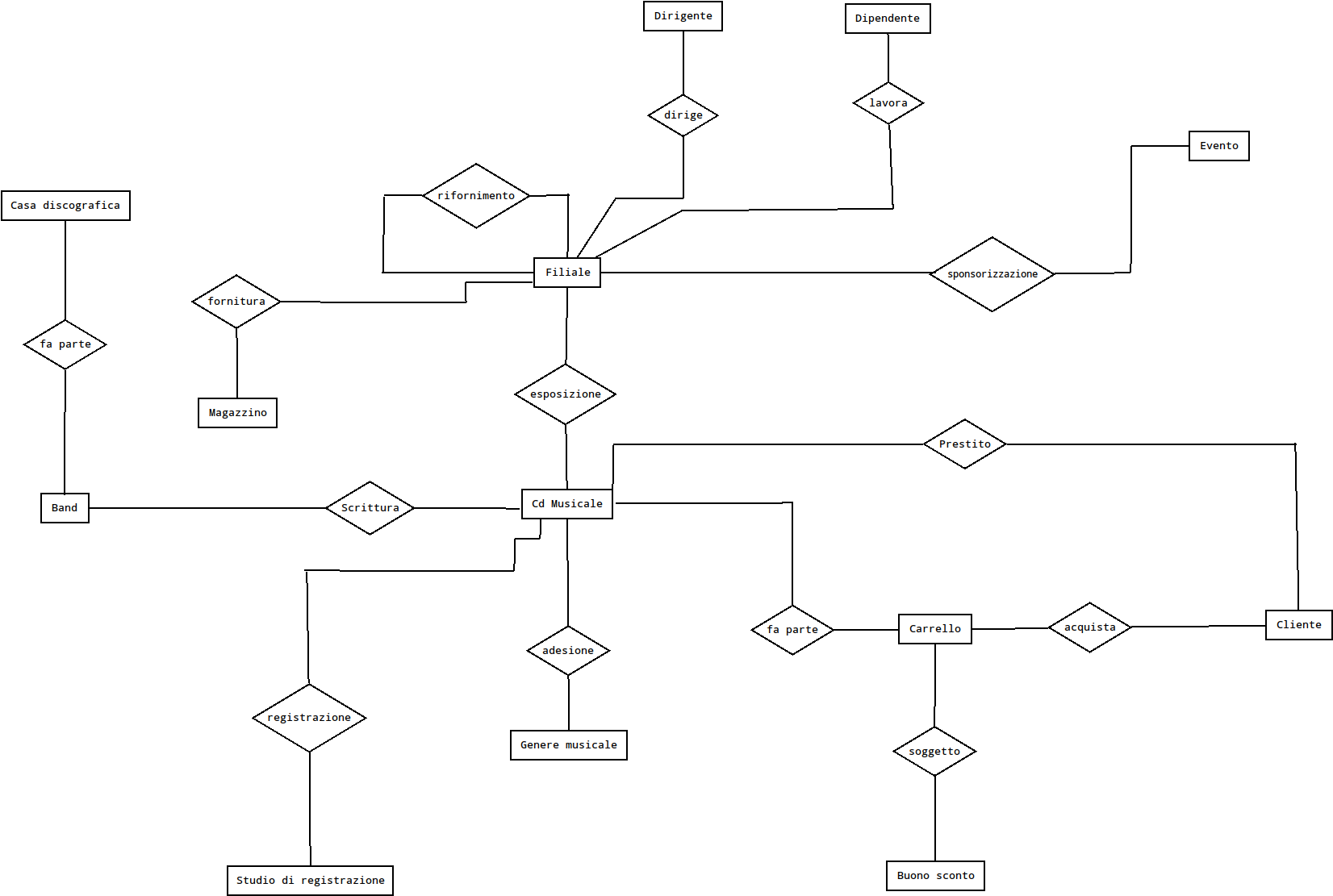
Infine per il cd musicale deve essere anche gestito lo studio di registrazione che si è occupato di produrre il cd. Dello studio di registrazione di memorizzano: nome e e l’indirizzo (composto da: via, cap, città, num civico e provincia). Un cd quindi è registrato da uno studio di registrazione e uno studio di registrazione registra molti cd, occorre tener memorizzato anche il costo della registrazione del cd.

E’ inoltre necessario gestire i clienti che acquistano i cd musicali con i propri rispettivi carrelli di acquisto. Dei clienti si memorizza: codice fiscale, nome, cognome e telefono. Il cliente acquista un carrello in una certa data, il quale contiene dei cd musicali. Il carrello è identificato da un numero (unico se associato a un determinato cliente) e un importo totale.

Inoltre la nota catena di negozi mette a disposizione dei buoni sconto per i vari clienti. Il buono sconto è identificato da un codice (univoco), un attributo booleano che indica se è valido oppure no e il valore dello sconto in percentuale. Il carrello può essere quindi soggetto a un buono sconto, che una volta utilizzato perde di validità.

Un cliente deve poter anche prendere in prestito dei cd musicali per poterli ascoltare e decidere se acquistarli successivamente oppure no. In particolare interessa gestire il vincolo che un cliente può prendere in prestito un determinato cd una ed una sola volta, dopodichè, se lo vuole, è costretto ad acquistarlo la volta sucessiva.

# Schema Scheletro



# Progettazione Concettuale

# Schema Er Completo

# SchemaER

# Schema logico

CASA\_DISCOGRAFICA(nome, anno\_fondazione)

BAND(cod\_band, nome, data\_fondazione, nome\_c)

FK: nome\_c REFERENCES CASA\_DISCOGRAFICA

STUDIO\_DI\_REGISTRAZIONE(nome, via, cap, citta, num\_civico, provincia)

CD\_MUSICALE(nome, cod\_band, nome\_studio, prezzo\_cad, durata, num\_brani, costo\_reg, citta)

FK: cod\_band REFERENCES BAND

FK: nome\_studio, citta REFERENCES STUDIO\_DI\_REGISTRAZIONE NOT NULL

GENERE\_MUSICALE(nome, popolarità)

ADESIONE(nome, cod\_band, nome\_cd)

FK: nome\_cd, cod\_band REFERENCES CD\_MUSICALE

FK: nome REFERENCES GENERE\_MUSICALE

MAGAZZINO(cod\_m, via, cap, citta, num\_civico, provincia)

DIRIGENTE(cf, nome, cognome, costo\_orario, telefono, via, cap, citta, num\_civico, provincia, anzianità)

FILIALE(cod\_f, costo\_personale, telefono, via, cap, citta, num\_civico, provincia, cod\_m)

FK: cod\_m REFERENCES MAGAZZINO NOT NULL

DIRIGE(cod\_f, cf\_dir)

FK: cod\_f REFERENCES FILIALE

FK: cf\_dir REFERENCES DIRIGENTE

AK: cf\_dir

DIPENDENTE(cf, nome, cognome, costo\_orario, telefono, via, cap, citta, num\_civico, provincia, cod\_f)

FK: cod\_f REFERENCES FILIALE NOT NULL

RIFORNIMENTO(cod\_fornitore, cod\_rifornisce)

FK: cod\_fornitore REFERENCES FILIALE

FK: cod\_rifornisce REFERENCES FILIALE

EVENTO(nome, data, località, cod\_f)

