SUPERMERCATO

Peca Marco 182746
Soprano Claudio 181099

INDICE:

1.Introduzione

Il progetto in esame consiste nella realizzazione di una base di dati per la gestione di un supermercato. La realizzazione deve prevedere la gestione degli impiegati del supermercato, dei reparti presenti nel supermercato, dei prodotti presenti nei reparti, dei fornitori dei prodotti, degli ordini fatti ai fornitori, delle vendite effettuate con scontrini e dei clienti ed i loro premi.

Il primo problema da risolvere era l'analisi e la disambiguazione del testo. Non appena abbiamo trovato concetti incompleti, ambigui o incoerenti abbiamo effettuato una disambiguazione.

Abbiamo proseguito poi con la *progettazione concettuale* della base di dati, rappresentando in modo naturale i dati di interesse per il progetto. Per tale realizzazione abbiamo usato la strategia **Mista** ovvero la tecnica che ci permette di usare tutte le strategie realizzative (**Inside-Out**, **Bottom-up** e **Top-down**). Inizialmente abbiamo individuato i concetti più importanti per il nostro problema, per poi proseguire a "macchia d'olio", partendo dalle singole relazioni per arrivare alle tabelle definite ed infine abbiamo raggruppati relazione o entita' dove necessario.

Una volta prese in considerazione tutte le problematiche del caso, siamo passati alla costruzione del **Modello E-R** (Entità-Relazione) indicando le Entità, con i relativi attributi e identificatori (Primary Key)e le relazioni con le varie cardinalità vigenti.
Abbiamo controllato che si rispettassero le proprietà generali (correttezza, completezza, leggibilità, minimalità) per realizzare uno schema concettuale di buona qualità.

Dopo la progettazione concettuale abbiamo proseguito con la *progettazione logica*, avvicinandoci il più possibile alla fase implementativa della base di dati, andando ad effettuare un'ottimizzazione del Modello E-R analizzando le ridondanze, eliminando le generalizzazioni, partizionando o accorpando le entità e le relazioni la dove pensavamo fosse necessario ed infine scegliendo in maniera definitiva le chiavi primarie e le chiavi esterne.

Infine siamo passati alla *progettazione fisica*, producendo il vero e proprio schema fisico del database, applicando tutte i vincoli preposti e cercando di rendere il più possibile la gestione degli "eventi" veritiera.
Per la realizzazione abbiamo utilizzato come DBMS (Data Base Management System) **MySql** Ver 5.1.31, mentre per la realizzazione grafica degli opportuni grafi nei passi suddetti abbiamo utilizzato Microsoft **Visio** 2007.

2.Specifiche del problema

Nell'analisi del problema posto, abbiamo applicato la suddivisione in blocchi, evidenziando le parole chiave e eliminando le parti meno significative.

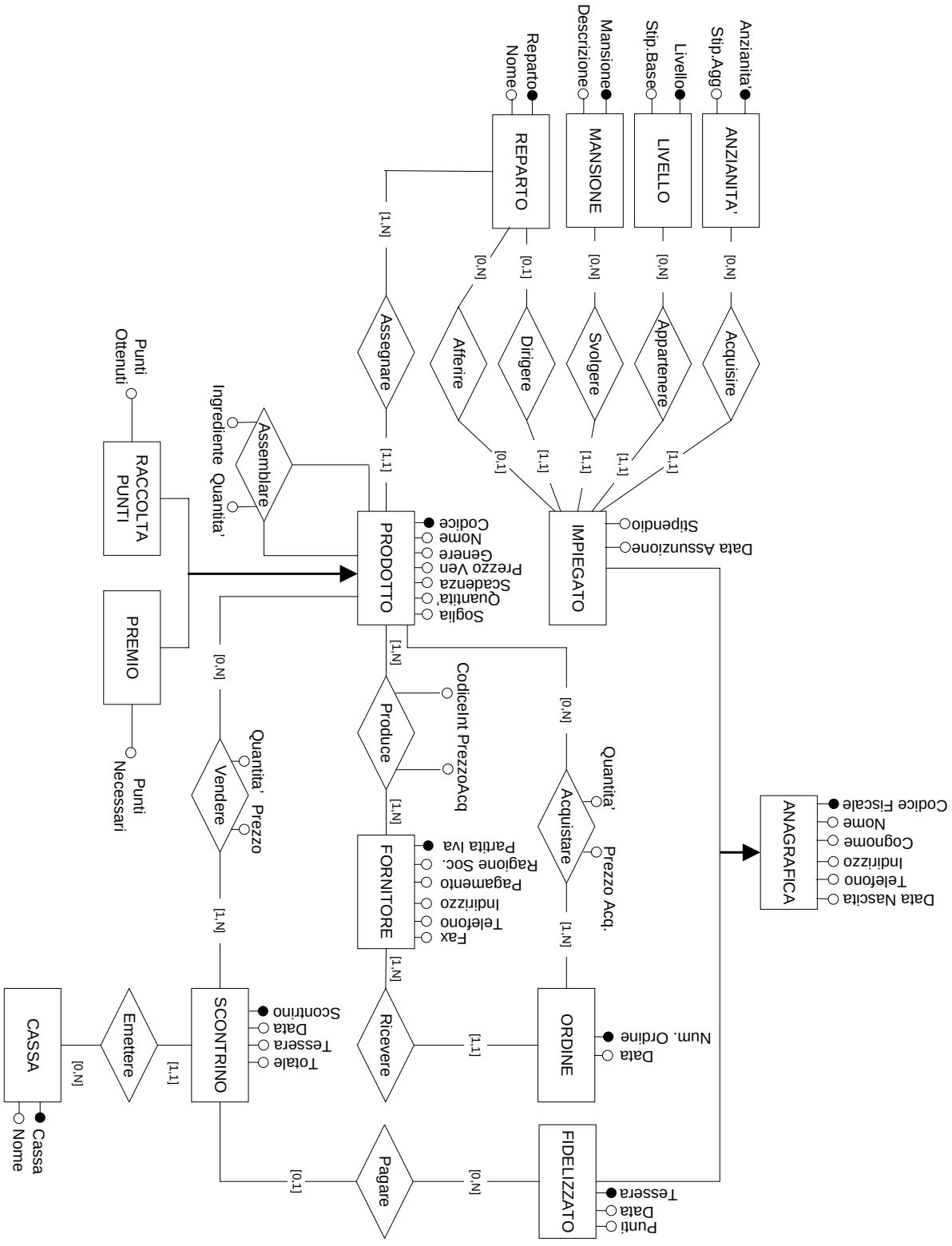
1 2 3	Il supermercato è strutturato in una serie di "Reparti". Ogni reparto ha un nome, un responsabile e una serie di impiegati, ognuno con una mansione specifica.
4 5 6 7	Di ogni "Impiegato" vogliamo memorizzare tutti i principali dati anagrafici […], oltre alla data di assunzione. Ogni impiegato lavora in un determinato reparto ed ha un particolare livello (che ne determina lo stipendio insieme all'anzianita' di servizio).
8 9 10 11	Il supermercato mette in vendita una serie di "Prodotti", dei quali vogliamo memorizzare: nome, genere, il prezzo al pubblico, la scadenza, il reparto in cui si trova, la quantita' presente in magazzino e la soglia minima necessaria prima di riordinarlo. Ad ogni prodotto e' assegnato un codice interno.
12 13 14 15	Alcuni prodotti non sono acquistati direttamente dai fornitori, ma "Assemblati" direttamente nel supermercato a partire da una serie di materie prime. Per questi assemblati vogliamo conoscere anche tutti i prodotti ingredienti e la quantità di essi necessaria alla loro preparazione.
16 17 18 19	La base di dati contiene una lista dei "Fornitori" a cui il supermercato fa riferimento. Per ogni fornitore vogliamo conoscere la ragione sociale, la partita IVA, la modalità di pagamento richiesta, oltre all'indirizzo e al numero di telefono e di fax.
20 21 22	Ogni fornitore può fornire, a un particolare prezzo, uno o più dei prodotti venduti dal supermercato, identificandoli con un proprio codice da indicare negli "Ordini".
23 24 25 26 27	Per i clienti del supermercato, alcuni di essi possono essere "Fidelizzati" perché hanno richiesto una tessera, che permette loro di accumulare punti con gli acquisti e ottenere premi. I clienti con tessera sono registrati con i loro dati anagrafici, il numero di tessera ed a ciascuno è associato il numero di punti correntemente accumulati.
28 29 30	I "Premi" delle raccolte punti sono prodotti del supermercato. Per ciascun Premio, oltre al prodotto associato, vogliamo conoscere il numero di punti necessari ad ottenerlo. I punti si ottengono acquistando particolari prodotti.

E’ necessario mantenere una “Lista” dei prodotti che partecipano alla raccolta punti, unitamente al numero di punti ottenibili con il loro acquisto. Ogni volta che il cliente fidelizzato ritira un premio, vengono scalati i corrispondenti punti dalla sua tessera.

I “registratori di cassa” del nostro supermercato ogni volta che un prodotto viene venduto, registrano i dati della vendita […], unitamente al numero di tessera del cliente al quale è stato venduto, se questo ne è in possesso.

Contemporaneamente, vengono aggiornate la disponibilità del prodotto ed il numero di punti del cliente (se il prodotto dava diritto a dei punti).

3.Progettazione Concettuale utilizzando lo schema E-R



Il Modello E-R e' stato realizzato con i seguenti passi:

- Creazione dell'entita' Anagrafica contenente gli attributi in comune tra Fidelizzati e Impiegati: Nome, Cognome, Indirizzo, Telefono, Codice Fiscale

- Creazione della generalizzazione Impiegato contenente la Data di Assunzione (riga 5) e lo Stipendio (riga 6) e della generalizzazione Fidelizzato contenente Tessera, Data e Punti (righe dalle 23 a 26) dall'entita' Anagrafica. Tali generalizzazioni sono di tipo Totale ed esclusiva in quanto un Anagrafica appartiene o ad Impiegato o a Fidelizzato

- Creazione delle relazioni tra l'entita' Impiegati e le entita' Reparto, Mansione, Anzianita, Livello (righe dalla 1 alla 7)

- Creazione dell'entita' Prodotto (riga 8) con gli attributi Codice, Nome, Genere, PrezzoVen, Scadenza, Quantita e Soglia (righe 9-11) legata al Reparto (riga 9) e alla relazione Assemblare relativa ai prodotti assemblati (riga 13) con attributi Ingrediente e Quantita' (riga 15)

- Creazione di due generalizzazioni dell'entita' Prodotto relative alla Raccolta Punti e Premio (righe 24,25) con i relativi attributi

- Creazione di un entita' Fornitore con gli attributi Partita Iva, Ragione Sociale, Pagamento, Indirizzo, Telefono e Fax (righe 16-19) e l'abbiamo legata all'entita' Prodotto con la relazione Produce (avente attributi Codice Int e Prezzo Acquisto)

- Creazione dell'entita' Ordine (righe 20-22) con gli attributi Numero Ordine e Data per poter differenziare i vari ordini. L'abbiamo legata con la relazione Acquistare (avente attributi Quantita' e Prezzo Acquisto) all'entita Prodotto e con la relazione Ricevere all'entita Fornitore.

- Creazione delle entita' Cassa e Scontrino e le abbiamo legate con la relazione Emettere.

