Università Degli Studi di Cassino e Lazio meridionale.

Corso di Basi di Dati
A.A. 2014/2015

Progettazione di un database per la gestione di un Franchising di vendita e assistenza informatica

PARTECIPANTI:

Daniele La Rosa
Ernesto Di Mambro
Marco Pontone
Maria Zonfrilli

Abstract:

Nel presente elaborato è stato implementato e sperimentato un database per un franchising di vendita e assistenza informatica.
Il franchising vuole gestire la vendita e il supporto tecnico dei suoi punti vendita. Ogni punto vendita ha i suoi dipendenti con determinate mansioni (Gestore, Venditore, Tecnico).
Il **gestore** si occupa delle fatturazioni e della gestione degli ordini dei prodotti nell'inventario.
Il **venditore** si occupa della vendita di uno specifico settore assegnato. Il **tecnico** effettua riparazioni, registra i clienti nei moduli di riparazione e utilizza l'inventario per la ricerca dei componenti di ricambio.

1- Introduzione al problema

L’introduzione di un programma applicativo semplificherebbe la gestione dei vari settori di ogni punto vendita, migliorerebbe la gestione dell’inventario e l’organizzazione degli ordini con un controllo automatico delle quantità dei singoli prodotti.
Ogni dipendente, rispetto alla sua mansione, può consultare l’inventario per conoscere la disponibilità di un prodotto, visualizzare lo storico di una riparazione o di fatturazione di vendita e verificare lo status di una riparazione in corso. Infine, grazie al sistema, il gestore del singolo punto vendita può velocemente visualizzare i prodotti con quantità ridotte ed effettuare velocemente una richiesta d’ordine.
Per ogni prodotto venduto viene creato una nuova fattura con le informazioni del cliente, il metodo di pagamento e il prodotto venduto. Il sistema deve automaticamente aggiornare le quantità nell’inventario per ogni prodotto venduto.
Tale sistema viene applicato anche durante il processo di riparazione con l'aggiunta delle informazioni riguardo il prodotto da riparare, una breve descrizione sul processo di riparazione. Inoltre il modulo di riparazioni conterrà i prodotti (in questa fase pezzi di ricambio) che successivamente verrano eliminati dall’inventario.

Quindi possiamo visualizzare 3 utenze con diritto di utilizzo e accesso strettamente simile:

A - Il **Gestore:**
- Visualizza l’inventario
- Visualizza i moduli di vendita
- Visualizza i moduli di riparazioni
- Visualizza le schede informative dei clienti
- Crea una nuova tabella con i prodotti a limitata disponibilità

B - Il **Venditore:**
- Visualizza l’inventario
- Crea i moduli di vendita

C - Il **Tecnico:**
- Visualizza l’inventario
- Crea i moduli di riparazione
- Visualizza i moduli di riparazione
- Modifica i moduli di riparazione

2 - Analisi dei requisiti

2.1 - Descrizione del problema

Si vuole automatizzare l'organizzazione interna di diversi punti vendita per ottenere una completa gestione dei seguenti sistemi:

- Gestione dell'**inventario**
- Registrazione dei **clienti** sia in ambito di vendita sia per supporto tecnico
- Gestione degli **ordini** dei prodotti in esaurimento
- Visualizzazione di uno **storico** di un cliente

Attualmente i punti vendita utilizzano già un sistema **cartaceo** per catalogare il magazzino ma si desidera creare un sistema automatizzato per la creazione di un modulo di ordini con una selezione automatica dei prodotti a scarsa disponibilità tale da regolare il quantitativo dei prodotti in magazzino.
Le **fatture** sono compilate e stampate manualmente per ogni vendita tramite un modulo e si vuole quindi realizzare un sistema automatizzato di auto-compilazione prendendo i dati del cliente dal modulo vendita/riparazione.
I **Moduli** cartacei sono stampati singolarmente per ogni vendita/riparazione. Quelli di vendita sono poi conservati in fascicoli mentre quelli di riparazione sono applicati sul prodotto da riparare successivamente conservati nei fascicoli al momento della consegna del prodotto riparato. Questo sistema cartaceo crea problemi nella ricerca di uno storico ordine/riparazione a causa dell'enorme quantitativo di fascicoli. Si vuole quindi creare un sistema di ricerca di uno storico ordine/ riparazione tramite parole chiavi come CF del cliente.

2.2 - Specifiche delle operazioni

Si vogliono automatizzare le seguenti operazioni:

Operazione	
1	Registrazione dati di un nuovo cliente
2	Modifica dati di un Cliente
3	Creazione di un Modulo Riparazione
4	Creazione di un Modulo Vendita
5	Creazione di un Modulo Ordine
6	Modifica Indice nella lista riparazioni
7	Modifica indice nella lista Vendite
8	Modifica indice nella lista Ordine
9	Modifica indice nella lista prodotti di una vendita
10	Modifica indice nella lista prodotti in riparazione
11	Registrazione di un nuovo Dipendente
12	Modifica Indice Lista Dipendenti
13	Registrazione di un nuovo Prodotto
14	Visualizzazione delle riparazioni effettuate da un determinato Tecnico
15	Visualizzazione delle vendite effettuate da un determinato Commesso
16	Visualizzazione degli ordini effettuati da un determinato Gestore
17	Visualizzazione della quantità disponibile di un prodotto in un determinato P.Vendita
18	Visualizzazione degli acquisti effettuati da un determinato Cliente
19	Visualizzazione delle riparazioni richieste da un determinato Cliente
20	Visualizzazione dei prodotti con quantità < 3
21	Visualizzazione di una lista di prodotti appartenenti ad una marca specificata e quantitativo nei rispettivi store
22	Visualizzazione di una lista di prodotti appartenenti ad una marca specificata, < €480
23	Modifica del quantitativo di un prodotto nell'inventario rispetto all'ordine effettuato
24	Visualizzazione dello status di una riparazione di un cliente specificato
25	Visualizzazione di una lista di riparazioni "repair"
26	Visualizzazione di una lista di riparazioni "Completo"
27	Visualizzazione di una lista di riparazioni "Consegnato"
28	Modifica dello status di una riparazione
29	Vista di fatture rispettive alle vendite
30	Vista di fatture rispettive alle riparazioni
31	Vista di fatture rispettive agli ordini

2.2 - Glossario

Termini	Descrizione	Dati	Collegamenti
Dipendente	Ha diverse mansioni nel punto vendita	ID, Nome, Cognome, CF, Età, Data assunzione, Mansione, Reparto	Vendita, Riparazione
Cliente	Viene registrato nei moduli vendita/riparazione	CF, Nome, Cognome, Recapito tel	Vendita, Riparazione, Fattura
Fattura	Viene creata alla vendita di un prodotto	N°Fattura, vendita, prezzo, CF acquirente, data emissione	Vendita, Riparazione, Inventario, Prodotto, Cliente, Dipendente
Inventario	Raccoglie l'elenco dei prodotti nel magazzino	Prodotto, Quantità	Prodotto
Prodotto	Prodotto contento nel magazzino	ID, Marca, Modello	
Ordine	Una lista di Prodotti da ordinare per il magazzino	Prodotto, Quantità da acquistare	Prodotto
Riparazione	Modulo creato per un'assistenza tecnica	ID, Cliente, Data entrata, Prodotto, descrizione, Status, Data uscita, Prezzo	Cliente, Prodotto, Dipendente, Inventario
Vendita	Modulo creato per una vendita di un prodotto	ID, Cliente, Data emissione, prodotto, prezzo, metodo pagamento	Cliente, Dipendente, Inventario