FK: cod\_f REFERENCES FILIALE NOT NULL

ESPOSIZIONE(nome\_cd, cod\_band, cod\_f, quantita)

FK: nome\_cd, cod\_band REFERENCES CD\_MUSICALE

FK: cod\_f REFERENCES FILIALE

CLIENTE(cf, nome, cognome, telefono, spesa\_totale)

PRESTITO(data, cf, nome\_cd, cod\_band, costo, cod\_f)

FK: cf REFERENCES CLIENTE

FK: nome\_cd, cod\_band REFERENCES CD\_MUSICALE

FK: cod\_f REFERENCES FILIALE NOT NULL

BUONO\_SCONTO(cod\_b, sconto, valido)

CARRELLO(num, cf, data, importo\_totale, cod\_f)

FK: cf REFERENCES CLIENTE

FK: cod\_f REFERENCES FILIALE NOT NULL

FA\_PARTE(num, cf, nome\_cd, cod\_band, qta\_copie)

FK: num, cf REFERENCES CARRELLO

FK: nome\_cd, cod\_band REFERENCES CD\_MUSICALE

SOGGETTO(cod\_b, num, cf)

FK: num, cf REFERENCES CARRELLO

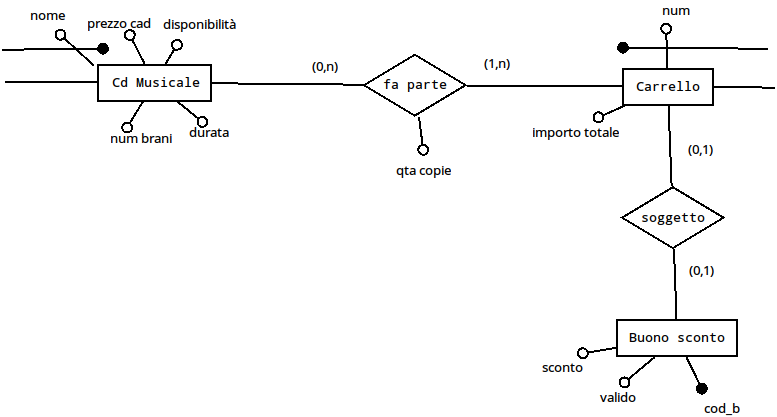
FK: cod\_b REFERENCES BUONO\_SCONTO

AK: num, cf

# Studio dati derivati

Nel progetto sono presenti diversi dati derivati, si è deciso di limitare lo studio a due soli di questi. In particolare si è studiato il dato derivato “importo totale” presente nell’entità carrello e “costo personale” presente nell’entità filiale.

1. Il dato derivato “importo totale” è calcolato come la somma dei prezzi di tutti i prodotti molitiplicato le quantità rispettive e tolto il valore di un buono sconto se utilizzato.



**Tabella dei volumi**

Si è considerata una gamma di scelta di 10000 cd differenti, in media un carrello contiene 2.5 cd e un cd fa parte di 3 carrelli (con 30000 record nell’associazione).

|  |  |  |
| --- | --- | --- |
| Concetto | Tipo | Volume |
| Cd Musicale | E | 10000 |
| Fa\_parte | R | 30000 |
| Carrello | E | 12000 |
| Soggetto | R | 3000 |
| Buono sconto | E | 6000 |

**Tabella delle operazioni**

Sono state considerate due operazioni:

1. Visualizzare importo totale di un carrello
2. Dato cd aggiungerlo a un carrello (dato cd e carrello)
3. Utilizzo di un buono sconto in un carrello (dato carrello e codice buono)

Si è supposto che l’operazione di visualizzazione dell’importo totale avvenga con minore frequenza rispetto all’aggiunta di un cd nel carrello.

|  |  |  |
| --- | --- | --- |
| Operazione | Tipo | Frequenza |
| Op1 | I | 400/g |
| Op2 | I | 500/g |
| Op3 | I | 50/g |

**Tabella accessi con dato derivato**

Si suppone che i dati del cd siano già dati (prezzo compreso), nell’operazione 2 non sarà quindi necessario andare ad effettuare una lettura sull’entità cd.

Si suppone invece di non conosce fin da subito il valore dello sconto del buono, sarà quindi necessaria un lettura extra sul buono per recuperarlo.

|  |  |  |  |
| --- | --- | --- | --- |
| Operazione | Concetto | Accessi | Tipo |
| Op1 | Carrello | 1 | L |
| Op2 | Fa\_parte | 1 | S |
| Op2 | Carrello | 1 | L |
| Op2 | Carrello | 1 | S |
| Op3 | Buono Sconto | 1 | L |
| Op3 | Soggetto | 1 | S |
| Op3 | Carrello | 1 | L |
| Op3 | Carrello | 1 | S |

Costo: 400+2500+300 = 3200/g

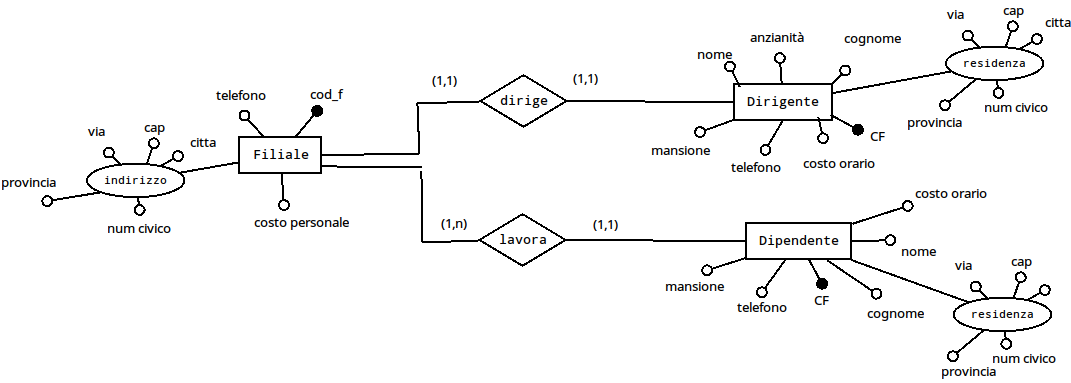
**Tabella accessi senza dato derivato**

|  |  |  |  |
| --- | --- | --- | --- |
| Operazione | Concetto | Accessi | Tipo |
| Op1 | Carrello | 1 | L |
| Op1 | Fa\_parte | 2.5 | L |
| Op1 | Cd Musicale | 2.5 | L |
| Op2 | Fa\_parte | 1 | S |
| Op3 | Soggetto | 1 | S |

Costo: 2400+1000+100 = 3500/g

In questo caso conviene tenere il dato derivato “importo totale”.

2) Il dato derivato “costo personale” dell’entità “filiale” è calcolato come la somma dei costi orari di tutti i dipendenti e dirigenti che lavorano in quella fialiale. Lo studio sul dato derivato è stato effettuato sullo schema er seguente, che rappresenta la forma d’implementazione scelta.