- Creazione della relazione Pagare tra Scontrino e Fidelizzato

- Ed in ultimo creazione della relazione Vendere tra Scontrino e Prodotto, che serve a vendere la merce venduta ed ad aggiornare le scorte di magazzino (riga 38)

Per ciascuna relazione sono state individuate le relative cardinalita'.

4.Formalizzazione dei vincoli

Regole di vincolo:

RV1 - In “Anagrafica” *si devono* per forza indicare i campi “ID_CodiceFisc”, “Nome”, “Cognome”, “Indirizzo”, “DataNascita” e “Tipo”;

RV2 - In “Anagrafica” il campo “Tipo” *deve* appartenere al seguente dominio: “I” (impiegato) o “F” (fidelizzato);

RV3 - In “Livelli” *si devono* obbligatoriamente inserire i valori nei campi “ID_Livello” e “StipendioBase”;

RV4 - In “Anzianita” *si devono* obbligatoriamente inserire per forza i valori nei campi “ID_Anzianita” e “StipendioAgg”;

RV5 - In “Mansioni” *si devono* obbligatoriamente inserire i valori nei campi “ID_Mansione” e “Descrizione”;

RV6 - In “Impiegati” *si devono* obbligatoriamente indicare i campi “ID_Impiegato”, “ID_Reparto”, “ID_Mansione”, “ID_Livello”, “ID_Anzianita”, “DataAssunzione”, “Stipendio”;

RV7 - In “Impiegati” *ci devono* essere record relativi a persone registrate in “Anagrafica” che hanno come “Tipo” il record “I”;

RV8 - In “Impiegati” la data di assunzione *non può essere* successiva a quella odierna;

RV9 - In “Reparti” *bisogna* obbligatoriamente inserire i valori nei campi “ID_Reparto” e “Nome”;

RV10 - Ogni reparto ha un solo responsabile;

RV11 - Due reparti non possono avere lo stesso responsabile;

RV12 - In “Prodotti” *vanno indicati* i valori nei campi “ID_Prodotto”, “ID_Reparto”, “Nome”, “Genere”, “PrezzoVen”, “Quantita”, “Soglia”, “Assemblato”, “PuntiRaccolta” e “PuntiNecessari”;

RV13 - *Non ci possono essere* prodotti con nomi uguali;

RV14 - La scadenza di un prodotto *non può essere* antecedente alla data odierna;

RV15 - In “Prodotti” il campo “Assemblato” *deve* avere record appartenenti al dominio: “1” (si) e “0” (no);

RV16 - I prodotti che non partecipano alla raccolta punti *devono* avere il campo “PuntiRaccolta” pari a 0;

RV17 - I prodotti che non sono premi *devono* avere il campo “PuntiNecessari” pari a 0;

RV18 - In “Relaz_Prod_Assemb” *bisogna* necessariamente indicare i campi “ID_Assemblato” e “ID_Ingrediente”;

RV19 - In “Relaz_Prod_Assemb” nel campo “ID_Assemblato” *vanno* inseriti soltanto record di prodotti con il campo “Assemblato” pari a “1”;

RV20 - In “Relaz_Prod_Assemb” nel campo “ID_Ingrediente” *vanno* inseriti soltanto recordi di prodotti con il campo “Assemblato” pari a “0”;

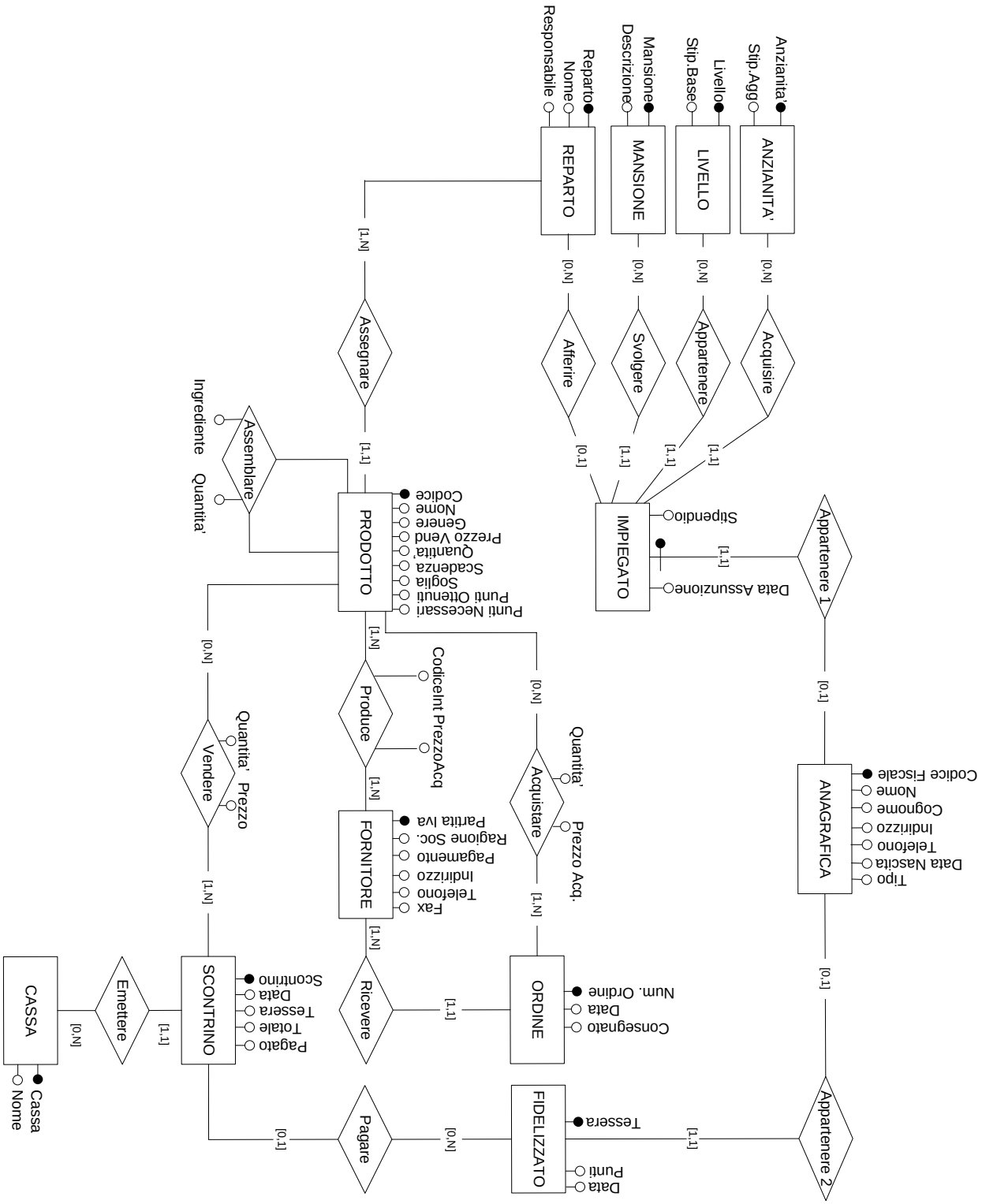
RV21 - In “Fornitori” *si devono* per forza indicare i campi “ID_PartitaIVA”, “RagioneSoc”, “Pagamento” e “Indirizzo”;

RV22 - In “Relaz_Prod_Forn” *si devono* obbligatoriamente inserire i valori nei campi “ID_CodiceInt”, “ID_Fornitore”, “ID_Prodotto”, “PrezzoAcq”;

RV23 - In “Ordini” *bisogna* inserire i valore nei campi “ID_Ordine”, “ID_Fornitore”, “Data”, “Consegnato”;
RV24 - La data dell’ordine *non può essere* antecedente alla data corrente;
RV25 - In “Dettaglio_Ordini” *vanno* necessariamente indicati i campi “ID_Ordine”, “ID_CodiceInt”, “ID_Prodotto”, “Quantita” e “PrezzoAcq”;
RV26 - Gli ordini *vanno* fatti su prodotti che i fornitori possono venderci;
RV27 - In “Fidelizzati” *bisogna* indicare i campi “ID_Fidelizzato”, “ID_Tessera”, “Data” e “PuntiAttuali”;
RV28 - In “Fidelizzati” *ci devono* essere record relativi a persone registrate in “Anagrafica” che hanno come “Tipo” il record “F”;
RV29 - La data di registrazione di un cliente fidelizzato *non può essere* antecedente la data corrente;
RV30 - In “Casse” *vanno* necessariamente inseriti i valori nei campi “ID_Cassa” e “Nome”;
RV31 - In “Scontrini” *vanno* indicati obbligatoriamente i campi “ID_Scontrino”, “ID_Cassa”, “Data”, “Totale” e “Pagato”;
RV32 - La data dello scontrino *non può essere* antecedente alla data corrente;
RV33 - In “Dettaglio_Scontrini” *si devono* necessariamente inserire i record nei campi “ID_Scontrino”, “ID_Prodotto”, “Quantita” e “PrezzoVen”.

Regole di derivazione:
RD1 - L’Anzianita *si ricava* facendo un conteggio degli anni di differenza tra la data corrente e la data di assunzione;
RD2 - L’importo dello stipendo *si ottiene* sommando StipendioBase(Livello) con StipendioAgg(Anzianita);
RD3 - Il totale dello scontrino *si calcola* sommando i vari importi dei prodotti;
RD4 - Gli importi dei prodotti *si calcolano* moltiplicando quantità per prezzo;
RD5 - Le quantità di prodotti disponibili in magazzino *vengono aggiornate* (aggiunte) non appena l’ordine viene consegnato;
RD6 - Le quantità di prodotti disponibili in magazzino *vengono aggiornate* (sottratte) non appena avviene il pagamento dello scontrino;
RD7 - I punti della tessera di un fidelizzato *vengono aggiunti* ogni volta che paga lo scontrino.
RD8 - I punti della tessera di un fidelizzato *vengono sottratti* ogni volta che ritira un premio.