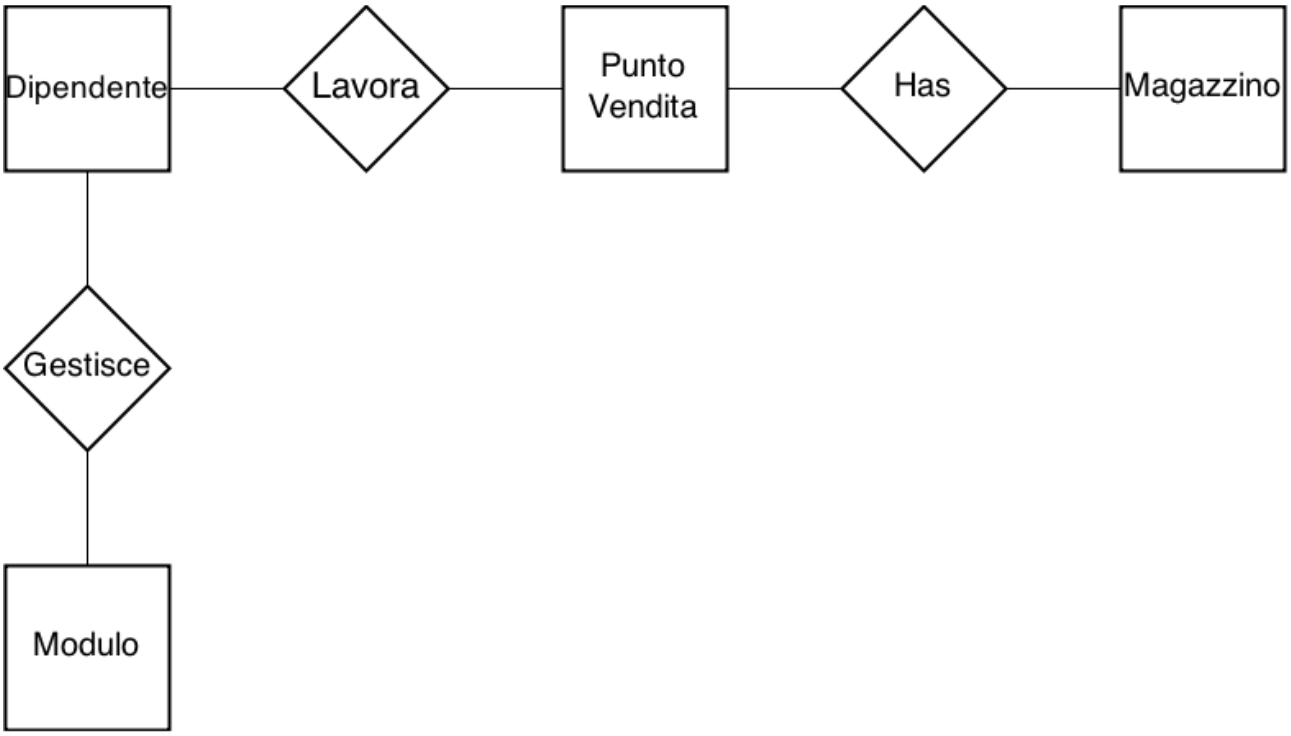
3 - Progettazione concettuale

3.1 - Schema concettuale

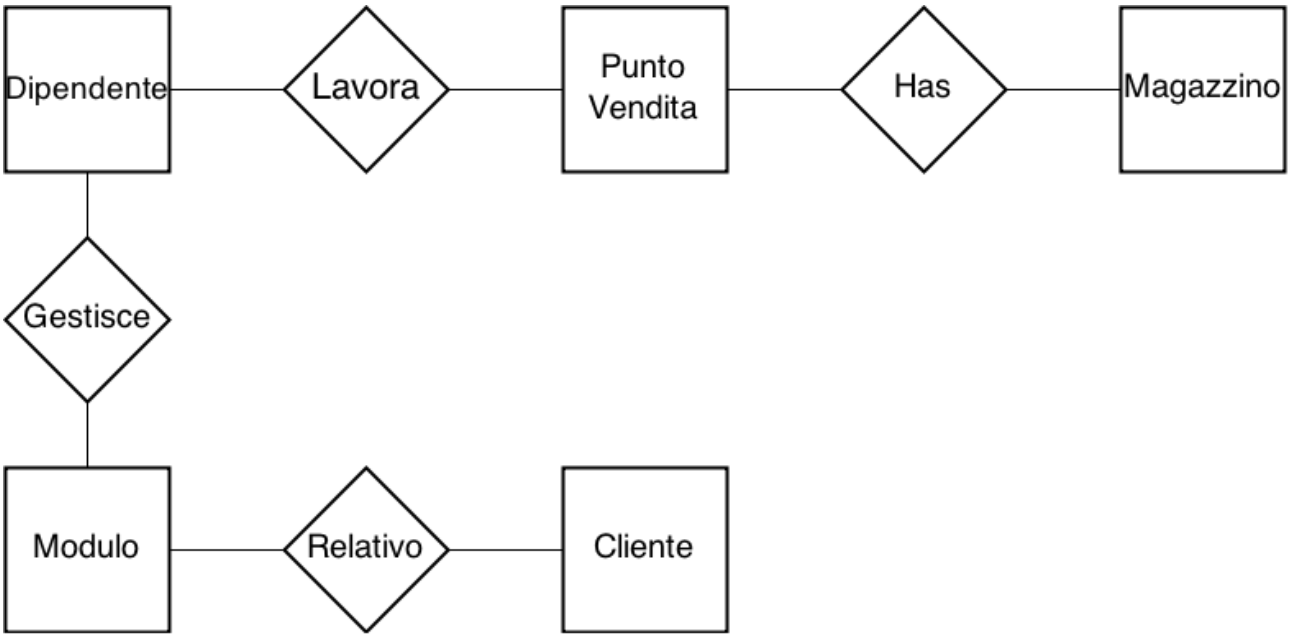
Sviluppo degli schemi secondo la metodologia **top-down**.

- Dalle specifiche del problema abbiamo generato un primo **schema a scheletro.** Tramite le entità “punto vendita”, “dipendente” e “magazzino” rappresentiamo uno schema principale. Esistono relazioni tra l’entità “dipendente” e “punto vendita” (lavora) e tra “punto vendita” e “magazzino” (has).

- Viene poi aggiunta l’entità “modulo” che rappresenterà tutti i moduli creati, per ora generalizzati, sia per la vendita che per la riparazione. Questa entità avrà una relazione con l’entità “dipendente” (gestisce).



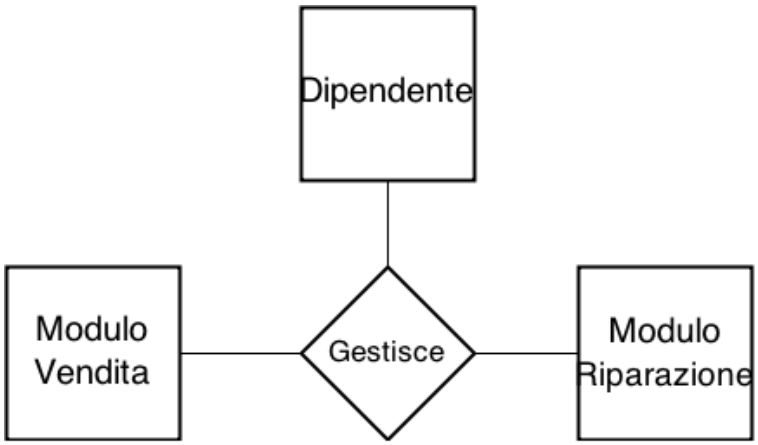
- Ora che abbiamo aggiunto l'entità "modulo" possiamo aggiungere a sua volta l'entità "cliente" che sarà in relazione con l'entità "modulo".



- Successivamente vengono aggiunti a ciascuna entità i rispettivi attributi

Entità	Attributi	Descrizione
Dipendente	CF Nome Cognome indirizzo Settore Data ass	Codice Fiscale Nome Cognome Indirizzo Lavoro/mansione Data di assunzione
Punto Vendita	Zona Gestore	Indirizzo Riferimento al gestore
Magazzino	Marca Modello Prezzo Costo Quantità	Marca prodotto Modello prodotto Prezzo al pubblico Costo Quantità disponibile
Cliente	CF Nome Cognome Indirizzo Telefono	Codice Fiscale Nome Cognome Indirizzo Recapito telefonico

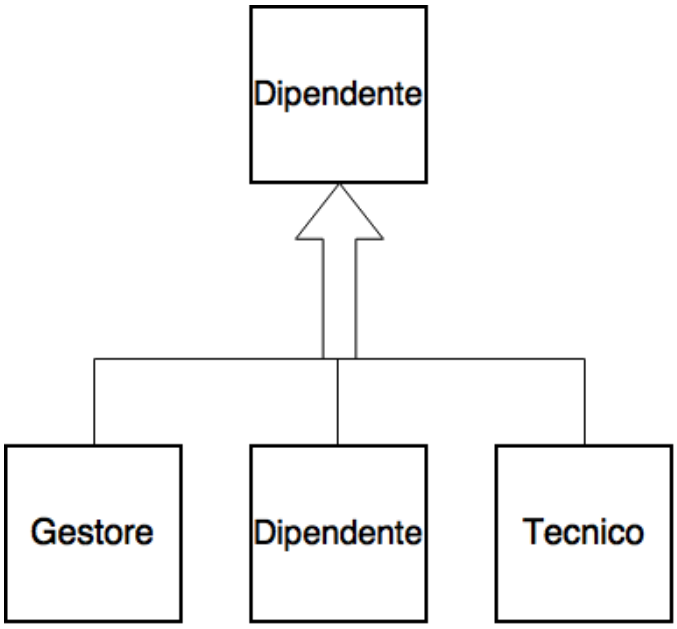
- Si prosegue con un ulteriore raffinamento suddividendo l'entità "modulo" in due entità separate: "**modulo riparazioni**" e "**modulo vendita**" che mantengono la medesima relazione con l'entità "dipendente". In seguito vengono aggiunti degli attributi distinti per ognuna delle due entità.



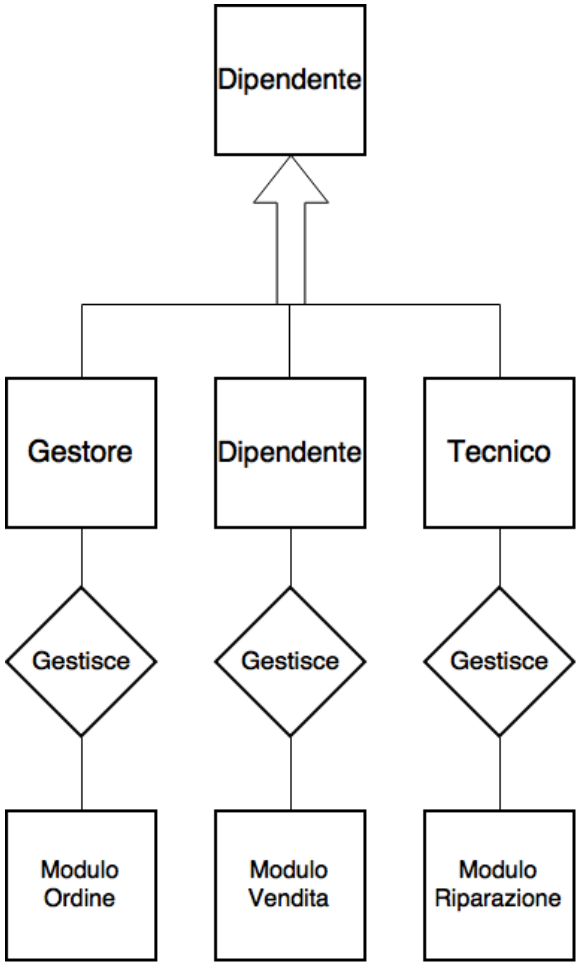
- Possiamo ora proseguire aggiungendo gli attributi necessari per i due moduli.

Entità	Attributi	Descrizione
Modulo Vendita	ID_vendita Data Cliente Prodotto Prezzo Operatore	Identificatore univoco Data della vendita Cliente Prodotto/i venduto/i Prezzo al cliente (totale) Dipendente in servizio
Modulo Riparazione	ID_Riparazione DataE Cliente ProdottoRip Difetto Prezzo DataU Tecnico	Identificatore univoco Data di entrata Cliente Prodotto in riparazione Difetto del prodotto Prezzo della riparazione Data di uscita Tecnico incaricato

- Considerando che ogni dipendente ha degli incarichi specifici a seconda del ruolo che ricopre all'interno del punto vendita l'entità "dipendente" viene suddivisa in **tre entità figlie**: "gestore", "dipendente" e "tecnico".