**Tabella dei volumi**

Si è considerata una catena di negozi abbastanza famosa con 200 filiali differenti sparse per la nazione. Ogni filiale è seguita da un dirigente diverso, ci saranno quindi 200 dirigenti. In una filiale lavorano più dipendenti, ma un dipendente lavora per una e una sola filiale. Sono stati considerati in media 30 dipendenti per filiale.

|  |  |  |
| --- | --- | --- |
| Concetto | Tipo | Volume |
| Filiale | E | 200 |
| Dirige | R | 200 |
| Dirigente | E | 200 |
| Dipendente | E | 6000 |
| Lavora | R | 6000 |

**Tabella delle operazioni**

Sono state considerate tre operazioni:

1. Modifica del costo orario di un dipendente, dato codice fiscale
2. Modifica del costo orario di un dirigente, dato codice fiscale
3. Controllo sul costo del personale di una determinata filiale

Sono tutte operazioni che avvengono abbastanza di rado in una azienda, ma quando si parla di una catena di negozi queste aumentano all’aumentare delle sedi e dei dipendenti, è stato quindi necessario tenerne conto.

|  |  |  |
| --- | --- | --- |
| Operazione | Tipo | Frequenza |
| Op1 | I | 10/g |
| Op2 | I | 2/g |
| Op3 | I | 5/g |

**Tabella accessi con dato derivato**

|  |  |  |  |
| --- | --- | --- | --- |
| Operazione | Concetto | Accessi | Tipo |
| Op1 | Dipendente | 1 | L |
| Op1 | Dipendente | 1 | S |
| Op1 | Lavora | 1 | L |
| Op1 | Filiale | 1 | L |
| Op1 | Filiale | 1 | S |
| Op2 | Dirigente | 1 | L |
| Op2 | Dirigente | 1 | S |
| Op2 | Dirige | 1 | L |
| Op2 | Filiale | 1 | L |
| Op2 | Filiale | 1 | S |
| Op3 | Filiale | 1 | L |

Costo: 70+14+5 = 89/g

**Tabella accessi senza dato derivato**

|  |  |  |  |
| --- | --- | --- | --- |
| Operazione | Concetto | Accessi | Tipo |
| Op1 | Dipendente | 1 | L |
| Op1 | DIpendente | 1 | S |
| Op2 | Dirigente | 1 | L |
| Op2 | Dirigente | 1 | S |
| Op3 | Filiale | 1 | L |
| Op3 | Dirige | 1 | L |
| Op3 | Dirigente | 1 | L |
| Op3 | Lavora | 30 | L |
| Op3 | Dipendente | 30 | L |

Costo: 30+6+315 = 351/g

Come mostra lo studio appena svolto, inaspettatamente, conviene tenere il dato derivato in questo caso.

# Query di creazione

Di seguito si presentano le query per la creazione dell’intero database. Si evidenziano in verde i vincoli imposti non esprimibili durante la fase di progettazione.

CREATE TABLE CASA\_DISCOGRAFICA(

nome VARCHAR(30) PRIMARY KEY,

anno\_fondazione INTEGER NOT NULL

);

CREATE TABLE BAND(

cod\_band INTEGER PRIMARY KEY,

nome VARCHAR(30) NOT NULL,

data\_fondazione date NOT NULL,

nome\_c VARCHAR(30) NOT NULL,

FOREIGN KEY (nome\_c) REFERENCES CASA\_DISCOGRAFICA(nome)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE STUDIO\_DI\_REGISTRAZIONE(

nome VARCHAR(30) NOT NULL,

via VARCHAR(60) NOT NULL,

cap CHAR(5) NOT NULL,

citta VARCHAR(60) NOT NULL,

num\_civico INTEGER NOT NULL,

provincia VARCHAR(60) NOT NULL,

PRIMARY KEY(nome,citta),

CHECK(num\_civico > 0 )

);

CREATE TABLE CD\_MUSICALE(

nome VARCHAR(60) NOT NULL,

cod\_band INTEGER NOT NULL,

nome\_studio VARCHAR(60) NOT NULL,

prezzo\_cad INTEGER NOT NULL,

durata TIME NOT NULL,

num\_brani INTEGER NOT NULL,

costo\_reg DECIMAL(10,3) NOT NULL,

citta VARCHAR(60) NOT NULL,

PRIMARY KEY(nome, cod\_band),

FOREIGN KEY (nome\_studio, citta) REFERENCES STUDIO\_DI\_REGISTRAZIONE(nome, citta)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (cod\_band) REFERENCES BAND(cod\_band)

ON DELETE CASCADE ON UPDATE CASCADE,

CHECK(prezzo\_cad > 0),

CHECK(durata > '00:02:00' ),

CHECK(num\_brani > 0)

);

CREATE TABLE GENERE\_MUSICALE(

nome VARCHAR(60) PRIMARY KEY,

popolarita INTEGER NOT NULL,

CHECK(popolarita > 1500)

);

CREATE TABLE ADESIONE(

genere VARCHAR(60) NOT NULL,

cod\_band INTEGER NOT NULL,

nome\_cd VARCHAR(60) NOT NULL,

PRIMARY KEY(genere, cod\_band, nome\_cd),

FOREIGN KEY (nome\_cd, cod\_band) REFERENCES CD\_MUSICALE(nome, cod\_band)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (genere) REFERENCES GENERE\_MUSICALE(nome)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE MAGAZZINO(

cod\_m INTEGER PRIMARY KEY,

via VARCHAR(60) NOT NULL,

cap CHAR(5) NOT NULL,

citta VARCHAR(60) NOT NULL,

num\_civico INTEGER NOT NULL,

provincia VARCHAR(60) NOT NULL,

CHECK(num\_civico > 0 )

);

CREATE TABLE DIRIGENTE(

cf CHAR(16) PRIMARY KEY,

nome VARCHAR(60) NOT NULL,

cognome VARCHAR(60) NOT NULL,

costo\_orario DECIMAL(10,3) NOT NULL,

telefono VARCHAR(10),

via VARCHAR(60) NOT NULL,

cap CHAR(5) NOT NULL,

citta VARCHAR(60) NOT NULL,

num\_civico INTEGER NOT NULL,

provincia VARCHAR(60) NOT NULL,

anzianita INTEGER NOT NULL,

CHECK(num\_civico > 0 ),

CHECK(anzianita > 0 ),

CHECK(costo\_orario > 0)

);

CREATE TABLE FILIALE(

cod\_f INTEGER PRIMARY KEY,

costo\_personale DECIMAL(15,3) NOT NULL,

telefono VARCHAR(10) NOT NULL,

via VARCHAR(60) NOT NULL,

cap CHAR(5) NOT NULL,

citta VARCHAR(60) NOT NULL,

num\_civico INTEGER NOT NULL,

provincia VARCHAR(60) NOT NULL,

cod\_m INTEGER NOT NULL,

FOREIGN KEY (cod\_m) REFERENCES MAGAZZINO(cod\_m)

ON DELETE CASCADE ON UPDATE CASCADE,

CHECK(num\_civico > 0),

CHECK(costo\_personale >= 0)

);