5.Ristrutturazione ed ottimizzazione dello Schema E-R



La fase di ottimizzazione del Modello E-R ha evidenziato i seguenti problemi:

- le generalizzazioni non sono previste nel modello relazionale. Per ovviare a questo problema abbiamo aggiunto un attributo (Tipo) all'entita' Anagrafica per indicare se una persona appartiene agli Impiegati o ai Fidelizzati, creando rispettivamente le due relazioni Appartenere1 ed Appartenere2 che legano le tre entita' tra di loro

- le generalizzazioni Punti Raccolta e Premio invece sono state eliminate accorpando gli attributi nell'entita' Prodotto in quanto lo spreco di memoria risulta ininfluente rispetto alle Selezioni addizionali necessarie se avessimo creato le relazioni

- per risolvere il problema dello scarico dei prodotti venduti dal magazzino e per il calcolo dei Punti Raccolta da accreditare ai Fidelizzati, abbiamo aggiunto un attributo (Pagato) all'entita' Scontrino; tale attributo ci permettera' in fase realizzativa di attivare una procedura automatica che effettuera' le operazioni indicate sopra

- per risolvere il problema del carico dei prodotti ordinati nel magazzino abbiamo aggiunto un attributo (Consegnato) all'entita' Ordine; tale attributo ci permettera' in fase realizzativa di attivare una procedura automatica che effettuera' le operazioni indicate sopra

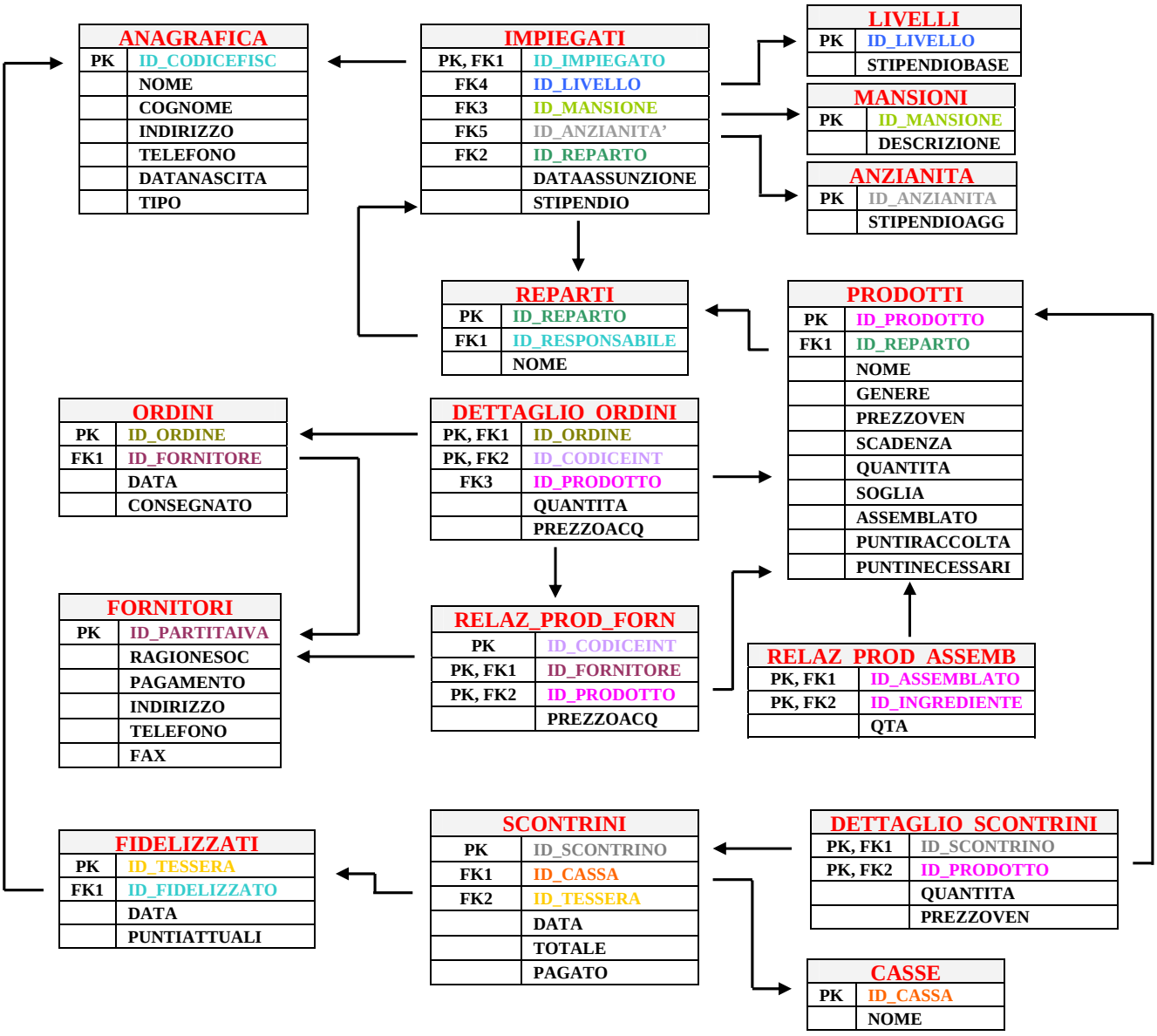
- il problema delle ridondanze e' stato valutato molto attentamente, ma siamo giunti alla conclusione che era necessario mantenerle, infatti un ordine effettuato in passato puo' aver acquistato un prodotto ad un prezzo inferiore a quello odierno e per una eventuale analisi degli aumenti dei prezzi questo dato sarebbe rilevante, in caso omettessimo questo dato, l'analisi non sarebbe piu' praticabile. Stessa cosa sul prezzo di vendita, se volessi infatti sapere a quali prezzi ho venduto un particolare prodotto nel passato e omettessi una copia del prezzo di vendita, l'analisi non sarebbe possibile. In conclusione nella relazione Acquistare e nella relazione Vendere sono stati mantenuti gli attributi relativi al Prezzo di Acquisto e al Prezzo di Vendita

6.Traduzione nel modello relazionale

La traduzione nel modello relazionale adotta la seguente regola stilistica: le chiavi primarie sono identificate con la sottolineatura mentre le chiavi esterne sono in *corsivo*.

- ANAGRAFICA (ID_CodiceFisc, Nome, Cognome, Indirizzo, Telefono, DataNascita, Tipo)
- IMPIEGATI (*ID_Impiegato*, *ID_Reparto*, *ID_Mansione*, *ID_Livello*, *ID_Anzianita*, DataAssunzione, Stipendio)
- REPARTI (ID_Reparto, *ID_Responsabile*, Nome)
- MANSIONI (ID_Mansione, Descrizione)
- LIVELLI (ID_Livello, StipendioBase)
- ANZIANITA (ID_Anzianita, StipendioAgg)
- PRODOTTI (ID_Prodotto, *ID_Reparto*, Nome, Genere, PrezzoVen, Scadenza, Quantita, Soglia, Assemblato, PuntiRaccolta, PuntiNecessari)
- RELAZ_PROD_ASSEMB (*ID_Assemblato*, *ID_Ingrediente*, Quantita)
- FORNITORI (ID_PartitaIva, RagioneSoc, Pagamento, Indirizzo, Telefono, Fax)
- RELAZ_PROD_FORN (ID_CodiceInt, *ID_Fornitore*, *ID_Prodotto*, PrezzoAcq)
- ORDINI (ID_Ordine, *ID_Fornitore*, Data, Consegnato)
- DETTAGLIO_ORDINI (*ID_Ordine*, *ID_CodiceInt*, *ID_Prodotto*, Quantita, PrezzoAcq)
- FIDELIZZATI (ID_Tessera, *ID_Fidelizzato*, Data, PuntiAttuali)
- CASSE (ID_Cassa, Nome)
- SCONTRINI (ID_Scontrino, *ID_Cassa*, *ID_Tessera*, Data, Totale, Pagato)
- DETTAGLIO_SCONTRINI (*ID_Scontrino*, *ID_Prodotto*, Quantita, PrezzoVen)

Schema Logico



Descrizione Tabelle

ANAGRAFICA = Tabella contenente i dati anagrafici relativi alle persone che possono essere Clienti o Impiegati del Supermercato

"ID_CodiceFisc"	**Codice univoco identificante il codice fiscale delle persone**
"Nome"	**Nome della persona**
"Cognome"	**Cognome della persona**
"Indirizzo"	**Indirizzo della persona**
"Telefono"	**Numero di telefono della persona**
"DataNascita"	**Data di nascita della persona**
"Tipo"	**Campo indicante se Impiegato o Cliente**

LIVELLI = Tabella contenente i Livelli raggiungibili dagli impiegati con il corrispondente Stipendio Base

"ID_Livello"	**Codice univoco identificante il livello**
"StipendioBase"	**Stipendio Base relativo al livello**

ANZIANITA = Tabella contenente le Anzianita' raggiungibili dagli impiegati con il corrispondente Stipendio Addizionale

"ID_Anzianita"	**Codice univoco identificante l'anzianita' di servizio**
"StipendioAgg"	**Stipendio Addizionale relativo all'anzianita' di servizio**

MANSIONI = Tabella contenente le Mansioni svolte dagli impiegati

"ID_Mansione"	**Codice univoco identificante la mansione svolta**
"Descrizione"	**Descrizione della mansione**

IMPIEGATI = Tabella che associa gli Impiegati all'Anagrafica, al Reparto, alla Mansione, al Livello e all'Anzianita'

"ID_Impiegato"	**FK -> Indicante l'anagrafica corrispondente**
"ID_Reparto"	**FK -> Indicante il reparto associato**
"ID_Mansione"	**FK -> Indicante la mansione svolta**
"ID_Livello"	**FK -> Indicante il livello economico**
"ID_Anzianita"	**FK -> Indicante l'anzianita' di servizio**
"DataAssunzione"	**Data di assunzione**
"Stipendio"	**Campo derivato dalla somma StipendioBase+StipendioAgg**

REPARTI = Tabella contente i Reparti ed il relativo Responsabile

"ID_Reparto"	**Codice univoco identificante il reparto**
"Nome"	**Nome del reparto**
"ID_Responsabile"	**FK -> Indicante l'impiegato corrispondente**

PRODOTTI = Tabella contene i dati relativi Prodotti presenti nel Supermercato

“ID_Prodotto”	Codice univoco identificante il prodotto generato da trigger
“ID_Reparto”	FK -> Indicante in che reparto si trova il prodotto
“Nome”	Nome del prodotto
“Genere”	Genere del prodotto
“PrezzoVen”	Prezzo di vendita del prodotto
“Scadenza”	Data di scadenza del prodotto
“Quantita”	Quantita’ presente in magazzino
“Soglia”	Quantita’ minima sotto la quale effettuare il riordino
“Assemblato”	S indica che e’ un prodotto Assemblato da piu’ Prodotti N indica che e’ un prodotto semplice
“PuntiRaccolta”	0 indica che il prodotto non partecipa alla Raccolta Punti x indica il numero di punti che il prodotto fa accumulare
“PuntiNecessari”	0 indica che il prodotto non e’ un Premio x indica il numero di punti necessari per averlo come Premio

RELAZ_PROD_ASSEMB = Tabella che associa i prodotti Assemblati con gli ingredienti necessari a comporlo con le relative quantita’

“ID_Assemblato”	FK -> Indicante il codice del prodotto assemblato
“ID_Ingrediente”	FK -> Indicante il prodotto ingrediente che compone il prodotto Assemblato
“Quantita”	Indica la quantita di prodotto ingrediente necessaria

FORNITORI = Tabella contenente i dati relativi ai Fornitori del Supermercato

“ID_PartitaIva”	Codice univoco identificante la partita iva del fornitore
“RagioneSoc”	Ragione Sociale del fornitore
“Indirizzo”	Indirizzo del fornitore
“Telefono”	Numero di telefono del fornitore
“Fax”	Numero di fax del fornitore
“Pagamento”	Metodo di pagamento relativo al fornitore

ORDINI = Tabella che associa gli Ordini ai Fornitori

“ID_Ordine”	Codice univoco generato da trigger indicante il numero d’ordine
“ID_Fornitore”	FK -> Indicante il fornitore a cui si effettua l’ordine
“Data”	Data di emissione dell’ordine
“Consegnato”	Indica se l’ordine e’ stato consegnato dal Fornitore

DETTAGLIO_ORDINI = Tabella che associa agli Ordini i Codici Interni dei prodotti dei Fornitori ai Codici dei prodotti del Supermercato, con le quantita' ordinate e prezzo di acquisto

"ID_Ordine"	**Codice univoco generato da trigger indicante il numero d'ordine**
"ID_CodiceInt"	**FK -> Indicante il codice interno del fornitore relativo al prodotto**
"ID_Prodotto"	**FK -> Indicante il codice del supermercato relativo al prodotto**
"Quantita"	**Numero di pezzi**
"PrezzoAcq"	**Prezzo di acquisto di ogni singolo pezzo**

CASSE = Tabella contenente le Casse presenti nel supermercato

"ID_Cassa"	**Codice univoco identificante la cassa del supermercato**
"Nome"	**Nome della cassa**