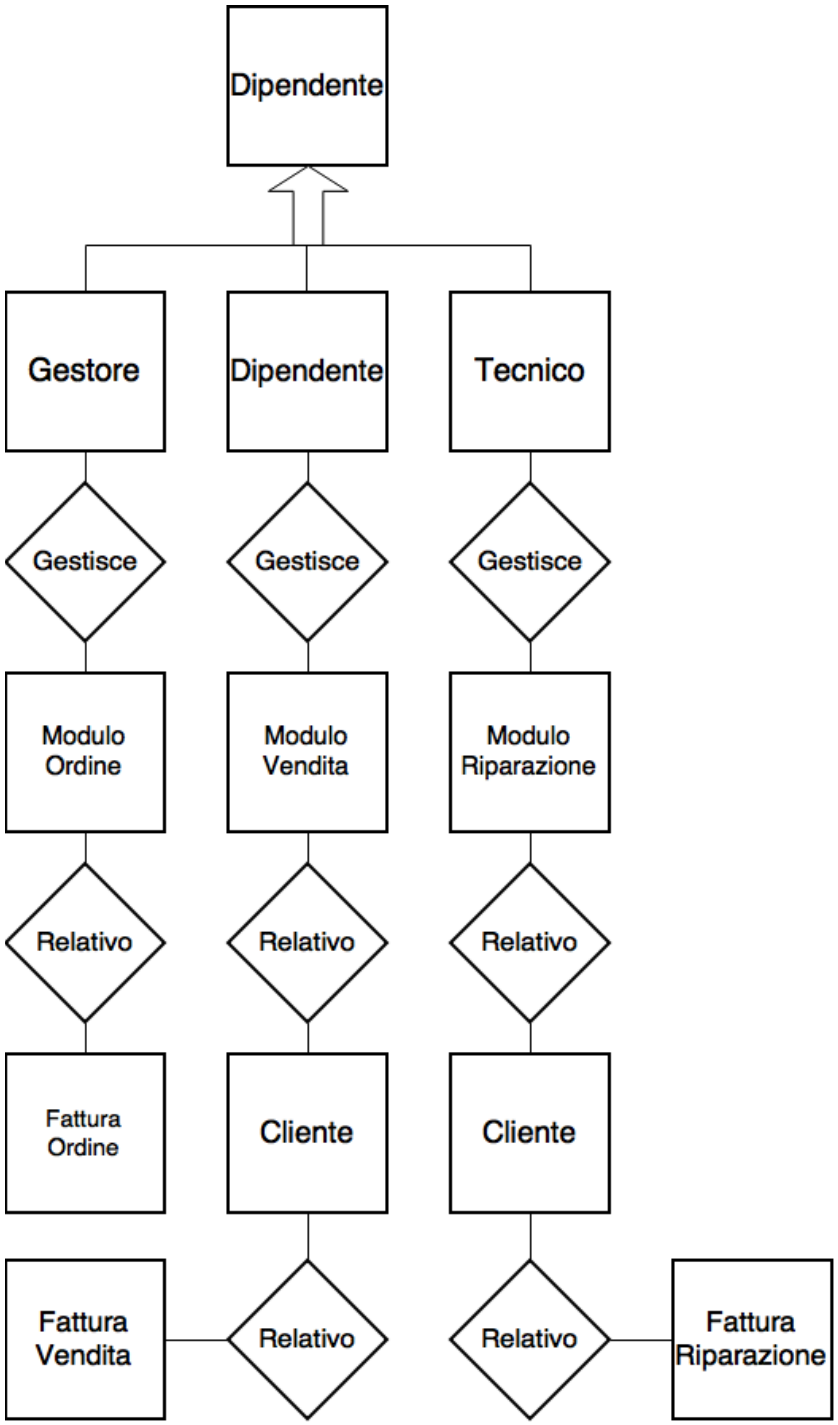


- I tre dipendenti sono si distinti ma non hanno bisogno di un elenco di attributi separati in quanto l'unica distinzione è il loro ruolo e la relazione separata con i diversi moduli (vendita, riparazione, ordine).
- Analogamente all'entità "dipendente" anche l'entità "modulo" viene suddivisa in tre entità differenti

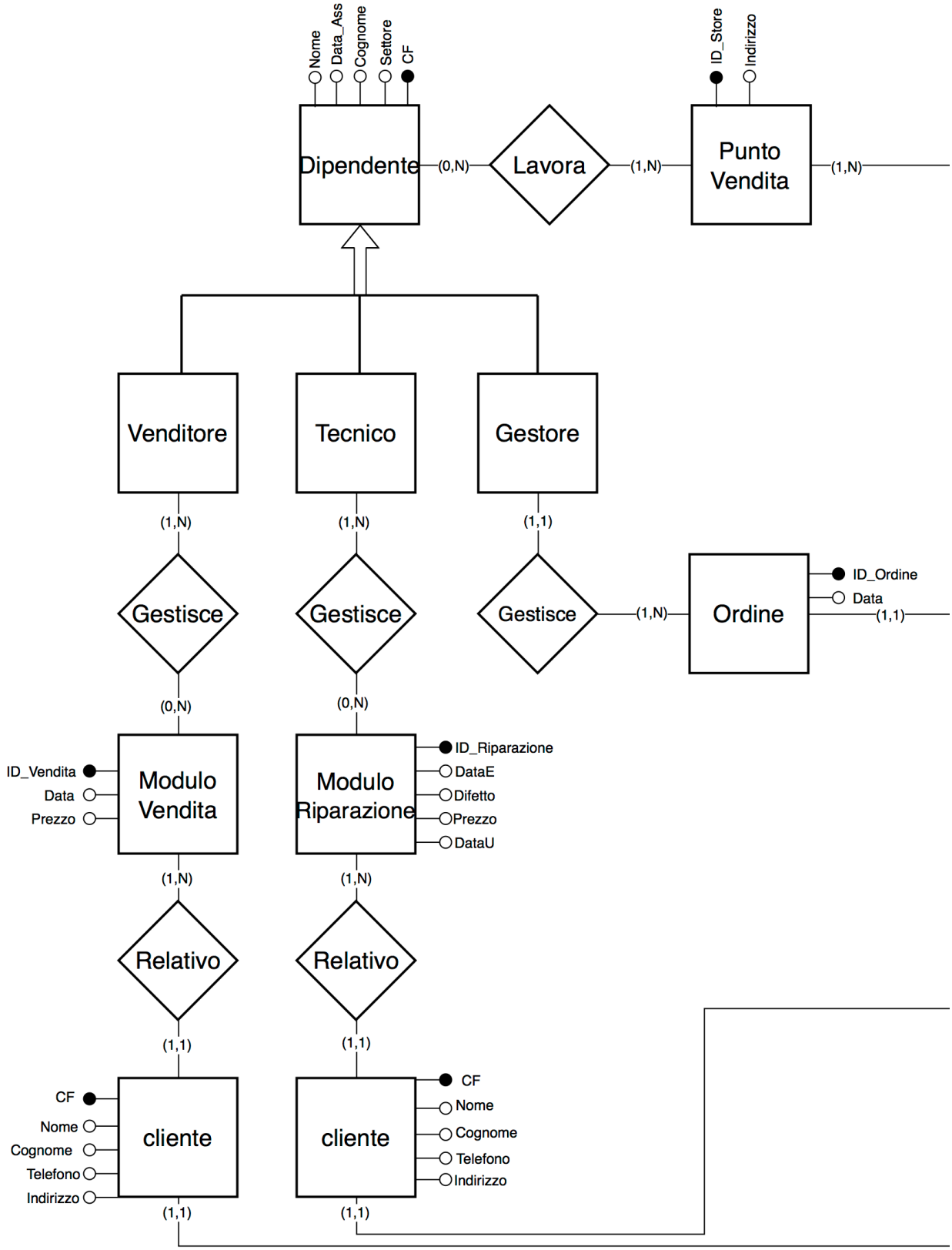


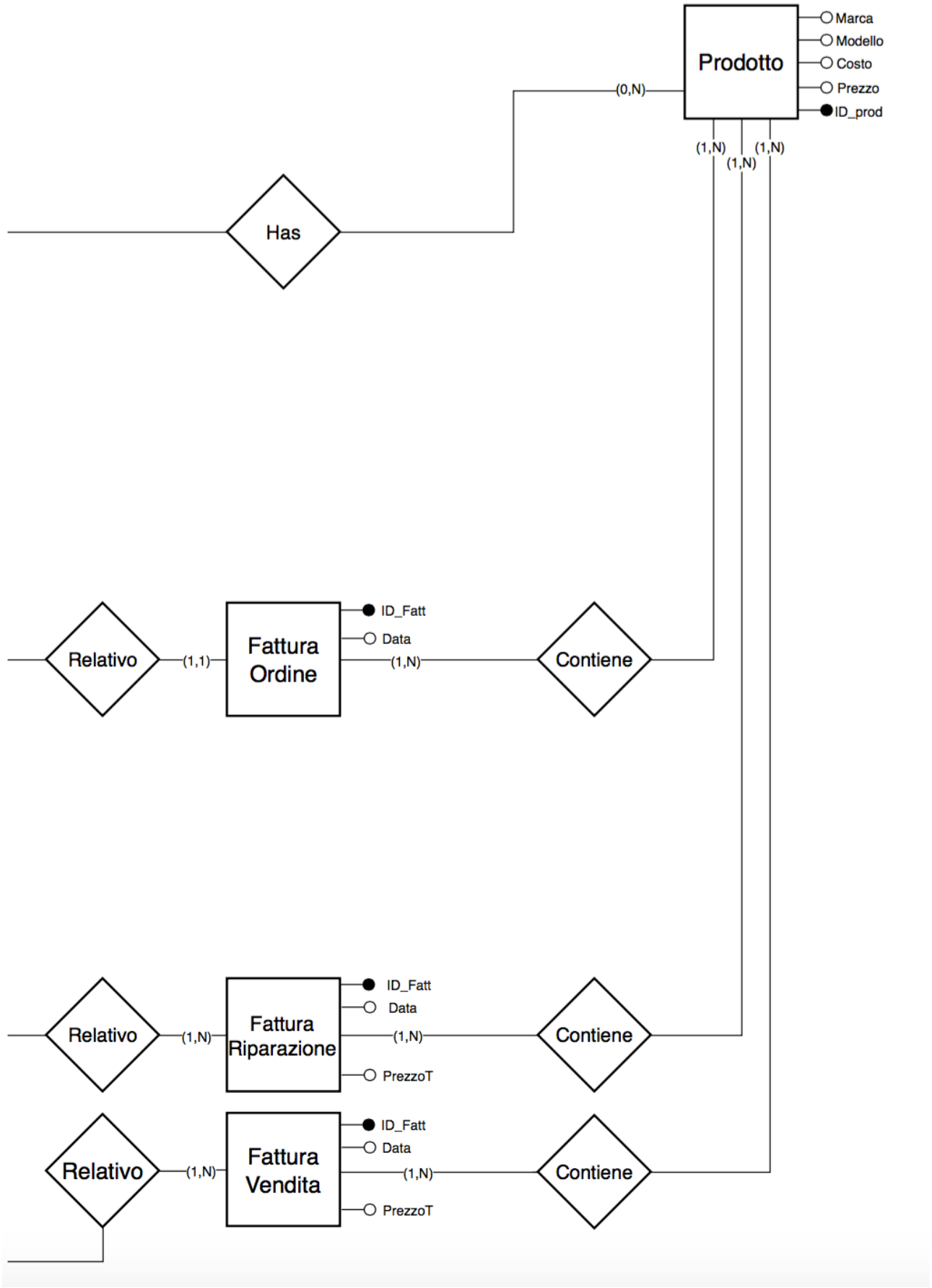
- I moduli di vendita e di riparazione sono *simili* in quanto entrambi legati, tramite una relazione, al cliente che a sua volta è messo in relazione con la fattura che verrà generata dopo il servizio di vendita o riparazione. Il **modulo ordine** invece è messo direttamente in relazione con la fattura poiché si tratta di una procedura interna gestita esclusivamente dal gestore di ogni punto vendita.