CREATE TABLE DIRIGE(

cod\_f INTEGER PRIMARY KEY,

cf\_dir CHAR(16) NOT NULL,

UNIQUE(cf\_dir),

FOREIGN KEY (cod\_f) REFERENCES FILIALE(cod\_f)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (cf\_dir) REFERENCES DIRIGENTE(cf)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE DIPENDENTE(

cf CHAR(16) PRIMARY KEY,

nome VARCHAR(60) NOT NULL,

cognome VARCHAR(60) NOT NULL,

costo\_orario DECIMAL(10,3) NOT NULL,

telefono VARCHAR(10),

via VARCHAR(60) NOT NULL,

cap CHAR(5) NOT NULL,

citta VARCHAR(60) NOT NULL,

num\_civico INTEGER NOT NULL,

provincia VARCHAR(60) NOT NULL,

cod\_f INTEGER NOT NULL,

FOREIGN KEY (cod\_f) REFERENCES FILIALE(cod\_f)

ON DELETE CASCADE ON UPDATE CASCADE,

CHECK(num\_civico > 0 ),

CHECK(costo\_orario > 0)

);

CREATE TABLE RIFORNIMENTO(

cod\_fornitore INTEGER NOT NULL,

cod\_rifornisce INTEGER NOT NULL,

PRIMARY KEY(cod\_fornitore, cod\_rifornisce),

FOREIGN KEY (cod\_fornitore) REFERENCES FILIALE(cod\_f)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (cod\_rifornisce) REFERENCES FILIALE(cod\_f)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE EVENTO(

nome VARCHAR(60) NOT NULL,

data DATE NOT NULL,

localita VARCHAR(60) NOT NULL,

cod\_f INTEGER NOT NULL,

PRIMARY KEY(nome, data),

FOREIGN KEY (cod\_f) REFERENCES FILIALE(cod\_f)

ON DELETE CASCADE ON UPDATE CASCADE,

CHECK(LENGTH(localita) >= 2),

CHECK(LENGTH(nome) >= 3)

);

CREATE TABLE ESPOSIZIONE(

nome\_cd VARCHAR(60) NOT NULL,

cod\_band INTEGER NOT NULL,

cod\_f INTEGER NOT NULL,

quantita INTEGER,

PRIMARY KEY(nome\_cd, cod\_band, cod\_f),

FOREIGN KEY (nome\_cd, cod\_band) REFERENCES CD\_MUSICALE(nome, cod\_band)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (cod\_f) REFERENCES FILIALE(cod\_f)

ON DELETE CASCADE ON UPDATE CASCADE,

CHECK (quantita >= 0)

);

CREATE TABLE CLIENTE(

cf CHAR(16) PRIMARY KEY,

nome VARCHAR(60) NOT NULL,

cognome VARCHAR(60) NOT NULL,

telefono VARCHAR(10),

spesa\_totale DECIMAL(10,3) NOT NULL

);

CREATE TABLE PRESTITO(

data date NOT NULL,

cf CHAR(16) NOT NULL,

nome\_cd VARCHAR(60) NOT NULL,

cod\_band INTEGER NOT NULL,

costo DECIMAL (10,3) NOT NULL,

cod\_f INTEGER NOT NULL,

PRIMARY KEY(cf, nome\_cd, cod\_band),

FOREIGN KEY (nome\_cd, cod\_band) REFERENCES CD\_MUSICALE(nome, cod\_band)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (cf) REFERENCES CLIENTE(cf)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (cod\_f) REFERENCES FILIALE(cod\_f)

ON DELETE CASCADE ON UPDATE CASCADE,

CHECK(costo > 0)

);

CREATE TABLE BUONO\_SCONTO(

cod\_b INTEGER PRIMARY KEY,

sconto DECIMAL(3,2),

valido BOOLEAN NOT NULL,

CHECK(sconto <= 1)

);

CREATE TABLE CARRELLO(

num INTEGER NOT NULL,

cf CHAR(16) NOT NULL,

data date NOT NULL,

importo\_totale DECIMAL(10,3),

cod\_f INTEGER NOT NULL,

PRIMARY KEY(num, cf),

FOREIGN KEY (cf) REFERENCES CLIENTE(cf)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (cod\_f) REFERENCES FILIALE(cod\_f)

ON DELETE CASCADE ON UPDATE CASCADE,

CHECK(importo\_totale >= 0),

CHECK(num > 0)

);

CREATE TABLE FA\_PARTE(

num INTEGER NOT NULL,

cf CHAR(16) NOT NULL,

nome\_cd VARCHAR(60) NOT NULL,

cod\_band INTEGER NOT NULL,

qta\_copie INTEGER NOT NULL,

PRIMARY KEY(num, cf, nome\_cd, cod\_band),

FOREIGN KEY (num, cf) REFERENCES CARRELLO(num, cf)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (nome\_cd, cod\_band) REFERENCES CD\_MUSICALE(nome, cod\_band)

ON DELETE CASCADE ON UPDATE CASCADE,

CHECK(qta\_copie > 0)

);

CREATE TABLE SOGGETTO(

cod\_b INTEGER PRIMARY KEY,

num INTEGER,

cf CHAR(16),

UNIQUE(num, cf),

FOREIGN KEY (num, cf) REFERENCES CARRELLO(num, cf)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (cod\_b) REFERENCES BUONO\_SCONTO(cod\_b)

ON DELETE CASCADE ON UPDATE CASCADE

);

# Query Inserimento

Di seguito sono riportate le query per l’inserimento dei dati puramente a scopo dimostrativo.

INSERT INTO CASA\_DISCOGRAFICA VALUES('Bertelsmann Music Group', '1987');

INSERT INTO CASA\_DISCOGRAFICA VALUES('EMI','1931');

INSERT INTO CASA\_DISCOGRAFICA VALUES('Sony Music','1991');

INSERT INTO CASA\_DISCOGRAFICA VALUES('PolyGram','1962');

INSERT INTO CASA\_DISCOGRAFICA VALUES('Universal Music Group','1934');

INSERT INTO BAND VALUES('234','Beatles','2000-11-22','Bertelsmann Music Group');

INSERT INTO BAND VALUES('4254','Anthrax','2001-3-2','EMI');

INSERT INTO BAND VALUES('12','Blink 182','1987-5-25','Bertelsmann Music Group');

INSERT INTO BAND VALUES('53','Bon jovi','2005-1-11','Universal Music Group');

INSERT INTO BAND VALUES('863','Chumbawamba','2000-3-4','EMI');

INSERT INTO BAND VALUES('48','The cure','2005-5-6','Universal Music Group');

INSERT INTO BAND VALUES('3','Depeche mode','2006-6-4','Bertelsmann Music Group');

INSERT INTO BAND VALUES('93','The doors','2007-11-3','EMI');

INSERT INTO BAND VALUES('354','Eagles','1988-1-22','Bertelsmann Music Group');

INSERT INTO BAND VALUES('111','Evereything but the girl','2015-3-5','PolyGram');

INSERT INTO BAND VALUES('232','Foo fighters','2012-3-4','PolyGram');

INSERT INTO BAND VALUES('222','Genesis','2013-2-1','Bertelsmann Music Group');

INSERT INTO BAND VALUES('331','Green day','2000-11-2','Bertelsmann Music Group');

INSERT INTO BAND VALUES('89','Jamiroquai','2004-3-3','Sony Music');

INSERT INTO STUDIO\_DI\_REGISTRAZIONE VALUES('Lemon studio','Roma','41019','Soliera','23','Modena');

INSERT INTO STUDIO\_DI\_REGISTRAZIONE VALUES('Donkey','Croce lama','41019','Modena','43','Modena');