FIDELIZZATI = Tabella che associa i Fidelizzati all'Anagrafica

"ID_Tessera"	**Codice univoco identificante il numero di tessera del fidelizzato**
"ID_Fidelizzato"	**FK -> Indicante l'anagrafica relativa al fidelizzato**
"Data"	**Data di fidelizzazione**
"PuntiAttuali"	**Punti accumulati presenti nella tessera del fidelizzato**

SCONTRINI = Tabella che contiene gli Scontrini effettuati dalle Casse

"ID_Scontrino"	**Codice univoco identificante il numero dello scontrino emesso da una particolare cassa**
"ID_Cassa"	**FK -> Indicante quale cassa ha emesso lo scontrino**
"ID_Tessera"	**FK -> Indicante l'eventuale tessera del fidelizzato**
"Data"	**Data di emissione dello scontrino**
"Totale"	**Campo derivato dalla somma di tutte le Quantita*PrezzoVen della tabella Dettaglio_Scontrini per lo stesso ID_Scontrino**
"Pagato"	**Indica se lo scontrino e' stato emesso e pagato**

DETTAGLIO_SCONTRINI = Tabella che associa ad ogni Scontrino i prodotti acquistati

"ID_Scontrino"	**FK -> Indicante a quale scontrino si riferiscono i prodotti**
"ID_Prodotto"	**FK -> Indicante quale prodotto e' stato acquistato**
"Quantita"	**Numero pezzi acquistati**
"PrezzoVen"	**Prezzo di vendita di ogni pezzo**

7.Implementazione Progetto

1) Fornire le istruzioni per la creazione del DB e degli oggetti che lo costituiscono

/* Codice SQL per la Creazione ed uso dello Schema Supermercato */

```
DROP DATABASE IF EXISTS Supermercato;
CREATE DATABASE Supermercato;
USE Supermercato;
```

/* Codice SQL per la Creazione della Tabella Anagrafica */

```
DROP TABLE IF EXISTS Supermercato.Anagrafica;
CREATE TABLE Anagrafica (
 ID_CodiceFisc CHAR(16) NOT NULL UNIQUE,
 Nome VARCHAR(15) NOT NULL,
 Cognome VARCHAR(20) NOT NULL,
 Indirizzo VARCHAR(60) NOT NULL,
 Telefono VARCHAR(15),
 DataNascita DATE NOT NULL,
 Tipo ENUM('I','F') NOT NULL,
 PRIMARY KEY (ID_CodiceFisc)
);
```

/* Codice SQL per la Creazione della Tabella Livello */

```
DROP TABLE IF EXISTS Supermercato.Livelli;
CREATE TABLE Livelli (
 ID_Livello INTEGER(2) UNSIGNED NOT NULL UNIQUE,
 StipendioBase DECIMAL(8,2) UNSIGNED NOT NULL,
 PRIMARY KEY (ID_Livello)
);
```

/* Codice SQL per la Creazione della Tabella Anzianita */

```
DROP TABLE IF EXISTS Supermercato.Anzianita;
CREATE TABLE Anzianita (
 ID_Anzianita INTEGER(2) UNSIGNED NOT NULL UNIQUE,
 StipendioAgg DECIMAL(6,2) UNSIGNED NOT NULL,
 PRIMARY KEY (ID_Anzianita)
);
```

/* Codice SQL per la Creazione della Tabella Mansioni */

```
DROP TABLE IF EXISTS Supermercato.Mansioni;
CREATE TABLE Mansioni (
 ID_Mansione INTEGER(2) UNSIGNED NOT NULL UNIQUE,
 Descrizione VARCHAR(15) NOT NULL,
 PRIMARY KEY (ID_Mansione)
);
```

```
/* Codice SQL per la Creazione della Tabella Impiegati */

DROP TABLE IF EXISTS Supermercato.Impiegati;
CREATE TABLE Impiegati (
 ID_Impiegato CHAR(16) NOT NULL UNIQUE,
 ID_Reparto INTEGER(3) UNSIGNED DEFAULT 0,
 ID_Mansione INTEGER(2) UNSIGNED NOT NULL DEFAULT 0,
 ID_Livello INTEGER(2) UNSIGNED NOT NULL DEFAULT 1,
 ID_Anzianita INTEGER(2) UNSIGNED NOT NULL DEFAULT 0,
 DataAssunzione DATE NOT NULL,
 Stipendio DECIMAL(8,2) UNSIGNED NOT NULL,
 PRIMARY KEY (ID_Impiegato),
 FOREIGN KEY (ID_Impiegato) REFERENCES Anagrafica(ID_CodiceFisc)
 ON DELETE CASCADE
 ON UPDATE CASCADE,
 FOREIGN KEY (ID_Mansione) REFERENCES Mansioni(ID_Mansione)
 ON DELETE NO ACTION
 ON UPDATE CASCADE,
 FOREIGN KEY (ID_Livello) REFERENCES Livelli(ID_Livello)
 ON DELETE NO ACTION
 ON UPDATE CASCADE,
 FOREIGN KEY (ID_Anzianita) REFERENCES Anzianita(ID_Anzianita)
 ON DELETE NO ACTION
 ON UPDATE CASCADE
);

/*
Trigger che prima dell'inserimento di un impiegato:
1) Verifica che i dati che si inseriscono siano effettivamente di un impiegato (altr. errore)
2) Verifica che la Data di Assunzione sia <= CURDATE() (altr. errore)
3) Calcola l'anzianita' in anni facendo YEAR(CURDATE-DataAssunzione)
*/
DROP TRIGGER IF EXISTS Supermercato.check_ins_calcola_stipendio;
DELIMITER //
CREATE TRIGGER check_ins_calcola_stipendio BEFORE INSERT ON Impiegati
FOR EACH ROW
begin
  if (SELECT Tipo FROM Anagrafica A WHERE A.ID_CodiceFisc=NEW.ID_Impiegato)='I'
then
    if (NEW.DataAssunzione>CURDATE()) then
      SET NEW.DataAssunzione:=NULL;
    else
      SET NEW.ID_Anzianita:=
YEAR(FROM_DAYS(DATEDIFF(CURDATE(),NEW.DataAssunzione)));
      SET NEW.Stipendio:=
(SELECT StipendioBase FROM Livelli L WHERE L.ID_Livello=NEW.ID_Livello) +
(SELECT StipendioAgg FROM Anzianita A WHERE A.ID_Anzianita=NEW.ID_Anzianita);
  end if;
else
  SET NEW.ID_Impiegato:=NULL;
end if;
```

```
end;
//
DELIMITER ;

/*
Trigger che durante la modifica di un impiegato:
Solo se e' cambiato ID_Anzianita o ID_Livello o Stipendio ricalcola, l'anzianita e lo stipendio.
*/
DROP TRIGGER IF EXISTS Supermercato.upd_calcola_stipendio;
DELIMITER //
CREATE TRIGGER upd_calcola_stipendio BEFORE UPDATE ON Impiegati
FOR EACH ROW
begin
  if NEW.ID_Anzianita IS NOT NULL OR NEW.ID_Livello IS NOT NULL OR NEW.Stipendio is
NOT NULL then
  SET NEW.ID_Anzianita:=
YEAR(FROM_DAYS(ABS(DATEDIFF(CURDATE(),NEW.DataAssunzione))));
  SET NEW.Stipendio:=
(SELECT StipendioBase FROM Livelli L WHERE L.ID_Livello=NEW.ID_Livello) +
(SELECT StipendioAgg FROM Anzianita A WHERE A.ID_Anzianita=NEW.ID_Anzianita);
  end if;
end;
//
DELIMITER ;

/* Codice SQL per la Creazione della Tabella Reparti */

DROP TABLE IF EXISTS Supermercato.Reparti;
CREATE TABLE Reparti (
 ID_Reparto INTEGER(3) UNSIGNED NOT NULL UNIQUE,
 Nome VARCHAR(20) NOT NULL,
 ID_Responsabile CHAR(16) UNIQUE,
 PRIMARY KEY (ID_Reparto),
 FOREIGN KEY (ID_Responsabile) REFERENCES Impiegati(ID_Impiegato)
 ON DELETE SET NULL
 ON UPDATE CASCADE
);

/* Codice SQL per aggiungere la FOREIGN KEY alla Tabella Impiegati */

ALTER TABLE Impiegati ADD FOREIGN KEY (ID_Reparto) REFERENCES
Reparti(ID_Reparto) ON DELETE NO ACTION ON UPDATE CASCADE;

/* Codice SQL per la Creazione della Tabella Prodotti */

DROP TABLE IF EXISTS Supermercato.Prodotti;
CREATE TABLE Prodotti (
 ID_Prodotto INTEGER(6) UNSIGNED NOT NULL UNIQUE,
 ID_Reparto INTEGER(3) UNSIGNED NOT NULL DEFAULT 0,
 Nome VARCHAR(20) NOT NULL UNIQUE,
 Genere VARCHAR(15) NOT NULL,
 PrezzoVen DECIMAL(6,2) UNSIGNED NOT NULL,
 Scadenza DATE DEFAULT NULL,
```

```
 Quantita INTEGER(3) UNSIGNED NOT NULL DEFAULT 0,
 Soglia INTEGER(3) UNSIGNED NOT NULL DEFAULT 0,
 Assemblato ENUM('1','0') NOT NULL DEFAULT '0',
 PuntiRaccolta INTEGER(4) UNSIGNED NOT NULL DEFAULT 0,
 PuntiNecessari INTEGER(6) UNSIGNED NOT NULL DEFAULT 0,
 PRIMARY KEY (ID_Prodotto),
 FOREIGN KEY (ID_Reparto) REFERENCES Reparti(ID_Reparto)
 ON DELETE NO ACTION
 ON UPDATE CASCADE
);

/*
Trigger sull'inserimento di nuovi prodotti che:
1) Verifica che la Data di Scadenza del prodotto sia >= CURDATE() (altr. errore ed esce)
2) Verifica che il prodotto non dia Punti Raccolta e contemporaneamente sia un Premio
*/
DROP TRIGGER IF EXISTS Supermercato.check_ins_prodotti;
DELIMITER //
CREATE TRIGGER check_ins_prodotti BEFORE INSERT ON Prodotti
FOR EACH ROW
begin
  if NEW.Scadenza<CURDATE() then
    SET NEW.ID_Prodotto:=NULL;
  else
     if NEW.PuntiNecessari=NEW.PuntiRaccolta AND NEW.PuntiNecessari<>0 then
       SET NEW.PuntiNecessari:=NULL;
     end if;
  end if;
end;
//
DELIMITER ;

/* Codice SQL per la Creazione della Tabella Relaz_Prod_Assemb */

DROP TABLE IF EXISTS Supermercato.Relaz_Prod_Assemb;
CREATE TABLE Relaz_Prod_Assemb (
 ID_Assemblato INTEGER(6) UNSIGNED NOT NULL,
 ID_Ingrediente INTEGER(6) UNSIGNED NOT NULL,
 Quantita INTEGER(3) UNSIGNED NOT NULL DEFAULT 0,
 PRIMARY KEY (ID_Assemblato, ID_Ingrediente),
 FOREIGN KEY (ID_Assemblato) REFERENCES Prodotti(ID_Prodotto)
 ON DELETE NO ACTION
 ON UPDATE CASCADE,
 FOREIGN KEY (ID_Ingrediente) REFERENCES Prodotti(ID_Prodotto)
 ON DELETE NO ACTION
 ON UPDATE CASCADE
);
```

```
/*
Trigger che verifica durante l'inserimento degli ingredienti di un assemblato:
1) l'ingrediente inserito non sia un assemblato egli stesso (altrimenti rest. errore)
2) il codice dell'assemblato inserito sia effettivamente un assemblato (altr. rest. errore)
*/
DROP TRIGGER IF EXISTS  Supermercato.check_ins_assemb;
DELIMITER //
CREATE TRIGGER check_ins_assemb BEFORE INSERT ON Relaz_Prod_Assemb
FOR EACH ROW
begin
  if (SELECT Assemblato FROM Prodotti P WHERE P.ID_Prodotto=NEW.ID_Ingrediente)='1'
then
    SET NEW.ID_Ingrediente:=NULL;
  end if;
  if (SELECT Assemblato FROM Prodotti P WHERE P.ID_Prodotto=NEW.ID_Assemblato)='0'
then
    SET NEW.ID_Assemblato:=NULL;
  end if;
end;
//
DELIMITER ;

/*
Trigger che verifica durante l'aggiornamento degli ingredienti di un assemblato:
1) l'ingrediente modificato non sia un assemblato egli stesso (altrimenti rest. errore)
2) il codice dell'assemblato inserito sia effettivamente un assemblato (altr. rest. errore)
*/
DROP TRIGGER IF EXISTS Supermercato.check_upd_calcola_stipendio;
DELIMITER //
CREATE TRIGGER check_upd_assemb BEFORE UPDATE ON Relaz_Prod_Assemb
FOR EACH ROW
begin
  if NEW.ID_Assemblato IS NOT NULL then
    if (SELECT Assemblato FROM Prodotti P WHERE P.ID_Prodotto=NEW.ID_Assemblato)='0'
then
      SET NEW.ID_Assemblato:=NULL;
    end if;
  end if;
  if NEW.ID_Ingrediente IS NOT NULL then
    if (SELECT Assemblato FROM Prodotti P WHERE P.ID_Prodotto=NEW.ID_Ingrediente)='1'
then
      SET NEW.ID_Ingrediente:=NULL;
    end if;
  end if;
end;
//
DELIMITER ;
```