Entità	Attributo	Descrizione
Fattura Ordine	ID_Fattura Data	Codice identificativo Data di emissione Fattura
Fattura Riparazione	ID_Fattura_R Data PrezzoT	Codice identificativo Data di emissione Fattura Prezzo totale
Fattura Vendita	ID_Fattura_V Data PrezzoT	Codice identificativo Data emissione Fattura Prezzo al totale



- Concludiamo inoltre con l'inserimento degli attributi sullo schema e l'inserimento delle cardinalità


Nome
Data_Ass
Cognome
Settore
CF
Dipendente
(0,N)
Lavora
(1,N)
ID_Store
Indirizzo
Punto Vendita
(1,N)
Venditore
Tecnico
Gestore
(1,N)
(1,N)
(1,1)
Gestisce
Gestisce
Gestisce
(1,N)
Ordine
ID_Ordine
Data
(1,1)
(0,N)
(0,N)
ID_Vendita
Data
Prezzo
Modulo Vendita
Modulo Riparazione
ID_Riparazione
DataE
Difetto
Prezzo
DataU
(1,N)
(1,N)
Relativo
Relativo
(1,1)
(1,1)
CF
Nome
Cognome
Telefono
Indirizzo
cliente
cliente
CF
Nome
Cognome
Telefono
Indirizzo
(1,1)
(1,1)



- Procedendo con la progettazione si decide di modificare lo schema eliminando l'entità Magazzino e inserendo un' entità Prodotto in quanto la tabella Magazzino si andrà a creare automaticamente come tabella intermedia tra Prodotto e Punto_Vendita per la relazione n-m

- Procediamo con un elenco complessivo delle entità e dei relativi attributi

Entità	Descrizione	Attributi
Punto_Vendita	Entità che raccoglie le informazioni di ogni punto vendita	**ID_Store (PK)** Indirizzo
Prodotto	Raccoglie le informazioni di tutti i prodotti vendibili dai Punti_Vendita	**ID_prod (PK)** Marca Modello Categoria Costo Prezzo
Dipendente	Raccoglie informazioni relative alle persone che lavorano nei Punti_Vendita	**CF (PK)** Nome Cognome Data_Assunzione Settore
Cliente	Raccoglie le informazioni dei clienti venuti a registrare in seguito ad una vendita o richiesta di riparazione	**CF (CF)** Nome Cognome Indirizzo Telefono
Modulo_Vendita	Raccoglie le informazioni relative ad una vendita	**ID_Vendita (PK)** Data Prezzo_T **Cliente_CF (FK)**
Modulo_Riparazione	Raccoglie informazioni relative ad una richiesta di assistenza tecnica	**ID_Riparazione** Difetto Data_Entrata Data_Uscita **Cliente_CF (FK)** Prezzo Stato
Modulo_Ordine	Raccoglie alcune informazioni relative ad un ordine di prodotti per il rifornimento del magazzino	**ID_Ordine (PK)** Data **Dipendente_CF (FK)**
Fattura_Vendita	Raccoglie i dati finali di una vendita	**ID_Fatt_Ven (PK)** Data **Cliente_CF (FK)** Nome Cognome Indirizzo Prezzo_T
Fattura_Riparazione	Raccoglie i dati finali di una riparazione	**ID_Fatt_Rip (PK)** Data_E Data_U **Cliente_CF (FK)** Nome Cognome Indirizzo Difetto Prezzo_T
Fattura_Ordine	Raccoglie i dati finali di un ordine	**ID_Fatt_ord (PK)** Data Prezzo_T

- Procediamo con l’elenco delle associazioni

Relazione	Descrizione	Entità Coinvolte
Lavora	Associa un Dipendente ad un punto vendita	Dipendente (1,1) Punto Vendita (1,N)
Gestisce	Associa un venditore ad un modulo vendita	Venditore (1,N) Modulo_Vendita (0,N)
Gestisce	Associa un tecnico ad un modulo riparazione	Tecnico (1,N) Modulo_Riparazione (0,N)
Gestisce	Associa un gestore ad un ordine	Gestore (1,1) Ordine (1,N)
Relativo	Associa un modulo vendita ad un cliente	Modulo_Vendita (1,N) Cliente (1,1)
Relativo	Associa un modulo riparazione a un cliente	Modulo_Riparazione (1,N) Cliente (1,1)
Relativo	Associa una fattura vendita ad un cliente	Fattura_Vendita (1,N) Cliente (1,1)
Relativo	Associa una fattura riparazione ad un cliente	Fattura_Riparazione (1,N) Cliente (1,1)
Relativo	Associa una fattura ordine ad un ordine	Fattura_Ordine (1,1) Ordine (1,1)
Contiene	Associa dei prodotti ad una fattura ordine	Fattura_Ordine (1,N) Prodotto (1,N)
Contiene	Associa dei prodotti ad una fattura riparazione	Fattura_Riparazione (1,N) Prodotto (1,N)
Contiene	Associa dei prodotti ad una fattura di vendita	Fattura_Vendita (1,N) Prodotto (1,N)
Has	Associa dei prodotti appartenenti ai vari punti vendita	Punto_Vendita (1,N) Prodotto (0,N)

3.2 Vincoli di integrità dei dati

Regole di Vincolo

RV	
1	Non è ammessa la duplicazione di tuple nelle tabelle
2	Gli identificatori delle entità non possono avere valore nullo
3	Non possono essere registrati due clienti con lo stesso numero di telefono
4	Non può essere registrato un dipendente in più di un punto vendita
5	Non è possibile compilare un modulo vendita/Riparazione senza la registrazione del cliente
6	Non è possibile compilare un modulo di vendita se il prodotto è inesistente
7	Non è possibile registrare un modulo riparazione se il prodotto non esiste nell'elenco prodotti
8	Durante la registrazione di un cliente il numero di telefono è strettamente necessario
9	Non è possibile inserire due o più prodotti con la stessa PK in un elenco vendita/riparazione

3.3 Analisi della qualità

CORRETTEZZA Lo schema è corretto in quanto sono stati utilizzati correttamente i costrutti del modello Entità - Relazione.

COMPLETEZZA Lo schema concettuale rappresenta tutti i dati di interesse e tutte le operazioni possono essere eseguite a partire dai concetti descritti. Per questo lo schema può essere definito completo.

LEGGIBILITA' Lo schema concettuale è leggibile, autoesplicativo e rappresenta i requisiti in maniera naturale e facilmente comprensibile.

MINIMALITA' I dati sono stati sintetizzati nel migliore dei modi per avere da una parte uno schema semplice e dall'altra un'ottima flessibilità dei dati in modo tale da diminuire le ricerche con grandi tabelle, ma usare piccole tabelle e raggrupparle tra loro solo quando è necessario.

4 Progettazione Logica

4.1 Analisi delle prestazioni

Tabella Volumi	
Punto Vendita	3
Dipendente	30
Prodotto	4000
Modulo_Vendita	50.000/Anno
Modulo_Riparazione	2.000/Anno
Modulo_Ordine	40/Anno
Cliente	50.000/Anno
Fattura_Riparazione	2.000/Anno
Fattura_Vendita	50.000/Anno
Fattura_Ordine	40/Anno

Tabella Operazioni	
Op1	50.000/Anno
Op2	1000/Anno
Op3	2.000/Anno
Op4	50.000/Anno
Op5	40/Anno
Op6	50.000/Anno
Op7	50.000/Anno
Op8	40/Anno
Op9	100.000/Anno
Op10	2.000/Anno
Op11	50.000/Anno
Op12	5/Anno
Op13	500/Anno
Op14	20/Giorno
Op15	15/Giorno
Op16	5/Mese
Op17	20/Giorno
Op18	2/Giorno
Op19	4/Giorno
Op20	1/Settimana
Op21	5/Giorno
Op22	1/Giorno
Op23	1/Mese
Op24	5/Giorno
Op25	20/Giorno
Op26	5/Giorno
Op27	2/Giorno
Op28	30/Giorno
Op29	130/Giorno
Op30	6/Giorno
Op31	1/Mese

4.2 Ristrutturazione dello schema E-R

- Riorganizzazione e ottimizzazione dello schema concettuale per la sepmlificazione della traduzione allo schema logico

4.2.1 ANALISI DELLE RIDONDANZE ED ELIMINAZIONE DELLE GERARCHIE

Una delle modifiche da apportare per alleggerire il DataBase è quello di eliminare le tre Entità Fatture e renderle visibili solo tramite a delle istruzioni di Viste.

In questo modo ottimizziamo la memoria del database eliminando un gran numero di tuple (circa 55.000/Anno). L'idea è quella di utilizzare tre istruzioni di tipo Vista che creano tabelle momentanee attue alla visualizzazione di dati accorpati prelevati da varie tabelle.

Questo meccanismo richiede un leggero calcolo in più al DB ma può risparmiare molta memoria, tramite le istruzioni viste le informazioni di un cliente, prelevati dalla tabella Cliente, le informazioni di un modulo vendita o riparazioni, prelevati dalla tabella Modulo_Vendita, Modulo_Riparazioni vengono tutte accorpate in un unica tupla.

Per la creazione di un sistema flessibile, il database presenta molte tabelle intermedie necessarie per il rilevamento di relazione tra un' entità e l'altra, data l'impossibilità di anticipare la creazione di tuple nelle tabelle intermedie prima delle loro entità principali abbiamo optato nel ottimizzare l'automatizzazione della modifica dei quantitativi dei prodotti nell'inventario e nel calcolo del prezzo totale di una vendita/riparazione/ordine sfruttando algoritmi di calcolo tramite il software di interfaccia.

In questo modo evitiamo di utilizzare TRIGGER precedentemente testati e non funzionanti a causa di incompatibilità con il tipo di sistema di tabelle e optiamo con l'uso di algoritmi inseriti all'interno del software di interfaccia che, nel momento in cui viene effettuata una vendita il software esegue una serie di istruzioni verso il database in una sequenza ben precisa: 1- Creazione di un nuovo cliente (se non esistente) 2- creazione di un modulo vendita (senza il prezzo) 3- Creazione delle tuple nella tabella Lista_Prodotti_Vendita 4- Il software calcola il prezzo complessivo 5- Modifica della tupla Modulo_Vendita appena creata inserendo il prezzo appena calcolato.

Questo meccanismo funziona allo stesso modo per l'incremento e decremento della quantità nel magazzino: Una volta creato un modulo vendita successivamente il software esegue l'operazione di modifica del quantitativo prodotto magazzino, idem nel caso di un ordine.

4.2.2 PARTIZIONAMENTO/ACCORPAMENTO DI CONCETTI

Non sono necessari opportuni accorpamenti o partizionamenti in quanto le entità di fattura sono già state trasformate in viste.

4.2.3 ELIMINAZIONE DI ATTRIBUTI COMPOSTI

Si ritiene che il sistema possiede già una corretta composizione e un corretto utilizzo dei dati, rendendo inutile un' ulteriore modifica.