INSERT INTO STUDIO\_DI\_REGISTRAZIONE VALUES('Rock studio','Papa giovanni','41019','Limidi','33','Modena');

INSERT INTO STUDIO\_DI\_REGISTRAZIONE VALUES('Ross','Carpi ravarino','41019','Carpi','234','Modena');

INSERT INTO STUDIO\_DI\_REGISTRAZIONE VALUES('Veritas','Freccia','75938','Agrigento','123','Reggio Emilia');

INSERT INTO STUDIO\_DI\_REGISTRAZIONE VALUES('Riverdale studio','Arcobaleno','98538','Salerno','444','Bologna');

INSERT INTO STUDIO\_DI\_REGISTRAZIONE VALUES('Lake studio','Mulo','41019','Campogalliano','333','Modena');

INSERT INTO CD\_MUSICALE VALUES('La voce del padrone','234','Lemon studio','7','01:00:00','6','234','Soliera');

INSERT INTO CD\_MUSICALE VALUES('Una donna per amico','234','Lemon studio','8','01:00:00','6','234','Soliera');

INSERT INTO CD\_MUSICALE VALUES('Ovunque proteggi ','4254','Lemon studio','8','01:00:00','7','435','Soliera');

INSERT INTO CD\_MUSICALE VALUES('I buoni e i cattivi','4254','Lemon studio','9','01:20:00','7','555','Soliera');

INSERT INTO CD\_MUSICALE VALUES('WoW','12','Lemon studio','20','00:40:00','7','444','Soliera');

INSERT INTO CD\_MUSICALE VALUES('A sangue freddo','12','Donkey','12','01:00:00','7','324','Modena');

INSERT INTO CD\_MUSICALE VALUES('Dreamland','53','Donkey','5','00:40:00','8','643','Modena');

INSERT INTO CD\_MUSICALE VALUES('A Race with the Devil','53','Donkey','5','00:40:00','9','325','Modena');

INSERT INTO CD\_MUSICALE VALUES('Carboni','863','Donkey','6','00:45:00','9','545','Modena');

INSERT INTO CD\_MUSICALE VALUES('Darwin!','863','Rock studio','8','00:45:00','9','342','Limidi');

INSERT INTO CD\_MUSICALE VALUES('Pompa','48','Rock studio','10','00:45:00','9','355','Limidi');

INSERT INTO CD\_MUSICALE VALUES('Il tuffatore','48','Rock studio','11','00:47:00','7','354','Limidi');

INSERT INTO CD\_MUSICALE VALUES('Reset','3','Rock studio','12','00:47:00','8','666','Limidi');

INSERT INTO CD\_MUSICALE VALUES('Latin Lover','3','Rock studio','12','00:47:00','8','454','Limidi');

INSERT INTO CD\_MUSICALE VALUES('Il vile','93','Ross','13','00:47:00','7','234','Carpi');

INSERT INTO CD\_MUSICALE VALUES('Kinotto','93','Ross','15','00:48:00','7','565','Carpi');

INSERT INTO CD\_MUSICALE VALUES('Non e per sempre','354','Veritas','12','01:47:00','7','575','Agrigento');

INSERT INTO CD\_MUSICALE VALUES('Rockmantico','354','Veritas','6','00:43:00','7','345','Agrigento');

INSERT INTO CD\_MUSICALE VALUES('Ossigeno','111','Riverdale studio','5','00:40:00','8','345','Salerno');

INSERT INTO CD\_MUSICALE VALUES('Kanaglia','232','Riverdale studio','6','00:57:00','8','123','Salerno');

INSERT INTO CD\_MUSICALE VALUES('Fatti Sentire','222','Lake studio','6','00:47:00','8','765','Campogalliano');

INSERT INTO CD\_MUSICALE VALUES('Peter Pan','331','Lake studio','7','00:27:00','6','1111','Campogalliano');

INSERT INTO CD\_MUSICALE VALUES('Everything Is Love','89','Lake studio','8','00:47:00','6','455','Campogalliano');

INSERT INTO GENERE\_MUSICALE VALUES('Hard Rock','1980');

INSERT INTO GENERE\_MUSICALE VALUES('Pop','2010');

INSERT INTO GENERE\_MUSICALE VALUES('Heavy Metal','1980');

INSERT INTO GENERE\_MUSICALE VALUES('Blues','1960');

INSERT INTO ADESIONE VALUES('Hard Rock','234','La voce del padrone');

INSERT INTO ADESIONE VALUES('Hard Rock','234','Una donna per amico');

INSERT INTO ADESIONE VALUES('Hard Rock','4254','Ovunque proteggi ');

INSERT INTO ADESIONE VALUES('Hard Rock','4254','I buoni e i cattivi');

INSERT INTO ADESIONE VALUES('Hard Rock','12','WoW');

INSERT INTO ADESIONE VALUES('Hard Rock','12','A sangue freddo');

INSERT INTO ADESIONE VALUES('Hard Rock','53','Dreamland');

INSERT INTO ADESIONE VALUES('Heavy Metal','53','A Race with the Devil');

INSERT INTO ADESIONE VALUES('Heavy Metal','863','Carboni');

INSERT INTO ADESIONE VALUES('Heavy Metal','863','Darwin!');

INSERT INTO ADESIONE VALUES('Heavy Metal','48','Pompa');

INSERT INTO ADESIONE VALUES('Blues','48','Il tuffatore');

INSERT INTO ADESIONE VALUES('Blues','3','Reset');

INSERT INTO ADESIONE VALUES('Blues','3','Latin Lover');

INSERT INTO ADESIONE VALUES('Blues','93','Il vile');

INSERT INTO ADESIONE VALUES('Pop','93','Kinotto');

INSERT INTO ADESIONE VALUES('Pop','354','Non e per sempre');

INSERT INTO ADESIONE VALUES('Pop','354','Rockmantico');

INSERT INTO ADESIONE VALUES('Pop','111','Ossigeno');

INSERT INTO ADESIONE VALUES('Pop','232','Kanaglia');

INSERT INTO ADESIONE VALUES('Pop','222','Fatti Sentire');

INSERT INTO ADESIONE VALUES('Hard Rock','331','Peter Pan');

INSERT INTO ADESIONE VALUES('Hard Rock','89','Everything Is Love');

INSERT INTO MAGAZZINO VALUES('1','Cascata','41019','Modena','12','Modena');

INSERT INTO MAGAZZINO VALUES('2','Roma','34244','Salerno','3','Salerno');

INSERT INTO MAGAZZINO VALUES('3','Palladio','75938','Rolo','23','Palermo');

INSERT INTO DIRIGENTE VALUES('ABC','Daniele','Bruco','70','3557349485','Roma','41019','Soliera','32','Modena','15');

INSERT INTO DIRIGENTE VALUES('EFG','Matteo','Aguzzo','75','3348385839','Cielo','23542','Milano','43','Milano','18');

INSERT INTO DIRIGENTE VALUES('HIJ','Luigi','Sedo','75','3285434567','Sole','32454','Arezzo','23','Arezzo','20');

INSERT INTO FILIALE VALUES('10','0','059635483','Roma','41019','Modena','5','Modena','1');