/* Codice SQL per la Creazione della Tabella Fornitori */

```
DROP TABLE IF EXISTS Supermercato.Fornitori;
CREATE TABLE Fornitori (
 ID_PartitaIva CHAR(11) NOT NULL UNIQUE,
 RagioneSoc VARCHAR(30) NOT NULL,
 Pagamento VARCHAR(20) NOT NULL,
 Indirizzo VARCHAR(30) NOT NULL,
 Telefono VARCHAR(15),
 Fax VARCHAR(15),
 PRIMARY KEY (ID_PartitaIva)
);
```

/* Codice SQL per la Creazione della Tabella Relaz_Prod_Forn */

```
DROP TABLE IF EXISTS Supermercato.Relaz_Prod_Forn;
CREATE TABLE Relaz_Prod_Forn (
 ID_CodiceInt INTEGER(6) UNSIGNED NOT NULL,
 ID_Fornitore CHAR(11) NOT NULL,
 ID_Prodotto INTEGER(6) UNSIGNED NOT NULL,
 PrezzoAcq DECIMAL(6,2) UNSIGNED NOT NULL DEFAULT 0,
 PRIMARY KEY (ID_CodiceInt, ID_Fornitore, ID_Prodotto),
 FOREIGN KEY (ID_Fornitore) REFERENCES Fornitori(ID_PartitaIva)
 ON DELETE NO ACTION
 ON UPDATE CASCADE,
 FOREIGN KEY (ID_Prodotto) REFERENCES Prodotti(ID_Prodotto)
 ON DELETE NO ACTION
 ON UPDATE CASCADE
);
```

/* Codice SQL per la Creazione della Tabella Ordini */

```
DROP TABLE IF EXISTS Supermercato.Ordini;
CREATE TABLE Ordini (
 ID_Ordine INTEGER(7) UNSIGNED NOT NULL,
 ID_Fornitore CHAR(11) NOT NULL,
 Data DATE NOT NULL,
 Consegnato ENUM('1','0') NOT NULL DEFAULT '0',
 PRIMARY KEY (ID_Ordine),
 FOREIGN KEY (ID_Fornitore) REFERENCES Fornitori(ID_PartitaIva)
 ON DELETE NO ACTION
 ON UPDATE CASCADE
);
```

```
/*
Trigger per verificare i dati prima dell'inserimento negli Ordini:
1) Verifica che la Data di inserimento dell'ordine non sia < CURDATE() (altr. errore)
*/
DROP TRIGGER IF EXISTS Supermercato.ins_ordini;
DELIMITER //
CREATE TRIGGER ins_ordini BEFORE INSERT ON Ordini
FOR EACH ROW
begin
  if NEW.Data<CURDATE() then
    SET NEW.Data:=NULL;
  end if;
end;
//
DELIMITER ;

/*
Trigger per verificare i dati prima dell'inserimento negli Ordini:
1) Verifica che la Data di inserimento dell'ordine non < CURDATE() (altr. errore)
2) Verifica se Consegnato='1' e prima era '0' effettua il caricamento dei prodotti nel magazzino
*/
DROP TRIGGER IF EXISTS Supermercato.upd_ordini;
DELIMITER //
CREATE TRIGGER upd_ordini BEFORE UPDATE ON Ordini
FOR EACH ROW
begin
  if NEW.Data<CURDATE() then
    SET NEW.Data:=NULL;
  else
    if (NEW.Consegnato='1') AND (OLD.Consegnato='0') then
      if (NEW.ID_Ordine IS NOT NULL) then
        CALL carica_ordine(NEW.ID_Ordine);
      else
        CALL carica_ordine(OLD.ID_Ordine);
      end if;
    else
      if (NEW.Consegnato='0') AND (OLD.Consegnato='1') then
        SET NEW.Consegnato=NULL;
      end if;
    end if;
  end if;
end;
//
DELIMITER ;
```

```
/*
Procedura che effettua lo scaricamento dei prodottti dal magazzino per lo scontrino passato nel
parametro myscontrino e restituisce il totale dello scontrino nel parametro mytotale
*/
DROP PROCEDURE IF EXISTS Supermercato.carica_ordine;
DELIMITER //
CREATE PROCEDURE carica_ordine (IN myordine INT)
begin
  DECLARE myprodotto INT;
  DECLARE myquantita INT;
  DECLARE done INT DEFAULT 0;
  DECLARE cursore CURSOR FOR
  SELECT ID_Prodotto, Quantita FROM Dettaglio_Ordini WHERE ID_Ordine=myordine;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  OPEN cursore;
  REPEAT
    FETCH cursore INTO myprodotto, myquantita;
    if NOT done then
      UPDATE Prodotti P SET P.Quantita:=P.Quantita+myquantita WHERE
P.ID_Prodotto=myprodotto;
    end if;
  UNTIL done END REPEAT;
  CLOSE cursore;
end
//
DELIMITER ;

/* Codice SQL per la Creazione della Tabella Dettaglio_Ordini */

DROP TABLE IF EXISTS Supermercato.Dettaglio_Ordini;
CREATE TABLE Dettaglio_Ordini (
 ID_Ordine INTEGER(7) UNSIGNED NOT NULL,
 ID_CodiceInt INTEGER(6) UNSIGNED NOT NULL,
 ID_Prodotto INTEGER(6) UNSIGNED NOT NULL,
 Quantita INTEGER(3) UNSIGNED NOT NULL DEFAULT 0,
 PrezzoAcq DECIMAL(6,2) UNSIGNED NOT NULL DEFAULT 0,
 PRIMARY KEY (ID_Ordine, ID_CodiceInt),
 FOREIGN KEY (ID_Ordine) REFERENCES Ordini(ID_Ordine)
 ON DELETE CASCADE
 ON UPDATE CASCADE,
 FOREIGN KEY (ID_CodiceInt) REFERENCES Relaz_Prod_Forn(ID_CodiceInt)
 ON DELETE NO ACTION
 ON UPDATE CASCADE,
 FOREIGN KEY (ID_Prodotto) REFERENCES Prodotti(ID_Prodotto)
 ON DELETE NO ACTION
 ON UPDATE CASCADE
);
```

```
/*
Trigger per verificare i dati prima dell'inserimento dei dettagli ordini:
1) Verifica che ID_Prodotto sia in relazione a CodiceInt del Fornitore nella Tabella
Relaz_Prod_Forn
2) Verifica che il prodotto non sia assemblato (altr. errore)
3) Copia Prezzo Acquisto dalla Tabella Relaz_Prod_Forn
Relaz_Prod_Forn
*/
DROP TRIGGER IF EXISTS Supermercato.ins_dettaglio_ordini;
DELIMITER //
CREATE TRIGGER ins_dettaglio_ordini BEFORE INSERT ON Dettaglio_Ordini
FOR EACH ROW
begin
  DECLARE id INTEGER(7) UNSIGNED;
  SET id:= (SELECT ID_Prodotto FROM Relaz_Prod_Forn RPF WHERE
RPF.ID_Fornitore=(SELECT ID_Fornitore FROM Ordini O WHERE
O.ID_Ordine=NEW.ID_Ordine) AND RPF.ID_CodiceInt=NEW.ID_CodiceInt);
  if id IS NULL OR id<>NEW.ID_Prodotto then
    SET NEW.ID_Prodotto:=NULL;
  else
     if (SELECT Assemblato FROM Prodotti P WHERE P.ID_Prodotto=NEW.ID_Prodotto)='1'
then
      SET NEW.ID_Prodotto:=NULL;
    else
      SET NEW.PrezzoAcq:=(SELECT PrezzoAcq FROM Relaz_Prod_Forn RPF WHERE
RPF.ID_Fornitore=(SELECT ID_Fornitore FROM Ordini O WHERE
O.ID_Ordine=NEW.ID_Ordine) AND RPF.ID_CodiceInt=NEW.ID_CodiceInt);
    end if;
  end if;
end;
//
DELIMITER ;

/* Codice SQL per la Creazione della Tabella Fidelizzati */

DROP TABLE IF EXISTS Supermercato.Fidelizzati;
CREATE TABLE Fidelizzati (
 ID_Tessera INTEGER(6) UNSIGNED NOT NULL AUTO_INCREMENT,
 ID_Fidelizzato CHAR(16) NOT NULL UNIQUE,
 Data DATE NOT NULL,
 PuntiAttuali INTEGER(7) UNSIGNED NOT NULL DEFAULT 0,
 PRIMARY KEY (ID_Tessera),
 FOREIGN KEY (ID_Fidelizzato) REFERENCES Anagrafica(ID_CodiceFisc)
 ON DELETE NO ACTION
 ON UPDATE CASCADE
);
```

```
/*
Trigger che durante l'inserimento di un fidelizzato:
1) Verifica che i dati che si inseriscono siano effettivamente di un fidelizzato (altr. errore)
2) Verifica che la Data di Fidelizzazione sia <= CURDATE() (altr. errore)
*/
DROP TRIGGER IF EXISTS Supermercato.check_ins_fidelizzati;
DELIMITER //
CREATE TRIGGER check_ins_fidelizzati BEFORE INSERT ON Fidelizzati
FOR EACH ROW
begin
  if NOT (SELECT Tipo FROM Anagrafica A WHERE
A.ID_CodiceFisc=NEW.ID_Fidelizzato)='F' then
     SET NEW.ID_Fidelizzato:=NULL;
  end if;
  if (NEW.Data>CURDATE()) then
     SET NEW.Data:=NULL;
  end if;
end;
//
DELIMITER ;

/* Codice SQL per la Creazione della Tabella Casse */

DROP TABLE IF EXISTS Supermercato.Casse;
CREATE TABLE Casse (
 ID_Cassa INTEGER(2) UNSIGNED NOT NULL,
 Nome VARCHAR(20) NOT NULL,
 PRIMARY KEY (ID_Cassa)
 );

/* Codice SQL per la Creazione della Tabella Scontrini */

DROP TABLE IF EXISTS Supermercato.Scontrini;
CREATE TABLE Scontrini (
 ID_Scontrino INTEGER(8) UNSIGNED NOT NULL AUTO_INCREMENT,
 ID_Cassa INTEGER(2) UNSIGNED NOT NULL,
 ID_Tessera INTEGER(6) UNSIGNED DEFAULT 0,
 Data DATE NOT NULL,
 Totale DECIMAL(7,2) UNSIGNED NOT NULL,
 Pagato ENUM('1','0') NOT NULL DEFAULT '0',
 PRIMARY KEY (ID_Scontrino),
 FOREIGN KEY (ID_Cassa) REFERENCES Casse(ID_Cassa)
 ON DELETE NO ACTION
 ON UPDATE CASCADE,
 FOREIGN KEY (ID_Tessera) REFERENCES Fidelizzati(ID_Tessera)
 ON DELETE NO ACTION
 ON UPDATE CASCADE
);
```

```
/*
Trigger eseguito prima dell'inserimento degli scontrini che genera un nuovo ID_Scontrino: se
non ci sono scontrini nella Tabella ID_Scontrino = 1 altrimenti ID_Scontrino =
MAX(ID_Scontrino)+1.
Inoltre controlla che la data dello scontrino sia uguale a quella attuale.
*/
DROP TRIGGER IF EXISTS Supermercato.inc_id_scontrino;
DELIMITER //
CREATE TRIGGER inc_id_scontrino BEFORE INSERT ON Scontrini
FOR EACH ROW
begin
  SET NEW.ID_Scontrino:= (SELECT MAX(ID_Scontrino) FROM Scontrini);
  if NEW.ID_Scontrino IS NULL then
    SET NEW.ID_Scontrino:=1;
  else
    SET NEW.ID_Scontrino:=NEW.ID_Scontrino+1;
  end if;
  if NEW.Data<>CURDATE() then
    SET NEW.Data:=NULL;
  end if;
end;
//
DELIMITER ;

/*
Trigger eseguito prima dell'update dello scontrino che se Pagato='1' chiama la procedura che
scarica i prodotti venduti e calcola il totale dello scontrino e calcola il totale dei punti raccolta
aggiornando le rispettive Tabelle
*/
DROP TRIGGER IF EXISTS Supermercato.up_id_scontrino;
DELIMITER //
CREATE TRIGGER up_id_scontrino BEFORE UPDATE ON Scontrini
FOR EACH ROW
begin
  DECLARE xpunti INT;
  DECLARE xtessera INT;
  SET xpunti:=0;
  SET xtessera:=0;
  if (NEW.ID_Tessera IS NOT NULL) then
    SET xtessera:=NEW.ID_Tessera;
  else
    if (OLD.ID_Tessera IS NOT NULL) then
      SET xtessera:=OLD.ID_Tessera;
    end if;
  end if;
  if (NEW.Pagato='1') AND (OLD.Pagato='0') then
    if (NEW.ID_Scontrino IS NOT NULL) then
      CALL scarica_scontrino(NEW.ID_Scontrino,NEW.Totale, xpunti);
    else
      CALL scarica_scontrino(OLD.ID_Scontrino,NEW.Totale, xpunti);
    end if;
```

```
   if (xtessera<>0) AND (xpunti<>0) then
     UPDATE Fidelizzati F SET F.PuntiAttuali:=F.PuntiAttuali+xpunti WHERE
F.ID_Tessera=xtessera;
   end if;
  else
   if (NEW.Pagato='0') AND (OLD.Pagato='1') then
     SET NEW.Pagato:=NULL;
   end if;
  end if;
end;
//
DELIMITER ;

/* Codice SQL per la Creazione della Tabella Dettaglio_Scontrini */

DROP TABLE IF EXISTS Supermercato.Dettaglio_Scontrini;
CREATE TABLE Dettaglio_Scontrini (
 ID_Scontrino INTEGER(8) UNSIGNED NOT NULL,
 ID_Prodotto INTEGER(6) UNSIGNED NOT NULL,
 Quantita INTEGER(2) UNSIGNED NOT NULL,
 PrezzoVen DECIMAL(6,2) UNSIGNED NOT NULL,
 PRIMARY KEY (ID_Scontrino, ID_Prodotto),
 FOREIGN KEY (ID_Scontrino) REFERENCES Scontrini(ID_Scontrino)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION,
 FOREIGN KEY (ID_Prodotto) REFERENCES Prodotti(ID_Prodotto)
 ON DELETE NO ACTION
 ON UPDATE CASCADE
);

/*
Trigger che durante l'inserimento di un Dettaglio Scontrino:
1) Setta il prezzo di vendita del prodotto come scritto nella Tabella Prodotti
*/
DROP TRIGGER IF EXISTS supermercato.ins_dettaglio_scontrini;
DELIMITER //
CREATE TRIGGER ins_dettaglio_scontrini BEFORE INSERT ON Dettaglio_Scontrini
FOR EACH ROW
begin
  SET NEW.PrezzoVen:=(SELECT PrezzoVen FROM Prodotti P WHERE
P.ID_Prodotto=NEW.ID_Prodotto);
end;
//
DELIMITER ;
```

```
/*
Procedura che effettua lo scaricamento dei prodottti dal magazzino per lo scontrino passato nel
parametro myscontrino e restituisce il totale dello scontrino nel parametro mytotale e restituisce
i puntiraccolta totali nel parametro mypunti
*/
DROP PROCEDURE IF EXISTS Supermercato.scarica_scontrino;
DELIMITER //
CREATE PROCEDURE scarica_scontrino (IN myscontrino INT, OUT mytotale DECIMAL(6,2),
OUT mypunti INT)
begin
  DECLARE myprodotto INT;
  DECLARE myquantita INT;
  DECLARE myprezzo DECIMAL(6,2) UNSIGNED DEFAULT 0;
  DECLARE myraccolta INT;
  DECLARE done INT DEFAULT 0;
  DECLARE cursore CURSOR FOR
  SELECT DS.ID_Prodotto, DS.Quantita, DS.PrezzoVen, P.PuntiRaccolta FROM
Dettaglio_Scontrini DS, Prodotti P WHERE DS.ID_Scontrino=myscontrino AND
DS.ID_Prodotto=P.ID_Prodotto;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  SET mytotale:=0;
  SET mypunti:=0;
  OPEN cursore;
  REPEAT
    FETCH cursore INTO myprodotto, myquantita, myprezzo, myraccolta;
    if NOT done then
      UPDATE Prodotti SET Quantita:=Quantita-myquantita WHERE ID_Prodotto=myprodotto;
      SET mytotale:=mytotale+myquantita*myprezzo;
      SET mypunti:=mypunti+myquantita*myraccolta;
    end if;
  UNTIL done END REPEAT;
  CLOSE cursore;
end
//
DELIMITER ;
```