4.3 traduzione verso il modello relazionale

- Di seguito verra affrontato il mappino delle associazioni binarie

  PK FK*

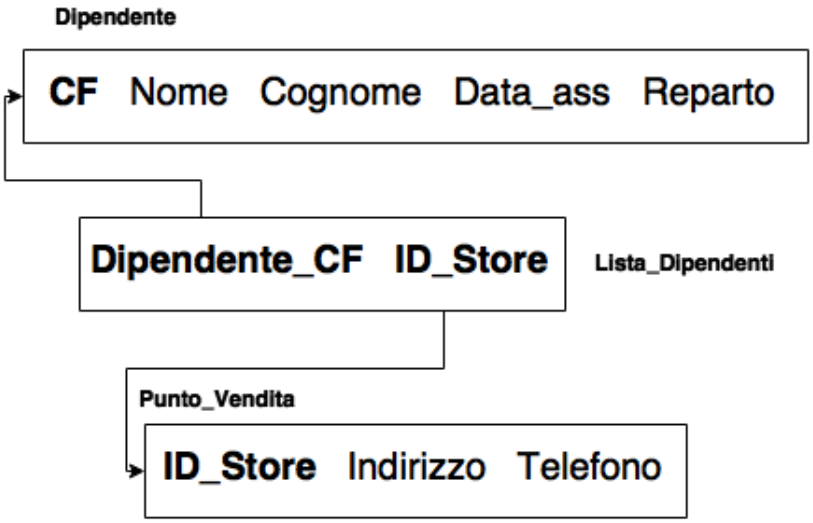
4.3.1 LISTA_DIPENDENTI

- Associazione M:N tra Dipendente e Punto_Vendita

Dipendente (CF, Nome, Cognome, Data_Assunzione, Settore)

Punto_Vendita (ID_Store, Indirizzo, Telefono)

Lista_Dipendenti (ID_Store, Dipendente_CF)



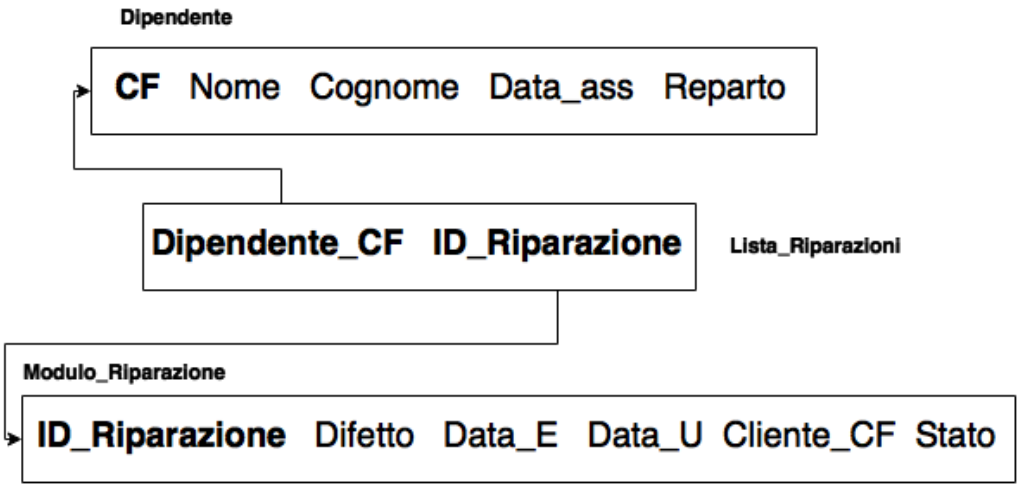
4.3.2 LISTA_RIPARAZIONI

- Associazione M:N tra Dipendente e Modulo_riparazione

Dipendente (CF, Nome, Cognome, Data_Assunzione, Reparto)

Modulo_Riparazione (ID_Riparazione, Difetto, Data_E, Data_U, Cliente_CF*, Stato)

Lista_Riparazioni (ID_Riparazione, Dipendente_CF)



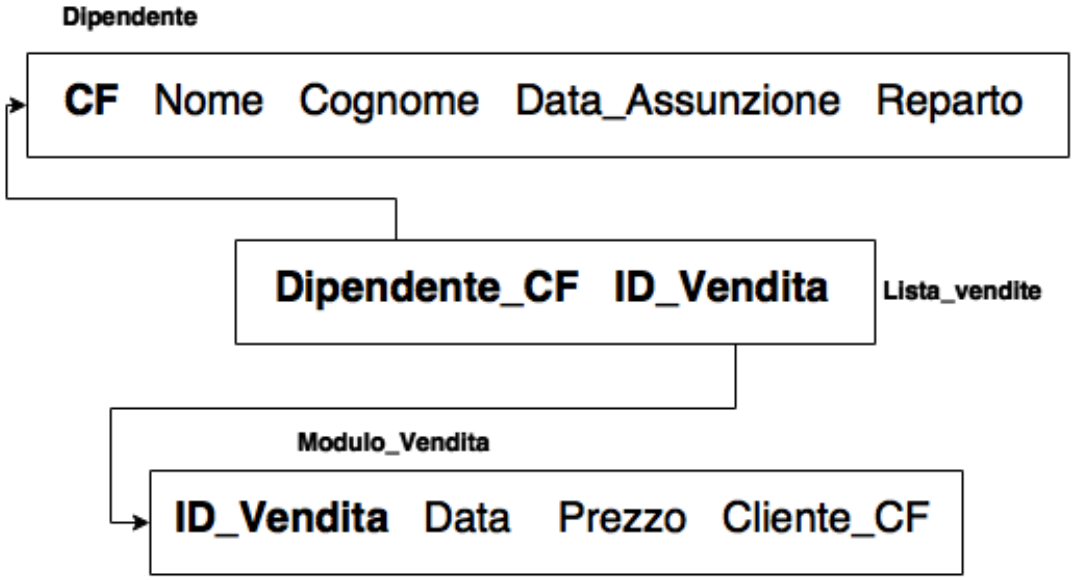
4.3.3 LISTA_VENDITE

- Associazione M:N tra Dipendente e Modulo_Vendita

Dipendente (CF, Nome, Cognome, Data_Assunzione, Reparto)

Modulo_Vendita (ID_Vendita, Data, Cliente_CF*)

Lista_Vendite (ID_Vendita, Dipendente_CF)



4.3.4 LISTA_PRODOTTI_VENDITA

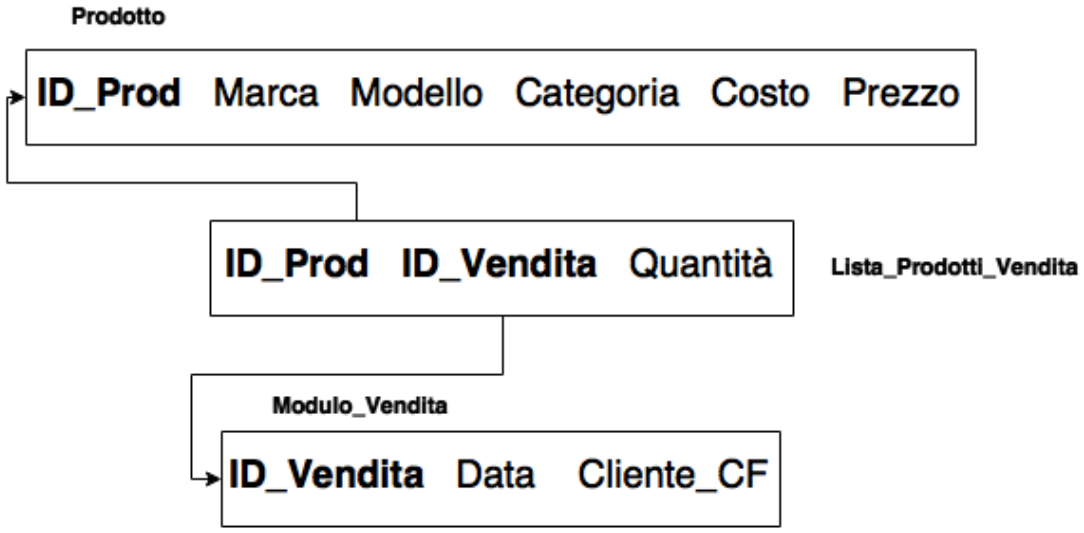
- Associazione M:N tra Prodotto e Modulo_Vendita

- Si aggiunge l'attributo Quantità per specificare il quantitativo di prodotti in una vendita

Modulo_Vendita (ID_Vendita, Data, Cliente_CF*)

Prodotto (ID_Prod, Marca, Modello, Categoria, Costo, Prezzo)

Lista_Prodotti_Vendita (ID_Vendita, ID_Prod, Quantità)



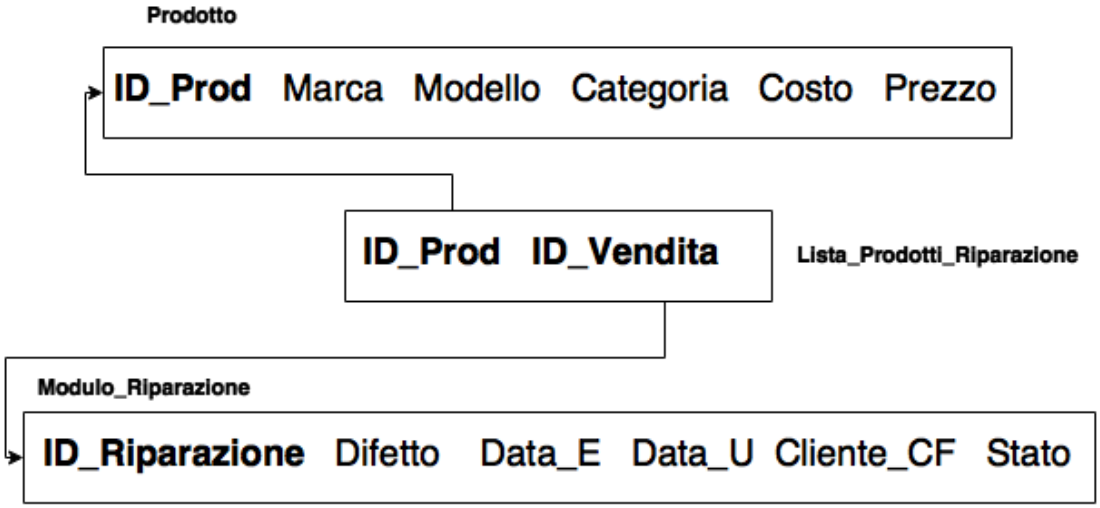
4.3.5 LISTA_PRODOTTI_RIPARAZIONE

- Associazione M:N tra Prodotto e Modulo_Riparazione

Prodotto (ID_Prod, Marca, Modello, Categoria, Costo, Prezzo)

Modulo_Riparazione (ID_Riparazione, Difetto, Data_E, Data_U, Cliente_CF*, Stato)

Lista_Prodotti_Riparazione (ID_Prod, ID_Riparazione)