INSERT INTO FILIALE VALUES('11','0','059837493','Sole','23424','Palermo','33','Palermo','2');

INSERT INTO FILIALE VALUES('12','0','059281293','Calo','75938','Rolo','56','Palermo','2');

INSERT INTO DIRIGE VALUES('10','ABC');

INSERT INTO DIRIGE VALUES('11','EFG');

INSERT INTO DIRIGE VALUES('12','HIJ');

INSERT INTO DIPENDENTE VALUES('QWE','Paolo','Tarozzi','25','123456788','croce','41019','Soliera','2','Modena','10');

INSERT INTO DIPENDENTE VALUES('RTY','Enrico','Sedoni','35','384293920','Croce lama','41019','Soliera','257','Modena','10');

INSERT INTO DIPENDENTE VALUES('UUI','Roberto','Pulizo','20','3456086234','Reti','23454','Roma','32','Roma','11');

INSERT INTO DIPENDENTE VALUES('IOP','Giulio','Sternieri','25','098765432','Viazzolo','41019','Soliera','33','Modena','11');

INSERT INTO DIPENDENTE VALUES('ASD','Isabella','Marchi','33','3928304933','Luigi galvani','75938','Correggio','12','Reggio Emilia','12');

INSERT INTO DIPENDENTE VALUES('FGH','Elisa','Carducci','31','3928342333','Paolo Sarti','75938','Correggio','32','Reggio Emilia','12');

INSERT INTO RIFORNIMENTO VALUES('11','12');

INSERT INTO RIFORNIMENTO VALUES('12','11');

INSERT INTO EVENTO VALUES('Rock in modena','2010-02-05','Modena','10');

INSERT INTO EVENTO VALUES('Blues insieme','2011-03-02','Modena','10');

INSERT INTO EVENTO VALUES('Metal in palermo','2010-05-12','Palermo','11');

INSERT INTO EVENTO VALUES('Music festival','2012-02-22','Salerno','12');

INSERT INTO EVENTO VALUES('Fosh Fest','2013-01-12','Modena','10');

INSERT INTO ESPOSIZIONE VALUES('La voce del padrone','234','10','15');

INSERT INTO ESPOSIZIONE VALUES('Una donna per amico','234','10','15');

INSERT INTO ESPOSIZIONE VALUES('Ovunque proteggi ','4254','10','15');

INSERT INTO ESPOSIZIONE VALUES('I buoni e i cattivi','4254','10','15');

INSERT INTO ESPOSIZIONE VALUES('WoW','12','10','15');

INSERT INTO ESPOSIZIONE VALUES('Dreamland','53','10','15');

INSERT INTO ESPOSIZIONE VALUES('Carboni','863','10','15');

INSERT INTO ESPOSIZIONE VALUES('Darwin!','863','10','15');

INSERT INTO ESPOSIZIONE VALUES('Reset','3','10','15');

INSERT INTO ESPOSIZIONE VALUES('Il vile','93','10','15');

INSERT INTO ESPOSIZIONE VALUES('Kinotto','93','10','15');

INSERT INTO ESPOSIZIONE VALUES('Rockmantico','354','10','15');

INSERT INTO ESPOSIZIONE VALUES('Peter Pan','331','10','15');

INSERT INTO ESPOSIZIONE VALUES('La voce del padrone','234','11','15');

INSERT INTO ESPOSIZIONE VALUES('Una donna per amico','234','11','15');

INSERT INTO ESPOSIZIONE VALUES('Ovunque proteggi ','4254','11','15');

INSERT INTO ESPOSIZIONE VALUES('WoW','12','11','15');

INSERT INTO ESPOSIZIONE VALUES('Dreamland','53','11','15');

INSERT INTO ESPOSIZIONE VALUES('Carboni','863','11','15');

INSERT INTO ESPOSIZIONE VALUES('Darwin!','863','11','15');

INSERT INTO ESPOSIZIONE VALUES('Reset','3','11','15');

INSERT INTO ESPOSIZIONE VALUES('Kinotto','93','11','15');

INSERT INTO ESPOSIZIONE VALUES('Rockmantico','354','11','15');

INSERT INTO ESPOSIZIONE VALUES('Peter Pan','331','11','15');

INSERT INTO ESPOSIZIONE VALUES('La voce del padrone','234','12','15');

INSERT INTO ESPOSIZIONE VALUES('Una donna per amico','234','12','15');

INSERT INTO ESPOSIZIONE VALUES('I buoni e i cattivi','4254','12','15');

INSERT INTO ESPOSIZIONE VALUES('WoW','12','12','15');

INSERT INTO ESPOSIZIONE VALUES('Dreamland','53','12','15');

INSERT INTO ESPOSIZIONE VALUES('Carboni','863','12','15');

INSERT INTO ESPOSIZIONE VALUES('Everything Is Love','89','12','15');

INSERT INTO ESPOSIZIONE VALUES('Kinotto','93','12','15');

INSERT INTO ESPOSIZIONE VALUES('Rockmantico','354','12','15');

INSERT INTO ESPOSIZIONE VALUES('Peter Pan','331','12','15');

INSERT INTO CLIENTE VALUES('ABCD','Luigi','Sanfilippo','09867764','0');

INSERT INTO CLIENTE VALUES('BCDE','Rulo','Pastulu','5372934','0');

INSERT INTO CLIENTE VALUES('FGHI','Francesco','Sanfilippo','5454555','0');

INSERT INTO CLIENTE VALUES('GFDT','Mario','Bianchi','67523473','0');

INSERT INTO CLIENTE VALUES('JEMCC','Paolo','Rossi','848473649','0');

INSERT INTO CLIENTE VALUES('SKEKS','Oca','Pule','73736393','0');

INSERT INTO CLIENTE VALUES('ETNSM','Lorella','Casella','09867334','0');

INSERT INTO CLIENTE VALUES('OWNCJC','Matteo','Arizona','33434223','0');

INSERT INTO CLIENTE VALUES('KMDKNE','SIngh','India','56234223','0');

INSERT INTO CLIENTE VALUES('GXNWLD','Enrico','Lupo','1233445','0');

INSERT INTO PRESTITO VALUES('2010-05-06','ABCD','Peter Pan','331','5','12');

INSERT INTO PRESTITO VALUES('2010-05-06','ABCD','Rockmantico','354','5','12');

INSERT INTO PRESTITO VALUES('2011-03-06','BCDE','Peter Pan','331','5','12');

INSERT INTO PRESTITO VALUES('2013-08-11','BCDE','I buoni e i cattivi','4254','5','12');

INSERT INTO PRESTITO VALUES('2010-05-06','FGHI','La voce del padrone','234','5','11');

INSERT INTO PRESTITO VALUES('2010-05-06','FGHI','Reset','3','5','11');

INSERT INTO BUONO\_SCONTO VALUES('1234','0.10','true');

INSERT INTO BUONO\_SCONTO VALUES('456','0.10','true');

INSERT INTO BUONO\_SCONTO VALUES('789','0.15','true');

INSERT INTO BUONO\_SCONTO VALUES('3421','0.15','true');

INSERT INTO BUONO\_SCONTO VALUES('3245','0.20','true');

INSERT INTO BUONO\_SCONTO VALUES('12345','0.20','true');