Inserimento dei valori nella Base Dati

/* Insert Livelli */

```
INSERT INTO Livelli VALUES (1,800);
INSERT INTO Livelli VALUES (2,1000);
INSERT INTO Livelli VALUES (3,1200);
INSERT INTO Livelli VALUES (4,1400);
INSERT INTO Livelli VALUES (5,1800);
INSERT INTO Livelli VALUES (6,3000);
INSERT INTO Livelli VALUES (7,5000);
INSERT INTO Livelli VALUES (8,10000);
INSERT INTO Livelli VALUES (9,20000);
INSERT INTO Livelli VALUES (10,50000);
```

/* Insert Anzianita */

```
INSERT INTO Anzianita VALUES (0,0);
INSERT INTO Anzianita VALUES (1,10);
INSERT INTO Anzianita VALUES (2,20);
INSERT INTO Anzianita VALUES (3,30);
INSERT INTO Anzianita VALUES (4,40);
INSERT INTO Anzianita VALUES (5,50);
INSERT INTO Anzianita VALUES (6,70);
INSERT INTO Anzianita VALUES (7,90);
INSERT INTO Anzianita VALUES (8,110);
INSERT INTO Anzianita VALUES (9,130);
INSERT INTO Anzianita VALUES (10,150);
INSERT INTO Anzianita VALUES (11,180);
INSERT INTO Anzianita VALUES (12,210);
INSERT INTO Anzianita VALUES (13,240);
INSERT INTO Anzianita VALUES (14,270);
INSERT INTO Anzianita VALUES (15,300);
INSERT INTO Anzianita VALUES (16,310);
INSERT INTO Anzianita VALUES (17,320);
INSERT INTO Anzianita VALUES (18,330);
INSERT INTO Anzianita VALUES (19,340);
INSERT INTO Anzianita VALUES (20,350);
INSERT INTO Anzianita VALUES (21,360);
INSERT INTO Anzianita VALUES (22,370);
INSERT INTO Anzianita VALUES (23,380);
INSERT INTO Anzianita VALUES (24,390);
INSERT INTO Anzianita VALUES (25,400);
INSERT INTO Anzianita VALUES (26,410);
INSERT INTO Anzianita VALUES (27,420);
INSERT INTO Anzianita VALUES (28,430);
INSERT INTO Anzianita VALUES (29,440);
INSERT INTO Anzianita VALUES (30,450);
INSERT INTO Anzianita VALUES (31,460);
INSERT INTO Anzianita VALUES (32,470);
INSERT INTO Anzianita VALUES (33,480);
INSERT INTO Anzianita VALUES (34,490);
```

```
INSERT INTO Anzianita VALUES (35,500);
INSERT INTO Anzianita VALUES (36,510);
INSERT INTO Anzianita VALUES (37,520);
INSERT INTO Anzianita VALUES (38,530);
INSERT INTO Anzianita VALUES (39,540);
INSERT INTO Anzianita VALUES (40,550);

/* Insert Mansioni */

INSERT INTO Mansioni VALUES (0,'NO MANSIONE');
INSERT INTO Mansioni VALUES (1,'MAGAZZINIERE');
INSERT INTO Mansioni VALUES (2,'COMMESSO');
INSERT INTO Mansioni VALUES (3,'CAPO REPARTO');
INSERT INTO Mansioni VALUES (4,'ADDETTO CASSA');
INSERT INTO Mansioni VALUES (5,'ADDETTO PULIZIE');
INSERT INTO Mansioni VALUES (6,'VICE DIRETTORE');
INSERT INTO Mansioni VALUES (7,'DIRETTORE');

/* Insert Reparti */

INSERT INTO Reparti VALUES (0,'NESSUN REPARTO',NULL);
INSERT INTO Reparti VALUES (1,'MAGAZZINO',NULL);
INSERT INTO Reparti VALUES (2,'PANETTERIA',NULL);
INSERT INTO Reparti VALUES (3,'MACELLERIA',NULL);
INSERT INTO Reparti VALUES (4,'PESCHERIA',NULL);
INSERT INTO Reparti VALUES (5,'CASSA',NULL);
INSERT INTO Reparti VALUES (6,'FRUTTERIA',NULL);
INSERT INTO Reparti VALUES (7,'VERDURA',NULL);
INSERT INTO Reparti VALUES (8,'SURGELATI',NULL);

/* Insert Casse */

INSERT INTO Casse VALUES (1,'Cassa 1');
INSERT INTO Casse VALUES (2,'Cassa 2');
INSERT INTO Casse VALUES (3,'Cassa 3');
INSERT INTO Casse VALUES (4,'Cassa 4');
INSERT INTO Casse VALUES (5,'Cassa 5');
INSERT INTO Casse VALUES (6,'Cassa 6');
INSERT INTO Casse VALUES (7,'Cassa 7');
INSERT INTO Casse VALUES (8,'Cassa Centrale');

/* Insert Anagrafica Impiegati */

INSERT INTO Anagrafica VALUES ('SPRCLD71H16D773Z','Claudio','Soprano','Via
Gregoriana 53','06/94032349',19710616,'I');
INSERT INTO Anagrafica VALUES ('RZZMNC72C11D773Z','Monica','Rizzi','Via
Gregoriana 53','06/22511108',19720311,'I');
INSERT INTO Anagrafica VALUES ('PCAMRC78H11D773Z','Marco','Peca','Via Le Mani
Dal Naso 11','0125/11032349',19780611,'I');
INSERT INTO Anagrafica VALUES ('MZZITL22A22B111E','Italo','Mazzitelli','Via Marco
Tullio 12','06/9421188',19220122,'I');
INSERT INTO Anagrafica VALUES ('LMBMCH79D05E111E','Michele','Lombardi','Via
Stranamore 1','0133/8372188',19790405,'I');
```