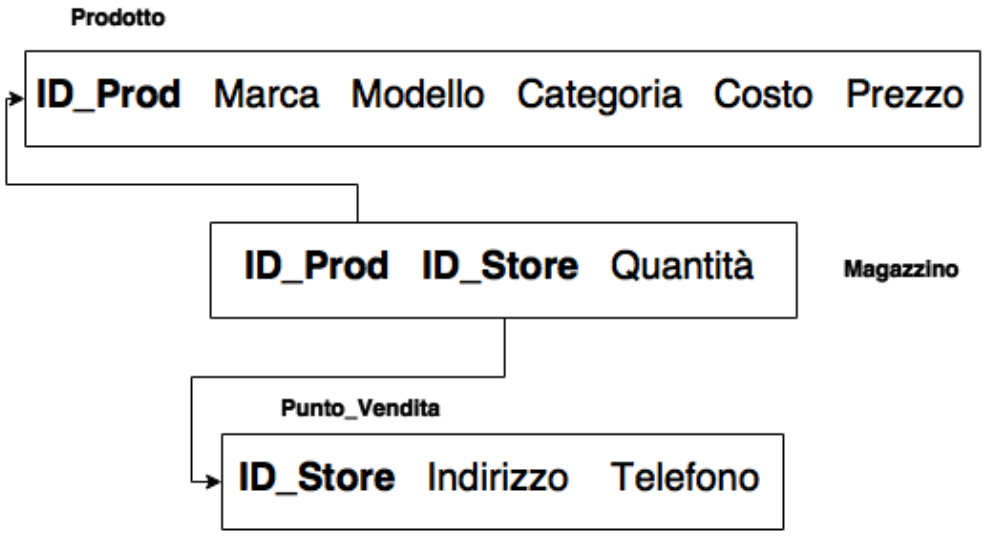
4.3.6 MAGAZZINO

- Associazione N:M tra Prodotto e Punto_Vendita
- Viene aggiunto anche un attributo Quantità per registrare il quantitativo del prodotto in uno specifico punto vendita

Prodotto (ID_Prod, Marca, Modello, Categoria, Costo, Prezzo)

Punto_Vendita (ID_Store, Indirizzo, Telefono)

Magazzino (ID_Store, ID_Prod, Quantità)



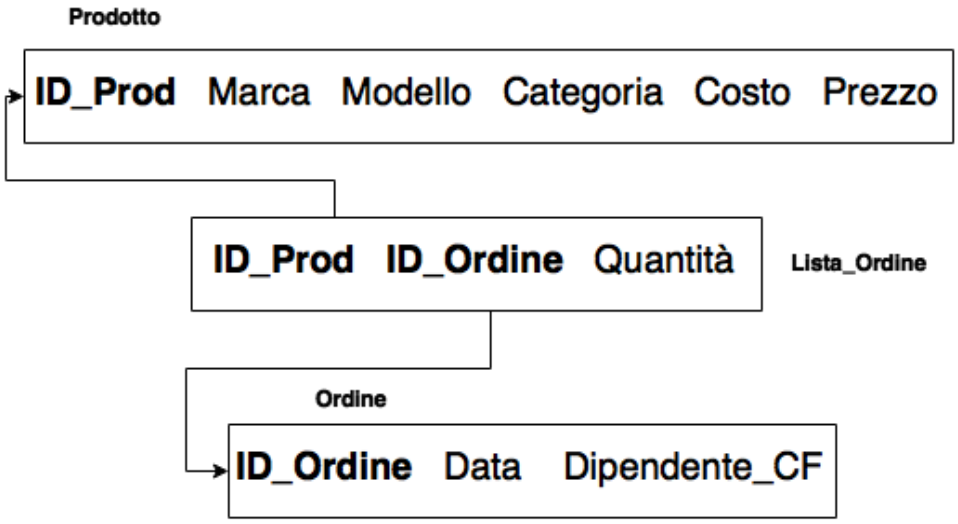
4.3.7 LISTA_ORDINE

- Associazione N:M tra Prodotto e Ordine
- Viene aggiunto l'attributo quantità per specificare il quantitativo del singolo prodotto da ordinare

Prodotto (ID_Prod, Marca, Modello, Categoria, Costo, Prezzo)

Ordine (ID_Ordine, Data, Dipendente_CF*)

Lista_Ordine (ID_Prod, ID_Ordine, Quantità)



5 Progettazione Fisica

Alcuni elementi sono prelevati dal forward engineering di Workbench

5.1 Definizione dello schema di base di dati

```
CREATE SCHEMA IF NOT EXISTS `mydb`;

USE mydb;
```

5.2 Definizione delle tabelle

```
-- -----------------------------------------------------
-- Table `mydb`.`Dipendente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Dipendente` (
  `CF` VARCHAR(16) NOT NULL,
  `Nome` VARCHAR(20) NOT NULL,
  `Cognome` VARCHAR(20) NOT NULL,
  `Data_Ass` DATE NOT NULL,
  `Settore` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`CF`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Punto_Vendita`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Punto_Vendita` (
  `ID_Store` INT NOT NULL,
  `Indirizzo` VARCHAR(30) NOT NULL,
  `Telefono` DOUBLE NOT NULL,
  PRIMARY KEY (`ID_Store`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Cliente` (
  `CF` VARCHAR(16) NOT NULL,
  `Nome` VARCHAR(20) NOT NULL,
  `Cognome` VARCHAR(20) NOT NULL,
  `Telefono` DOUBLE NOT NULL,
  `Indirizzo` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CF`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Modulo_Vendita`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Modulo_Vendita` (
  `ID_Vendita` INT NOT NULL,
  `Data` DATE NOT NULL,
  `Prezzo` FLOAT NULL,
  `Cliente_CF` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`ID_Vendita`),
  INDEX `fk_Modulo_Vendita_Cliente1_idx` (`Cliente_CF` ASC),
  CONSTRAINT `fk_Modulo_Vendita_Cliente1`
    FOREIGN KEY (`Cliente_CF`)
    REFERENCES `mydb`.`Cliente` (`CF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Modulo_Riparazione`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Modulo_Riparazione` (
  `ID_Riparazione` INT NOT NULL,
  `Difetto` VARCHAR(100) NULL,
  `Data_E` DATE NOT NULL,
  `Data_U` DATE NULL,
  `Cliente_CF` VARCHAR(16) NOT NULL,
  `Prezzo` FLOAT NULL,
  `Stato` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`ID_Riparazione`),
  INDEX `fk_Modulo_Riparazione_Cliente1_idx` (`Cliente_CF` ASC),
  CONSTRAINT `fk_Modulo_Riparazione_Cliente1`
    FOREIGN KEY (`Cliente_CF`)
    REFERENCES `mydb`.`Cliente` (`CF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Prodotto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Prodotto` (
  `ID_prod` INT NOT NULL,
  `Marca` VARCHAR(20) NOT NULL,
  `Modello` VARCHAR(20) NOT NULL,
  `Categoria` VARCHAR(20) NOT NULL,
  `Costo` FLOAT NOT NULL,
  `Prezzo` FLOAT NOT NULL,
  PRIMARY KEY (`ID_prod`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Ordine`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Ordine` (
  `ID_Ordine` INT NOT NULL,
  `Data` DATE NOT NULL,
  `Dipendente_CF` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`ID_Ordine`),
  INDEX `fk_Ordine_Dipendente1_idx` (`Dipendente_CF` ASC),
  CONSTRAINT `fk_Ordine_Dipendente1`
    FOREIGN KEY (`Dipendente_CF`)
    REFERENCES `mydb`.`Dipendente` (`CF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Lista_Riparazioni`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Lista_Riparazioni` (
  `Dipendente_CF` VARCHAR(16) NOT NULL,
  `Modulo_Riparazione_ID_Riparazione` INT NOT NULL,
  PRIMARY KEY (`Dipendente_CF`, `Modulo_Riparazione_ID_Riparazione`),
  INDEX `fk_Dipendente_has_Modulo_Riparazione_Modulo_Riparazione1_idx`
(`Modulo_Riparazione_ID_Riparazione` ASC),
  INDEX `fk_Dipendente_has_Modulo_Riparazione_Dipendente_idx` (`Dipendente_CF` ASC),
  CONSTRAINT `fk_Dipendente_has_Modulo_Riparazione_Dipendente`
    FOREIGN KEY (`Dipendente_CF`)
    REFERENCES `mydb`.`Dipendente` (`CF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Dipendente_has_Modulo_Riparazione_Modulo_Riparazione1`
    FOREIGN KEY (`Modulo_Riparazione_ID_Riparazione`)
    REFERENCES `mydb`.`Modulo_Riparazione` (`ID_Riparazione`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Lista_Vendite`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Lista_Vendite` (
  `Modulo_Vendita_ID_Vendita` INT NOT NULL,
  `Dipendente_CF` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`Modulo_Vendita_ID_Vendita`, `Dipendente_CF`),
  INDEX `fk_Modulo_Vendita_has_Dipendente_Dipendente1_idx` (`Dipendente_CF` ASC),
  INDEX `fk_Modulo_Vendita_has_Dipendente_Modulo_Vendita1_idx`
(`Modulo_Vendita_ID_Vendita` ASC),
  CONSTRAINT `fk_Modulo_Vendita_has_Dipendente_Modulo_Vendita1`
    FOREIGN KEY (`Modulo_Vendita_ID_Vendita`)
    REFERENCES `mydb`.`Modulo_Vendita` (`ID_Vendita`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Modulo_Vendita_has_Dipendente_Dipendente1`
    FOREIGN KEY (`Dipendente_CF`)
    REFERENCES `mydb`.`Dipendente` (`CF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Lista_Dipendenti`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Lista_Dipendenti` (
  `Dipendente_CF` VARCHAR(16) NOT NULL,
  `Punto_Vendita_ID_Store` INT NOT NULL,
  PRIMARY KEY (`Dipendente_CF`, `Punto_Vendita_ID_Store`),
  INDEX `fk_Dipendente_has_Punto Vendita_Punto Vendita1_idx` (`Punto_Vendita_ID_Store`
ASC),
  INDEX `fk_Dipendente_has_Punto Vendita_Dipendente1_idx` (`Dipendente_CF` ASC),
  CONSTRAINT `fk_Dipendente_has_Punto Vendita_Dipendente1`
    FOREIGN KEY (`Dipendente_CF`)
    REFERENCES `mydb`.`Dipendente` (`CF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Dipendente_has_Punto Vendita_Punto Vendita1`
    FOREIGN KEY (`Punto_Vendita_ID_Store`)
    REFERENCES `mydb`.`Punto_Vendita` (`ID_Store`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Magazzino`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Magazzino` (
  `Prodotto_ID_prod` INT NOT NULL,
  `Punto_Vendita_ID_Store` INT NOT NULL,
  `Quantità` INT NOT NULL,
  PRIMARY KEY (`Prodotto_ID_prod`, `Punto_Vendita_ID_Store`),
  INDEX `fk_Prodotto_has_Punto Vendita_Punto Vendita1_idx` (`Punto_Vendita_ID_Store` ASC),
  INDEX `fk_Prodotto_has_Punto Vendita_Prodotto1_idx` (`Prodotto_ID_prod` ASC),
  CONSTRAINT `fk_Prodotto_has_Punto Vendita_Prodotto1`
    FOREIGN KEY (`Prodotto_ID_prod`)
    REFERENCES `mydb`.`Prodotto` (`ID_prod`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Prodotto_has_Punto Vendita_Punto Vendita1`
    FOREIGN KEY (`Punto_Vendita_ID_Store`)
    REFERENCES `mydb`.`Punto_Vendita` (`ID_Store`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Lista_Ordine`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Lista_Ordine` (
  `Prodotto_ID_prod` INT NOT NULL,
  `Ordine_ID_Ordine` INT NOT NULL,
  `Quantità` INT NOT NULL,
  PRIMARY KEY (`Prodotto_ID_prod`, `Ordine_ID_Ordine`),
  INDEX `fk_Prodotto_has_Ordine_Ordine1_idx` (`Ordine_ID_Ordine` ASC),
  INDEX `fk_Prodotto_has_Ordine_Prodotto1_idx` (`Prodotto_ID_prod` ASC),
  CONSTRAINT `fk_Prodotto_has_Ordine_Prodotto1`
    FOREIGN KEY (`Prodotto_ID_prod`)
    REFERENCES `mydb`.`Prodotto` (`ID_prod`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Prodotto_has_Ordine_Ordine1`
    FOREIGN KEY (`Ordine_ID_Ordine`)
    REFERENCES `mydb`.`Ordine` (`ID_Ordine`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Lista_Prodotti_Riparazione`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Lista_Prodotti_Riparazione` (
  `Prodotto_ID_prod` INT NOT NULL,
  `Modulo_Riparazione_ID_Riparazione` INT NOT NULL,
  PRIMARY KEY (`Prodotto_ID_prod`, `Modulo_Riparazione_ID_Riparazione`),
  INDEX `fk_Prodotto_has_Modulo_Riparazione_Modulo_Riparazione1_idx`
(`Modulo_Riparazione_ID_Riparazione` ASC),
  INDEX `fk_Prodotto_has_Modulo_Riparazione_Prodotto1_idx` (`Prodotto_ID_prod` ASC),
  CONSTRAINT `fk_Prodotto_has_Modulo_Riparazione_Prodotto1`
    FOREIGN KEY (`Prodotto_ID_prod`)
    REFERENCES `mydb`.`Prodotto` (`ID_prod`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Prodotto_has_Modulo_Riparazione_Modulo_Riparazione1`
    FOREIGN KEY (`Modulo_Riparazione_ID_Riparazione`)
    REFERENCES `mydb`.`Modulo_Riparazione` (`ID_Riparazione`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Lista_Prodotti_Vendita`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Lista_Prodotti_Vendita` (
  `Modulo_Vendita_ID_Vendita` INT NOT NULL,
  `Prodotto_ID_prod` INT NOT NULL,
  `Quantità` INT NOT NULL,
  PRIMARY KEY (`Modulo_Vendita_ID_Vendita`, `Prodotto_ID_prod`),
  INDEX `fk_Modulo_Vendita_has_Prodotto_Prodotto1_idx` (`Prodotto_ID_prod` ASC),
  INDEX `fk_Modulo_Vendita_has_Prodotto_Modulo_Vendita1_idx` (`Modulo_Vendita_ID_Vendita`
ASC),
  CONSTRAINT `fk_Modulo_Vendita_has_Prodotto_Modulo_Vendita1`
    FOREIGN KEY (`Modulo_Vendita_ID_Vendita`)
    REFERENCES `mydb`.`Modulo_Vendita` (`ID_Vendita`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Modulo_Vendita_has_Prodotto_Prodotto1`
    FOREIGN KEY (`Prodotto_ID_prod`)
    REFERENCES `mydb`.`Prodotto` (`ID_prod`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

5.3 Formulazione delle interrogazioni