INSERT INTO CARRELLO VALUES('1','GFDT','2009-08-07','0','10');

INSERT INTO CARRELLO VALUES('2','JEMCC','2002-03-07','0','10');

INSERT INTO CARRELLO VALUES('3','SKEKS','2008-03-11','0','11');

INSERT INTO CARRELLO VALUES('4','ETNSM','2008-06-22','0','10');

INSERT INTO CARRELLO VALUES('5','OWNCJC','2008-06-07','0','10');

INSERT INTO CARRELLO VALUES('6','KMDKNE','2005-12-07','0','11');

INSERT INTO CARRELLO VALUES('1','GXNWLD','2008-03-17','0','12');

INSERT INTO CARRELLO VALUES('2','ABCD','2009-05-27','0','12');

INSERT INTO CARRELLO VALUES('3','BCDE','2008-06-07','0','12');

INSERT INTO CARRELLO VALUES('4','KMDKNE','2001-06-07','0','12');

INSERT INTO FA\_PARTE VALUES('1','GFDT','Peter Pan','331','3');

INSERT INTO FA\_PARTE VALUES('1','GFDT','WoW','12','3');

INSERT INTO FA\_PARTE VALUES('2','JEMCC','Peter Pan','331','2');

INSERT INTO FA\_PARTE VALUES('2','JEMCC','WoW','12','2');

INSERT INTO FA\_PARTE VALUES('3','SKEKS','La voce del padrone','234','1');

INSERT INTO FA\_PARTE VALUES('3','SKEKS','Una donna per amico','234','1');

INSERT INTO FA\_PARTE VALUES('4','ETNSM','Dreamland','53','4');

INSERT INTO FA\_PARTE VALUES('4','ETNSM','Peter Pan','331','4');

INSERT INTO FA\_PARTE VALUES('5','OWNCJC','La voce del padrone','234','1');

INSERT INTO FA\_PARTE VALUES('5','OWNCJC','WoW','12','1');

INSERT INTO FA\_PARTE VALUES('6','KMDKNE','Peter Pan','331','1');

INSERT INTO FA\_PARTE VALUES('6','KMDKNE','WoW','12','1');

INSERT INTO FA\_PARTE VALUES('1','GXNWLD','I buoni e i cattivi','4254','2');

INSERT INTO FA\_PARTE VALUES('2','ABCD','Peter Pan','331','1');

INSERT INTO FA\_PARTE VALUES('3','BCDE','WoW','12','2');

INSERT INTO FA\_PARTE VALUES('4','KMDKNE','La voce del padrone','234','1');

INSERT INTO SOGGETTO VALUES('1234','1','GFDT');

INSERT INTO SOGGETTO VALUES('456','2','JEMCC');

INSERT INTO SOGGETTO VALUES('789','3','BCDE');

INSERT INTO SOGGETTO VALUES('12345','1','GXNWLD');

# Triggers e stored procedure

Di seguito sono implementati i trigger principali necessari al corretto funzionamento del database. Sono inotre implementate due stored procedure “extra” che gestiscono il rifornimento dei cd.

/\* aggiornamento del "costo personale" di una filiale dopo “l’affidamento” a un dirigente \*/

CREATE OR REPLACE FUNCTION aggiorna\_costo\_personale() RETURNS trigger AS $$

DECLARE

costo DECIMAL(10,3);

BEGIN

costo = (SELECT costo\_orario from DIRIGENTE WHERE cf = NEW.cf\_dir);

UPDATE FILIALE

SET costo\_personale = costo\_personale + costo

WHERE cod\_f = NEW.cod\_f;

RETURN NEW;

END;

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER COSTO\_PERSONALE

AFTER INSERT ON DIRIGE

FOR EACH ROW EXECUTE PROCEDURE aggiorna\_costo\_personale();

/\* aggiornamento del "costo personale" di una filiale dopo l’assunzione di un dipendente \*/

CREATE OR REPLACE FUNCTION aggiorna\_costo\_personale2() RETURNS trigger AS $$

BEGIN

UPDATE FILIALE

SET costo\_personale = costo\_personale + NEW.costo\_orario

WHERE cod\_f = NEW.cod\_f;

RETURN NEW;

END;

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER COSTO\_PERSONALE2

AFTER INSERT ON DIPENDENTE

FOR EACH ROW EXECUTE PROCEDURE aggiorna\_costo\_personale2();

/\* aggiornamento del "costo personale" di una filiale dopo il licenziamento di un dipendente \*/

CREATE OR REPLACE FUNCTION riduci\_costo\_personale2() RETURNS trigger AS $$

BEGIN

UPDATE FILIALE

SET costo\_personale = costo\_personale - OLD.costo\_orario

WHERE cod\_f = OLD.cod\_f;

RETURN OLD;

END;

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER RIDUCI\_COSTO\_PERSONALE2

AFTER DELETE ON DIPENDENTE

FOR EACH ROW EXECUTE PROCEDURE riduci\_costo\_personale2();

/\* aggiornamento del "costo personale" di una filiale dopo la rimozione di un dirigente \*/

CREATE OR REPLACE FUNCTION riduci\_costo\_personale() RETURNS trigger AS $$

DECLARE

costo DECIMAL(10,3);

BEGIN

costo = (SELECT costo\_orario from DIRIGENTE WHERE cf = OLD.cf\_dir);

UPDATE FILIALE

SET costo\_personale = costo\_personale - costo

WHERE cod\_f = OLD.cod\_f;

RETURN OLD;

END;

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER RIDUCI\_COSTO\_PERSONALE

AFTER DELETE ON DIRIGE

FOR EACH ROW EXECUTE PROCEDURE riduci\_costo\_personale();

/\* trigger che aggiorna la spesa totale del cliente dopo l’effettuo di un prestito \*/

CREATE OR REPLACE FUNCTION aggiorna\_spesa\_totale\_p() RETURNS trigger AS $$

BEGIN

UPDATE CLIENTE

SET spesa\_totale = spesa\_totale + NEW.costo

WHERE cf = NEW.cf;

RETURN NEW;

END;

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER AGGIORNA\_SPESA\_TOTALE\_P

AFTER INSERT ON PRESTITO

FOR EACH ROW EXECUTE PROCEDURE aggiorna\_spesa\_totale\_p();

/\* trigger che aggiorna la spesa totale di un cliente in base ai prodotti inseriti all’interno di un carrello acquistato \*/

CREATE OR REPLACE FUNCTION aggiorna\_spesa\_totale\_acq() RETURNS trigger AS $$

DECLARE

prezzo DECIMAL(10,3);

BEGIN

prezzo= (select prezzo\_cad

from cd\_musicale

where nome = NEW.nome\_cd and cod\_band = NEW.cod\_band);

UPDATE CLIENTE

SET spesa\_totale = spesa\_totale + (prezzo\*NEW.qta\_copie)

WHERE cf = NEW.cf;

RETURN NEW;

END;

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER AGGIORNA\_SPESA\_TOTALE\_ACQ

AFTER INSERT ON FA\_PARTE

FOR EACH ROW EXECUTE PROCEDURE aggiorna\_spesa\_totale\_acq();

/\* trigger che diminuisce la quantita di cd disponibili di una filiale dopo un prestito \*/