INSERT INTO Anagrafica VALUES (’LMBSMN85E27F234E’,’Simona’,’Lombardi’,’Via da Qui 89’,’0444/8899900’,19790405,’I’);

/* Insert Anagrafica Fidelizzati */

INSERT INTO Anagrafica VALUES (’BBBCCC77E10F004X’,’Cacchetta’,’Babbione’,’Via Lucifero 13’,’0111/11112188’,19770510,’F’);
INSERT INTO Anagrafica VALUES (’DDDFNC72F15G005F’,’Franco’,’Daddede’,’Piazza Farnese 15’,’0222/22223244’,19720315,’F’);
INSERT INTO Anagrafica VALUES (’FFNNRC75G25H004F’,’Enrica’,’Fafone’,’Piazza Mazzini 35’,’0333/33333244’,19750725,’F’);
INSERT INTO Anagrafica VALUES (’LMBMGZ69B04F234E’,’Maria Grazia’,’Lombardi’,’Via da Qui 89’,’0444/9999900’,19690204,’F’);
INSERT INTO Anagrafica VALUES (’SPRSMN70H14D773X’,’Simona’,’Soprano’,’Via della Noce 41’,’06/944221155’,19700614,’F’);

/* Insert Impiegati */

INSERT INTO Impiegati VALUES (’SPRCLD71H16D773Z’,2,3,2,0,20050205,0);
INSERT INTO Impiegati VALUES (’RZZMNC72C11D773Z’,2,3,2,0,20060304,0);
INSERT INTO Impiegati VALUES (’PCAMRC78H11D773Z’,2,3,3,0,19990122,0);
INSERT INTO Impiegati VALUES (’MZZITL22A22B111E’,3,3,2,0,20000520,0);
INSERT INTO Impiegati VALUES (’LMBMCH79D05E111E’,3,2,2,0,20080401,0);
INSERT INTO Impiegati VALUES (’LMBSMN85E27F234E’,0,6,6,0,19980122,0);

/* Insert Fidelizzati */

INSERT INTO Fidelizzati (ID_Fidelizzato, Data) VALUES (’BBBCCC77E10F004X’,CURDATE());
INSERT INTO Fidelizzati (ID_Fidelizzato, Data) VALUES (’DDDFNC72F15G005F’,CURDATE());
INSERT INTO Fidelizzati (ID_Fidelizzato, Data) VALUES (’FFNNRC75G25H004F’,CURDATE());
INSERT INTO Fidelizzati (ID_Fidelizzato, Data) VALUES (’LMBMGZ69B04F234E’,CURDATE());
INSERT INTO Fidelizzati (ID_Fidelizzato, Data) VALUES (’SPRSMN70H14D773X’,CURDATE());

/* Insert prodotti normali */

INSERT INTO Prodotti VALUES (1, 1, ’Cavolo’, ’Alimentari’, 2.5, 20090224, 200, 500, ’0’, 10, 0);
INSERT INTO Prodotti VALUES (2, 1, ’Insalata’, ’Alimentari’, 2.1, 20090222, 300, 100, ’0’, 0, 100);
INSERT INTO Prodotti VALUES (3, 1, ’Acqua’, ’Alimentari’, 0.4, 20090501, 350, 450, ’0’, 20, 0);
INSERT INTO Prodotti VALUES (4, 1, ’Lievito’, ’Alimentari’, 1.2, 20090502, 350, 50, ’0’, 5, 0);
INSERT INTO Prodotti VALUES (5, 4, ’Salmone’, ’Pesce’, 7.5, 20090228, 350, 50, ’0’, 40, 0);
INSERT INTO Prodotti VALUES (6, 1, ’Mele’, ’Frutta’, 1.5, 20090228, 350, 50, ’0’, 5, 0);
INSERT INTO Prodotti VALUES (7, 1, ’Pere’, ’Frutta’, 1, 20090228, 350, 50, ’0’, 0, 200);
INSERT INTO Prodotti VALUES (8, 2, ’Farina’, ’Alimentari’, 3, 20101230, 350, 50, ’0’, 3, 0);
INSERT INTO Prodotti VALUES (9, 2, ’Sale’, ’Alimentari’, 3, NULL, 350, 50, ’0’, 2, 0);

```
INSERT INTO Prodotti VALUES (10, 4, 'Pesce spada', 'Pesce', 5.99,  20090228, 40, 20, '0', 100, 0);
INSERT INTO Prodotti VALUES (11, 8, 'Gelato esotico', 'Surgelati', 2.5,  NULL, 480, 150, '0', 0, 0);

/* Insert prodotti assemblati */

INSERT INTO Prodotti VALUES (100, 2, 'Pizza', 'Alimentari', 3,  CURDATE(), 350, 50, '1', 0, 0);
INSERT INTO Prodotti VALUES (110, 2, 'Pane', 'Alimentari', 3,  CURDATE(), 350, 50, '1', 0, 0);

/* Insert ingredienti prodotti assemblati */

INSERT INTO Relaz_Prod_Assemb VALUES (100, 3, 1);
INSERT INTO Relaz_Prod_Assemb VALUES (100, 4, 1);
INSERT INTO Relaz_Prod_Assemb VALUES (100, 5, 1);
INSERT INTO Relaz_Prod_Assemb VALUES (100, 8, 1);
INSERT INTO Relaz_Prod_Assemb VALUES (100, 9, 1);
INSERT INTO Relaz_Prod_Assemb VALUES (110, 4, 2);
INSERT INTO Relaz_Prod_Assemb VALUES (110, 3, 1);
INSERT INTO Relaz_Prod_Assemb VALUES (110, 8, 1);

/* Insert scontrini */

INSERT INTO Scontrini VALUES (0, 1, 1, CURDATE(), 0, '0');
INSERT INTO Scontrini VALUES (0, 1, NULL, CURDATE(), 0, '0');
INSERT INTO Scontrini VALUES (0, 5, NULL, CURDATE(), 0, '0');
INSERT INTO Scontrini VALUES (0, 2, NULL, CURDATE(), 0, '0');

/* Insert dettaglio scontrini */

INSERT INTO Dettaglio_Scontrini VALUES (1, 1, 100, 0);
INSERT INTO Dettaglio_Scontrini VALUES (1, 2, 57, 0);
INSERT INTO Dettaglio_Scontrini VALUES (1, 3, 48, 0);
INSERT INTO Dettaglio_Scontrini VALUES (1, 10, 2, 0);
INSERT INTO Dettaglio_Scontrini VALUES (2, 1, 50, 0);
INSERT INTO Dettaglio_Scontrini VALUES (2, 2, 100, 0);
INSERT INTO Dettaglio_Scontrini VALUES (3, 3, 200, 0);
INSERT INTO Dettaglio_Scontrini VALUES (3, 7, 40, 0);
INSERT INTO Dettaglio_Scontrini VALUES (3, 2, 10, 0);
INSERT INTO Dettaglio_Scontrini VALUES (3, 11, 1, 0);
INSERT INTO Dettaglio_Scontrini VALUES (4, 1, 10, 0);
INSERT INTO Dettaglio_Scontrini VALUES (4, 9, 10, 0);

/* Insert Fornitori */

INSERT INTO Fornitori VALUES ('00049872134', 'Oasi', 'Rate 3 mesi', 'Via Pari 17, Pescara 66023', '085/479865', '085/479860');
INSERT INTO Fornitori VALUES ('00057772134', 'Pirati', 'Contanti', 'Via Piazza 157, Pescara 66023', '085/457865', '085/433160');
INSERT INTO Fornitori VALUES ('00046812134', 'Sprint', 'Giroconto', 'Via Milano 97, Pescara 66023', '085/411105', '085/479870');
```

INSERT INTO Fornitori VALUES (’00149872174’, ’Zippo’, ’Rate 12 mesi’, ’Via Bari 64, Pescara 66023’, ’085/479321’, ’085/479009’);

/* Insert Relaz Prod Forn */

INSERT INTO Relaz_Prod_Forn VALUES (1, ’00049872134’, 3, 2.4);
INSERT INTO Relaz_Prod_Forn VALUES (2, ’00049872134’, 7, 2.8);
INSERT INTO Relaz_Prod_Forn VALUES (3, ’00049872134’, 8, 1.4);
INSERT INTO Relaz_Prod_Forn VALUES (1, ’00057772134’, 3, 2.1);
INSERT INTO Relaz_Prod_Forn VALUES (1, ’00046812134’, 3, 1.7);
INSERT INTO Relaz_Prod_Forn VALUES (2, ’00046812134’, 5, 2.4);

/* Insert Ordini */
INSERT INTO Ordini VALUES (1, ’00046812134’, CURDATE(),’0’);
INSERT INTO Ordini VALUES (2, ’00046812134’, 20090223,’0’);
INSERT INTO Ordini VALUES (3, ’00057772134’, CURDATE(),’0’);
INSERT INTO Ordini VALUES (4, ’00049872134’, 20090425,’0’);

/* Insert Dettaglio Ordini */

INSERT INTO Dettaglio_Ordini VALUES (1, 1, 3, 500, 0);
INSERT INTO Dettaglio_Ordini VALUES (1, 2, 5, 350, 0);
INSERT INTO Dettaglio_Ordini VALUES (3, 1, 3, 400, 0);
INSERT INTO Dettaglio_Ordini VALUES (4, 1, 3, 500, 0);
INSERT INTO Dettaglio_Ordini VALUES (4, 2, 7, 700, 0);

2) Per ogni relazione individuata, fornire le istruzioni di inserimento, modifica ed eliminazione delle istanze.