```
/*1- Registrazione di un nuovo Cliente*/
INSERT INTO Cliente VALUES
('BNCGCM94E22H531I', 'Giacomo', 'Bianchi', 3492467399, 'Roma');


/*2- Modifica dati di un Cliente*/
UPDATE Cliente
SET Indirizzo = 'Milano'
WHERE Nome ='Giacomo' AND Cognome = 'Bianchi';


/*3- Creazione di un Modulo Riparazione*/
INSERT INTO Modulo_Riparazione (ID_Riparazione, Difetto, Data_E, Cliente_CF, Stato) VALUES
(30600, 'controllo', '2015-06-12', 'BNCGCM94E22H531I', 'Repair');


/*4- Creazione di un Modulo Vendita*/
INSERT INTO Modulo_Vendita (ID_Vendita, Data, Cliente_CF) VALUES
(20500, '2015-06-11', 'BNCGCM94E22H531I');


/*5- Creazione di un Modulo Ordine*/
INSERT INTO Ordine (ID_Ordine, Data, Dipendente_CF) VALUES
(40400, '2015-06-11', 'RSISMN94E12H501I');


/*6- Modifica Indice Riparazioni*/
INSERT INTO Lista_Riparazioni (Dipendente_CF, Modulo_Riparazione_ID_Riparazione) VALUES
('BNCCRL94E12H501I', 30600);


/*7- Modifica Indice Vendite*/
INSERT INTO Lista_Vendite (Modulo_Vendita_ID_Vendita, Dipendente_CF) VALUES
(20500, 'RSSMRI94E12H501I');


/*8- Modifica Indice Ordine*/
INSERT INTO Lista_Ordine (Prodotto_ID_prod, Ordine_ID_Ordine, Quantità) VALUES
(10300, 40400, 2),
(10200, 40400, 1);


/*-9 Modifica Indice lista prodotti vendita*/
INSERT INTO Lista_Prodotti_Vendita (Modulo_Vendita_ID_Vendita, Prodotto_ID_prod, Quantità)
VALUES
(20500, 10300, 1),
(20500, 10200, 1);


/*10- Modifica Indice lista Prodotti in riparazione*/
INSERT INTO Lista_Prodotti_Riparazione (Prodotto_ID_prod,
Modulo_Riparazione_ID_Riparazione) VALUES
(10200, 30600);
```

```
/*11- Registrazione di un nuovo Dipendente*/
INSERT INTO Dipendente VALUES
('ABBFRN94E12H501I', 'Franco', 'Abbatecola', '2015-06-15', 'Vendita');


/*12- Modifica Indice Dipendenti*/
INSERT INTO Lista_Dipendenti VALUES
('ABBFRN94E12H501I', 1000);


/*13- Registrazione di un nuovo Prodotto*/
INSERT INTO Prodotto (ID_Prod, Marca, Modello, Categoria, Costo, Prezzo) VALUES
(10800, 'LG', 'PLEX2', 'Smartphone', 200.00, 399.99);


/*14- Visualizzazione delle riparazioni effettuate da un determinato Tecnico*/
SELECT T1.Modulo_Riparazione_ID_Riparazione, T2.Nome, T2.Cognome
FROM Lista_Riparazioni T1, Dipendente T2
WHERE T1.Dipendente_CF=T2.CF AND T2.Nome='Carlo' AND T2.Cognome='Bianchi';


/*15- Visualizza le vendite effettuate da un determinato Commesso*/
SELECT T1.Modulo_Vendita_ID_Vendita, T2.Nome, T2.Cognome
FROM Lista_Vendite T1, Dipendente T2
WHERE T1.Dipendente_CF = T2.CF AND T2.Nome='Mario' AND T2.Cognome='Rossi';


/*16-Visualizzazione degli Ordini effettuati da un determinato Gestore */
SELECT T1.ID_Ordine, T1.Data, T2.Nome, T2.Cognome
FROM Ordine T1, Dipendente T2
WHERE T1.Dipendente_CF = T2.CF AND T2.Nome='Franco' AND T2.Cognome='Verdi';


/*17- Visualizzazione della quantita disponibile di un prodotto in un determinato  P.Vendita*/
SELECT T1.Marca, T1.Modello, T2.Quantità, T3.Indirizzo
FROM Prodotto T1, Magazzino T2, Punto_Vendita T3
WHERE T1.ID_Prod=T2.Prodotto_ID_prod AND T2.Punto_Vendita_ID_Store=T3.ID_Store AND
T1.Marca='Apple' AND T1.Modello='iPhone4' AND T3.Indirizzo='Milano, 03343, 32';


/*18- Visualizzazione degli acquisti effettuati da un determinato cliente*/
SELECT T1.Nome, T1.Cognome,T2.ID_Vendita, T2.Data, T2.Prezzo
FROM Cliente T1, Modulo_Vendita T2
WHERE T1.CF=T2.Cliente_CF AND T1.Nome='Mario' AND T1.Cognome='Neri';


/*19-Visualizzazione delle riparazioni richieste da un determinato cliente*/
SELECT T1.Nome, T1.Cognome, T2.ID_Riparazione, T2.Data_E, T2.Difetto
FROM Cliente T1, Modulo_Riparazione T2
WHERE T1.CF=T2.Cliente_CF AND T1.Nome='Carlo' AND T1.Cognome='Rossi';
```

```
/*20-Visualizzazione dei Prodotti con quantià Ridotta*/
SELECT T1.Marca, T1.Modello, T2.Quantità
FROM Magazzino T2, Prodotto T1
WHERE T1.ID_Prod=T2.PRodotto_ID_Prod AND T2.Quantità<3;


/*21- Visualizzazione di una lista di prodotti appartenenti ad una marca specificata e quantità nei
rispettivi store*/
SELECT T1.Marca, T1.Modello, T2.Quantità, T3.Indirizzo
FROM Prodotto T1, Magazzino T2, Punto_Vendita T3
WHERE T1.ID_Prod=T2.Prodotto_ID_prod AND T2.Punto_Vendita_ID_Store=T3.ID_Store AND
T1.Marca='Samsung';


/*22- Visualizzazione di una lista di prodotti appartenenti ad una marca specifica e con un prezzo
pari a €400*/
SELECT Marca, Modello, Prezzo
FROM Prodotto
WHERE Marca='Samsung' AND Prezzo<480;


/*23- Modifica del quantitativo*/
UPDATE Magazzino
SET Quantità =+ 2
WHERE Prodotto_ID_prod=10400 AND Punto_Vendita_ID_Store='2000';


/*24- Visualizzazione dello status di una riparazione di un cliente specificato*/
SELECT T1.Nome, T1.Cognome, T2.ID_Riparazione, T2.Data_E, T2.Stato
FROM Cliente T1, Modulo_Riparazione T2
WHERE T1.CF=T2.Cliente_CF AND T1.Nome='Giacomo' AND T1.Cognome='Bianchi';