CREATE OR REPLACE FUNCTION riduce\_quantita\_cd\_pr() RETURNS trigger AS $$

BEGIN

UPDATE ESPOSIZIONE

SET quantita = quantita - 1

WHERE cod\_f = NEW.cod\_f and nome\_cd = NEW.nome\_cd

and cod\_band = NEW.cod\_band;

RETURN NEW;

END;

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER RIDUCE\_QT\_CD\_PR

AFTER INSERT ON PRESTITO

FOR EACH ROW EXECUTE PROCEDURE riduce\_quantita\_cd\_pr();

/\* trigger che diminuisce la quantita di cd disponibili di una filiale dopo l’inserimento di questi all’interno di un carrello \*/

CREATE OR REPLACE FUNCTION riduce\_quantita\_cd\_acq() RETURNS trigger AS $$

DECLARE

codf INTEGER;

BEGIN

codf = (select distinct cod\_f

from carrello c

where NEW.num = c.num and NEW.cf = c.cf

);

UPDATE ESPOSIZIONE

SET quantita = quantita - NEW.qta\_copie

WHERE cod\_f = codf and nome\_cd = NEW.nome\_cd

and cod\_band = NEW.cod\_band;

RETURN NEW;

END;

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER RIDUCE\_QT\_CD\_ACQ

AFTER INSERT ON FA\_PARTE

FOR EACH ROW EXECUTE PROCEDURE riduce\_quantita\_cd\_acq();

/\* trigger che aggiorna il dato derivato “importo totale” di un carrello in base ai cd inseriti all’interno di quest’ultimo \*/

CREATE OR REPLACE FUNCTION aumenta\_importo\_totale() RETURNS trigger AS $$

DECLARE

prezzo INTEGER;

BEGIN

prezzo = (select prezzo\_cad

from cd\_musicale

where nome = NEW.nome\_cd and cod\_band = NEW.cod\_band);

UPDATE CARRELLO

SET importo\_totale = importo\_totale + (prezzo\*NEW.qta\_copie)

WHERE num = NEW.num and cf = NEW.cf;

RETURN NEW;

END;

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER AUMENTA\_IMPORTO\_TOTALE

AFTER INSERT ON FA\_PARTE

FOR EACH ROW EXECUTE PROCEDURE aumenta\_importo\_totale() ;

/\* trigger che verifica la validità di un buono se utilizzato in un carrello. Se il buono viene utilizzato correttamente allora perderà la validità \*/

CREATE OR REPLACE FUNCTION controllo\_buono() RETURNS trigger AS $$

DECLARE

valido BOOLEAN;

prezzo\_sconto DECIMAL(10,3);

BEGIN

valido = (select b.valido

from buono\_sconto b

where b.cod\_b = NEW.cod\_b);

prezzo\_sconto = ((select importo\_totale

from carrello c

where cf = NEW.cf and num = NEW.num) \* (select sconto

from buono\_sconto

where cod\_b = NEW.cod\_b));

IF valido = false THEN RAISE EXCEPTION 'buono non valido';

ELSE IF valido = true THEN

UPDATE CARRELLO

SET importo\_totale = importo\_totale - prezzo\_sconto

WHERE num = NEW.num and cf = NEW.cf;

UPDATE BUONO\_SCONTO

SET valido = 'false'

WHERE cod\_b = NEW.cod\_b;

END IF;

END IF;

RETURN NEW;

END;

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER CONTROLLO\_BUONO

AFTER INSERT OR UPDATE ON SOGGETTO

FOR EACH ROW EXECUTE PROCEDURE controllo\_buono() ;

/\* stored procedure per il rifornimento standard con il magazzino di un determinato cd in una filiale. \*/

CREATE OR REPLACE FUNCTION rifornimento(quant INTEGER, nomecd VARCHAR, codband INTEGER, codf INTEGER) RETURNS INTEGER AS $$

DECLARE

Q INTEGER;

BEGIN

UPDATE ESPOSIZIONE

SET quantita = quantita + quant

WHERE cod\_f = codf and nome\_cd = nomecd and cod\_band = codband;

Q = (select quantita from esposizione WHERE cod\_f = codf and nome\_cd = nomecd and cod\_band = codband );

RETURN Q;

END;

$$ LANGUAGE 'plpgsql';

/\* stored procedure per il rifornimento di cd tra due determinate filiali. E’ utile quando si ha una necessità urgente di determinati cd e il magazzino è saturo di richieste di rifornimento. Lo scambio di cd tra due filiali è certamente più veloce quindi del rifornimento magazzino-filiale. E’ necessario inoltre tenere a mente determinati vincoli imposti dal problema:

1. La filiale può rifornirsi con un altra filiale solo se non è presente un magazzino nella provincia in cui è situata, in quel caso infatti il rifornimento del cd avverrebbe comunque in tempi sufficientemente brevi
2. La filiale scelta per il rifornimento deve trovarsi all’interno della lista delle filiali di rifornimento per la filiale (cioè all’interno dell’auto associazione “rifornimento”)
3. La filiale di rifornimento scelta deve avere una sufficiente disponibilità di quel cd musicale

\*/

CREATE OR REPLACE FUNCTION rifornimento\_filiali(quant INTEGER, nomecd VARCHAR, codband INTEGER, codf INTEGER, codf2 INTEGER) RETURNS INTEGER AS $$

DECLARE

codM INTEGER;

provM VARCHAR(60);

q2 INTEGER; /\* quantita cd disponibili della filiale scelta per il rifornimento\*/

q\_n INTEGER; /\* nuova quantita di cd disponibili per la filiale \*/

BEGIN

codM = (select cod\_m from filiale f where f.cod\_f = codf);

provM = (select provincia from magazzino m where m.cod\_m =codM );

IF provM = (select provincia from filiale f where f.cod\_f = codf) THEN RAISE EXCEPTION 'effettuare la procedura di rifornimento standard, il magazzino è sufficientemente vicino alla filiale per fornire il prodotto in tempi brevi';

END IF;

q2 = (select quantita from esposizione e where e.cod\_f = codf2 and nome\_cd = nomecd and cod\_band = codband);

/\* la filiale scelta per il rifornimento deve essere presente nella lista di filiali di rifornimento per quella determinata filiale (dentro l'associazione "rifornimento") \*/

IF (select count(\*) from rifornimento r where r.cod\_rifornisce = codf2 and r.cod\_fornitore = codf) < 1 THEN RAISE EXCEPTION 'la filiale scelta di rifornimento non è presente nella lista di filiali di rifornimento disponibili';

END IF;

IF (q2 - quant) < 0 THEN RAISE EXCEPTION 'quantita di cd disponibili della filiale di rifornimento non sufficiente';

ELSE

UPDATE ESPOSIZIONE

SET quantita = quantita + quant

WHERE nome\_cd = nomecd and cod\_band = cod\_band and cod\_f = codf;

UPDATE ESPOSIZIONE

SET quantita = quantita - quant

WHERE nome\_cd = nomecd and cod\_band = cod\_band and cod\_f = codf2;

END IF;

q\_n = (select quantita from esposizione e where nome\_cd = nomecd and cod\_band = cod\_band and cod\_f = codf);

RETURN q\_n;

END;

$$ LANGUAGE 'plpgsql';