INSERT INTO Livelli VALUES (x,y**);**
UPDATE Livelli SET A1=x,A2=y **WHERE** condition**;**
DELETE FROM Livelli WHERE condition**;**

INSERT INTO Anzianita VALUES (x,y**);**
UPDATE Anzianita SET A1=x,A2=y **WHERE** condition**;**
DELETE FROM Anzianita WHERE condition**;**

INSERT INTO Mansioni VALUES (x,'string'**);**
UPDATE Mansioni SET A1=x,A2=y **WHERE** condition**;**
DELETE FROM Mansioni WHERE condition**;**

INSERT INTO Reparti VALUES (x,'string1','string2'**);**
UPDATE Reparti SET A1=x,… **WHERE** condition**;**
DELETE FROM Reparti WHERE condition**;**

INSERT INTO Casse VALUES (x,'string'**);**
UPDATE Casse SET A1=x,… **WHERE** condition**;**
DELETE FROM Casse WHERE condition**;**

INSERT INTO Anagrafica VALUES ('string',…**);**
UPDATE Anagrafica SET A1=x,… **WHERE** condition**;**
DELETE FROM Anagrafica WHERE condition**;**

INSERT INTO Impiegati VALUES ('string',…**);**
UPDATE Impiegati SET A1=x,… **WHERE** condition**;**
DELETE FROM Impiegati WHERE condition**;**

INSERT INTO Fidelizzati (A1, A2**) VALUES (**'string',date**);**
UPDATE Fidelizzati SET A1=x,… **WHERE** condition**;**
DELETE FROM Fidelizzati WHERE condition**;**

INSERT INTO Prodotti VALUES (x,…**);**
UPDATE Prodotti SET A1=x,… **WHERE** condition**;**
DELETE FROM Prodotti WHERE condition**;**

INSERT INTO Relaz_Prod_Assemb VALUES (x,…**);**
UPDATE Relaz_Prod_Assemb SET A1=x,… **WHERE** condition**;**
DELETE FROM Relaz_Prod_Assemb WHERE condition**;**

INSERT INTO Scontrini VALUES (x, …**);**
UPDATE Scontrini SET A1=x,… **WHERE** condition**;**
DELETE FROM Scontrini WHERE condition**;**

INSERT INTO Dettaglio_Scontrini VALUES (x,…**);**
UPDATE Dettaglio_Scontrini SET A1=x,… **WHERE** condition**;**
DELETE FROM Dettaglio_Scontrini WHERE condition**;**

INSERT INTO Fornitori VALUES ('string', …**);**

UPDATE Fornitori SET A1='string',… **WHERE** condition**;**
DELETE FROM Fornitori WHERE condition**;**

INSERT INTO Relaz_Prod_Forn VALUES (x, …**);**
UPDATE Relaz_Prod_Forn SET A1=x,… **WHERE** condition**;**
DELETE FROM Relaz_Prod_Forn WHERE condition**;**

INSERT INTO Ordini VALUES (x,…**);**
UPDATE Ordini SET A1=x,… **WHERE** condition**;**
DELETE FROM Ordini WHERE condition**;**

INSERT INTO Dettaglio_Ordini VALUES (x,…**);**
UPDATE Dettaglio_Ordini SET A1=x,… **WHERE** condition**;**
DELETE FROM Dettaglio_Ordini WHERE condition**;**

3) Modifica del responsabile e degli impiegati di un reparto:

UPDATE Reparti SET ID_Responsabile = 'SPRCLD71H16D773Z' WHERE ID_Reparto = 4;
UPDATE Impiegati SET ID_Reparto = 4 WHERE ID_Impiegato = 'SPRCLD71H16D773Z';
UPDATE Impiegati SET ID_Reparto = 4 WHERE ID_Impiegato = 'PCEMRC88T13G482J';
UPDATE Impiegati SET ID_Reparto = 4 WHERE ID_Impiegato = 'LMBMCH79D05E111E';

4) Determinazione delle vendite per un reparto in un particolare periodo:

ID_Reparto=1

```
SELECT P.Nome, SUM(D.Quantita) AS Quantita_Vendute
FROM Dettaglio_Scontrini D, Scontrini S, Prodotti P, Reparti R
WHERE  D.ID_Scontrino = S.ID_Scontrino  AND
  D.ID_Prodotto = P.ID_Prodotto  AND
  P.ID_Reparto = R.ID_Reparto  AND
  S.DATA BETWEEN 20080101 AND 20091231 AND
  R.ID_REPARTO = 1
  GROUP BY P.Nome;
```

Nome	Quantita_Vendute
Acqua	248
Cavolo	160
Insalata	167
Pere	40

5) Determinazione dei prodotti più venduti in un determinato reparto:

ID_Reparto=1

```
SELECT P.Nome, sum(D.Quantita) as Pezzi_Venduti
FROM Dettaglio_Scontrini D, Prodotti P, Reparti R
WHERE  D.ID_Prodotto = P.ID_Prodotto  AND
  P.ID_Reparto = R.ID_Reparto  AND
  R.ID_Reparto = 1
GROUP BY P.ID_Prodotto
ORDER BY sum(D.Quantita) DESC;
```

Nome	Pezzi_Venduti
Acqua	248
Insalata	167
Cavolo	160
Pere	40

6) Modifica del prezzo di un prodotto:

ID_Prodotto=3

```
UPDATE Prodotti SET PrezzoVen = 2.5 WHERE ID_Prodotto = 3;
```

7) Modifica dei dati riguardanti le scorte di prodotto disponibili:

ID_Prodotto=3

```
UPDATE Prodotti SET Quantita=500 WHERE ID_Prodotto=3;
```

8) Per i prodotti “composti”, specifica degli “ingredienti” e delle quantità necessarie alla preparazione:

```
SELECT  P.Nome AS Prodotti_Assemblati, B.Nome AS Ingredienti, B.Quantita AS Quantita
FROM Relaz_Prod_Assemb R, Prodotti P, Prodotti B
WHERE R.ID_Assemblato = P.ID_Prodotto AND
R.ID_Ingrediente = B.ID_Prodotto
ORDER BY P.Nome;
```

Prodotti_Assemblati	Ingredienti	Quantita
Pane	Farina	350
Pane	Acqua	500
Pane	Lievito	350
Pizza	Salmone	350
Pizza	Farina	350
Pizza	Acqua	500
Pizza	Sale	350
Pizza	Lievito	350

9) Inclusione o esclusione di un prodotto dalla raccolta punti e indicazione del numero di puntiraccolta forniti dal prodotto:

ID_Prodotto=11

```
UPDATE Prodotti SET PuntiRaccolta = 50 WHERE ID_PRODOTTO = 11;
UPDATE Prodotti SET PuntiRaccolta = 0 WHERE ID_PRODOTTO = 11;
```

10) Determinazione dei prodotti sotto scorta:

```
SELECT Nome, Genere, Quantita, Soglia
FROM Prodotti
WHERE Quantita < Soglia;
```

Nome	Genere	Quantita	Soglia
Cavolo	Alimentari	200	500

11) Lista dei fornitori dai quali un determinato prodotto può essere acquistato, ordinati in base al prezzo richiesto;

```
SELECT ID_PartitaIva, RagioneSoc, PrezzoAcq
FROM Relaz_Prod_Forn R, Fornitori F
WHERE  R.ID_Fornitore = F.ID_PartitaIva AND
  R.ID_Prodotto = 3
ORDER BY  PrezzoAcq;
```

ID_PartitaIva	RagioneSoc	PrezzoAcq
00046812134	Sprint	1.70
00057772134	Pirati	2.10
00049872134	Oasi	2.40

12) Modifica del numero di punti necessari ad ottenere un determinato premio:

```
UPDATE Prodotti SET PuntiNecessari = 1500 WHERE ID_Prodotto = 11;
```

13) Verifica dei premi attualmente disponibili:

```
SELECT Nome, PuntiNecessari, Quantita
FROM Prodotti
WHERE  PuntiNecessari>0 AND Quantita>0;
```

Nome	PuntiNecessari	Quantita
Insalata	100	300
Pere	200	350
Gelato esotico	1500	480

14) Inserimento/Modifica dei prodotti forniti da un fornitore:

INSERT INTO Relaz_Prod_Forn VALUES (4, ’00049872134’, 7, 1.2);
UPDATE Relaz_Prod_Forn SET ID_Prodotto = 9 WHERE ID_Fornitore = ’00049872134’ AND ID_CodiceInt = 4;

15) Modifica del reparto di assegnazione di un impiegato e del suo livello:

UPDATE Impiegati SET ID_Reparto = 2, ID_Livello = 3 WHERE ID_Impiegato = ’SPRCLD71H16D773Z’;

16) Modifica del numero di punti assegnati al cliente:

UPDATE Fidelizzati SET PuntiAttuali = PuntiAttuali + 100 WHERE ID_Fidelizzato = ’BBBCCC77E10F004X’;

17) Determinazione dei premi a cui un cliente ha diritto:

Per poter rispondere a questa query, dobbiamo attivare le procedure di scarico scontrino attivabili solo quando uno scontrino e’ Pagato, quindi effettueremo prima un pagamento di uno scontrino in modo che vengano caricati i punti del fidelizzato

ID_FIDELIZZATO=1

UPDATE Scontrini SET Pagato=’1’ WHERE ID_Scontrino=1;

```
SELECT P.Nome, P.PuntiNecessari
FROM Fidelizzati F, Prodotti P
WHERE  P.PuntiNecessari <= F.PuntiAttuali AND P.PuntiNecessari<>0
ORDER BY P.PuntiNecessari;
```

Nome	PuntiNecessari
Insalata	100
Pere	200
Gelato esotico	1500

18) Ritiro di un premio da parte di un cliente:

ID_PRODOTTO=2
ID_FIDELIZZAT0=1

Per realizzare quest’aggiornamento usiamo una procedura

CALL ritira_premio(2,1);

```
DROP PROCEDURE IF EXISTS Supermercato.ritira_premio;
DELIMITER //
CREATE PROCEDURE ritira_premio (IN mypremio INT, IN mytessera INT)
begin
  DECLARE xpunti INT;
  SET xpunti=0;
```

```
  UPDATE Prodotti SET Quantita=Quantita-1 WHERE ID_Prodotto=mypremio;
  SELECT PuntiNecessari INTO xpunti FROM Prodotti WHERE ID_Prodotto=mypremio;
  UPDATE Fidelizzati SET PuntiAttuali=PuntiAttuali-xpunti WHERE ID_Tessera=mytessera;
end
//
DELIMITER ;
```

19) Determinazione dei prodotti più acquistati da un cliente:

ID_TESSERA=1

```
SELECT P.Nome, sum(D.Quantita) AS Pezzi_Acquistati
FROM Scontrini S, Dettaglio_Scontrini D, Prodotti P
WHERE  D.ID_Scontrino = S.ID_Scontrino AND
  D.ID_Prodotto = P.ID_Prodotto AND
  S.ID_Tessera = 1
GROUP BY P.ID_Prodotto
ORDER BY sum(D.Quantita) DESC;
```

Nome	Pezzi_Acquistati
Cavolo	100
Insalata	57
Acqua	48
Pesce spada	2

20) Determinazione della spesa totale effettuata da un cliente in un determinato periodo:

ID_FIDELIZZATO=1

```
SELECT A.Nome, SUM(S.Totale) AS Tot_spesa
FROM Scontrini S, Anagrafica A, Fidelizzati F
WHERE  S.Data BETWEEN 20080101 AND 20091231 AND
  S.ID_Tessera = 1 AND
  S.ID_Tessera = F.ID_Tessera AND
  F.ID_Fidelizzato = A.ID_CodiceFisc;
```

Nome	Tot_spesa
Cacchetta	400.88