/*25- Visualizza una lista di riparazioni con status "Repair" */
SELECT ID_Riparazione, Difetto, Data_E, Stato
FROM Modulo_Riparazione
WHERE Stato='Repair';


/*26- Visualizzazione di una lista di riparazioni “Completo"*/
SELECT ID_Riparazione, Difetto, Data_E, Stato
FROM Modulo_Riparazione
WHERE Stato='Completo';


/*27- Visualizzazione di una lista di riparazioni “Consegnato"*/
SELECT ID_Riparazione, Difetto, Data_E, Stato
FROM Modulo_Riparazione
WHERE Stato='Consegnato';


/*28- Modifica dello Status di una riparazione*/
UPDATE Modulo_Riparazione
SET Stato='Completo'
WHERE ID_Riparazione=30200;
```

5.4 Formulazione delle Viste

/*29- Vista di Fatture delle vendite */
CREATE VIEW Fattura_Vendita_VIEW (Nome, Cognome, Indirizzo, Telefono, CF, Data, Prezzo, ID_Vendita)
AS SELECT T1.Nome, T1.Cognome, T1.Indirizzo, T1.Telefono, T1.CF, T2.Data, T2.Prezzo, T2.ID_Vendita
FROM Cliente T1, Modulo_Vendita T2
WHERE T1.CF=T2.Cliente_CF;

/*30- Vista di fatture rispettive alle riparazioni */
CREATE VIEW Fattura_Riparazione_VIEW (Nome, Cognome, Indirizzo, Telefono, CF, Data, Difetto, Prezzo, ID_Riparazione)
AS SELECT T1.Nome, T1.Cognome, T1.Indirizzo, T1.Telefono, T1.CF, T2.Data_U, T2.Difetto, T2.Prezzo, T2.ID_Riparazione
FROM Cliente T1, Modulo_Riparazione T2
WHERE T1.CF=T2.Cliente_CF AND T2.Stato=Consegnato;

/*31- Vista di fatture rispettive agli ordini*/
CREATE VIEW Fattura_Ordine_VIEW (Nome, Cognome, Indirizzo, Data, ID_Ordine)
AS SELECT T1.Nome, T1.Cognome, T2.Indirizzo, T3.Data, T3.ID_Ordine
FROM Dipendente T1, Ordine T3, Punto_Vendita T2, Lista_Dipendenti T4
WHERE T4.Dipendente_CF=T1.CF AND T4.Punto_Vendita_ID_Store=T2.ID_Store AND T1.CF=T3.Dipendente_CF;

5.5 Formulazione dei Trigger

- Viene riportata la formulazione di un trigger di esempio anche se durante la progettazione di è scelto di usare un automazione tramite software e non tramite database per questioni di funzionamento non corretto.

/*TRIGGER 1- Imposta il prezzo della nuova tupla in modulo vendita calcolando la sommatoria della moltiplicazione tra il prezzo e la rispettiva quantità di ogni singolo prodotto */

```
CREATE TRIGGER in_Prezzo
AFTER INSERT ON Modulo_Vendita
FOR EACH ROW
UPDATE Modulo_Vendita
SET NEW.Prezzo=(SELECT SUM(TOT)
                FROM(SELECT ToT
                     FROM(SELECT Modulo_Vendita_ID_Vendita, Prezzo * Quantità AS TOT
                          FROM (SELECT T1.Modulo_Vendita_ID_Vendita, T1.Prodotto_ID_prod, T1.Quantità, T2.Prezzo
                                FROM Lista_Prodotti_Vendita T1, Prodotto T2
                                WHERE T1.Prodotto_ID_prod=T2.ID_prod)AS Prezzo_Quantità)AS TOT
                          WHERE Modulo_Vendita_ID_Vendita=(SELECT MAX(Modulo_Vendita_ID_Vendita)
                                                           FROM Modulo_Vendita))AS ToT);
```

5.6 Formulazione dell’inserimento di prova

- Durante i test sono stati utilizzati dati di esempio, di seguito le istruzioni di inserimento utilizzati in queste fasi:

```
insert into Prodotto values
(10100, 'Apple', 'iPhone4', 'Smartphone', 199.90, 399.95),
(10200, 'Apple', 'iPhone5', 'Smartphone', 250.00, 449.98),
(10300, 'Apple', 'iPhone6', 'Smartphone', 300.00, 499.99),
(10400, 'Samsung', 'Galaxy4', 'Smartphone', 200.00, 399.96),
(10500, 'Samsung', 'Galaxy5', 'Smartphone', 250.00, 449.50),
(10600, 'Samsung', 'Galaxy6', 'Smartphone', 300.00, 499.60),
(10700, 'Samsung', 'GalaxyNote', 'Smartphone', 350.00, 549.99);
```

```
insert into Punto_Vendita values
(1000, 'Roma, 03043, 53', 1776300178),
(2000, 'Milano, 03343, 32', 1774365385),
(3000, 'Torino, 50211, 113', 1637536423);
```

```
insert into Dipendente values
('RSSMRI94E12H501I', 'Mario', 'Rossi', '2010-04-01', 'Vendita'),
('BNCCRL94E12H501I', 'Carlo', 'Bianchi', '2011-05-01', 'Riparazion'),
('VRDFRN94E12H501I', 'Franco', 'Verdi', '2012-06-01', 'Gestione'),
('NRILCA94E12H501I', 'Luca', 'Neri', '2012-07-02', 'Vendita'),
('RSIPLO94E12H501I', 'Paolo', 'Rossi', '2013-04-12', 'Riparazion'),
('BNCMRI94E12H501I', 'Mario', 'Bianchi', '2012-06-13', 'Gestione'),
('VRDBRN94E12H501I', 'Bruno', 'Verdi', '2013-06-12', 'Vendita'),
('NRIGVN94E12H501I', 'Giovanni', 'Neri', '2014-12-03', 'Riparazion'),
('RSISMN94E12H501I', 'Simone', 'Rossi', '2014-11-04', 'Gestione');
```

```
insert into Lista_Dipendenti values
('RSSMRI94E12H501I', 1000),
('BNCCRL94E12H501I', 1000),
('VRDFRN94E12H501I', 1000),
('NRILCA94E12H501I', 2000),
('RSIPLO94E12H501I', 2000),
('BNCMRI94E12H501I', 2000),
('VRDBRN94E12H501I', 3000),
('NRIGVN94E12H501I', 3000),
('RSISMN94E12H501I', 3000);
```

```
insert into Magazzino values
(10200, 1000, 35),
(10300, 1000, 62),
(10400, 1000, 14),
(10500, 1000, 26),
(10600, 1000, 23),
(10700, 1000, 72),
(10100, 2000, 2),
(10200, 2000, 25),
(10300, 2000, 27),
(10400, 2000, 28),
(10500, 2000, 84),
(10600, 2000, 26),
(10700, 2000, 16),
(10100, 3000, 84),
(10200, 3000, 27),
(10300, 3000, 95),
(10400, 3000, 27),
(10500, 3000, 27),
(10600, 3000, 58),
(10700, 3000, 37);
```

```
insert into Cliente values
('RSSCRL94E22H531I', 'Carlo', 'Rossi', 3492467391, 'Roma'),
('NRIMRI94E22H531I', 'Mario', 'Neri', 3492467392, 'Roma'),
('VRIMCE94E22H531I', 'Michele', 'Verdi', 3492467393, 'Milano'),
('BNCLCA94E22H531I', 'Luca', 'Bianchi', 3492467394, 'Milano');
```

```
insert into Modulo_Vendita values
(20100, '2015-01-01', 850.00, 'RSSCRL94E22H531I'),
(20200, '2015-01-02', 450.00, 'VRIMCE94E22H531I'),
(20300, '2015-01-03', 450.00, 'BNCLCA94E22H531I'),
(20400, '2015-01-04', 900.00, 'NRIMRI94E22H531I');
```

```
insert into Lista_Vendite values
(20100, 'RSSMRI94E12H501I'),
(20200, 'NRILCA94E12H501I'),
(20300, 'VRDBRN94E12H501I'),
(20400, 'NRILCA94E12H501I');
```

```
insert into Modulo_Riparazione values
(30100, 'Garanzia', '2015-01-01', '2015-02-01', 'NRIMRI94E22H531I', 0.00, 'Completo'),
(30200, 'Batteria', '2015-01-02', '2015-02-02', 'VRIMCE94E22H531I', 10.00, 'Repair'),
(30300, 'Display', '2015-01-03', '2015-02-03', 'RSSCRL94E22H531I', 40.00, 'Completo'),
(30400, 'Microfono', '2015-01-04', '2015-02-04', 'BNCLCA94E22H531I', 5.00, 'Consegnato'),
(30500, 'Fusibile', '2015-01-05', '2015-02-05', 'RSSCRL94E22H531I', 2.00, 'Completo');



insert into Lista_Riparazioni values
('BNCCRL94E12H501I', 30100),
('BNCCRL94E12H501I', 30200),
('RSIPLO94E12H501I', 30300),
('NRIGVN94E12H501I', 30400),
('RSIPLO94E12H501I', 30500);



insert into Ordine values
(40100, '2015-05-01', 'VRDFRN94E12H501I'),
(40200, '2015-05-14', 'BNCMRI94E12H501I'),
(40300, '2015-06-01', 'RSISMN94E12H501I');



insert into Lista_Ordine values
(10200, 40100, 5),
(10300, 40100, 2),
(10400, 40100, 1),
(10300, 40200, 2),
(10500, 40300, 3);



insert into Lista_Prodotti_Vendita values
(20100, 10100, 1),
(20100, 10200, 1),
(20200, 10500, 1),
(20300, 10500, 1),
(20400, 10500, 2);



insert into Lista_Prodotti_Riparazione values
(10200, 30100),
(10100, 30200),
(10500, 30300),
(10500, 30400),
(10200, 30500);
```

6 Allegati

6.1.1 File SQL

Nella cartella “SQL” sono presenti i file .sql per:

- Creazione del database Create.sql
- Operazioni Operations_1_2.sql
- Creazione Viste Views_1_2.sql
- Creazione Trigger Trigger_1_1.sql
- un’ inserimento di esempio Insert_1_2.sql

6.1.2 File TXT

Nella cartella “TXT” sono presenti i file SQL convertiti in .txt

6.1.3 File EER

Nella cartella principale del Progetto è presente il file del modello EER

- Progetto_Esame_eer_1_2.mwb
- EER.pdf per una visualizzazione rapida dello schema

6.1.4 Traduzione ORACLE SQL

Nella cartella “ORACLE” sono presenti i vari file della traduzione da MySQL a Oracle SQL

6.1.5 File Relazione

La relazione è stata elaborata tramite Pages (Mac OSX).
Per questioni di compatibilità la relazione è stata esportata in .pdf
“Progetto_BdD2015E_1_2Final.pdf”

6.1.6 Illustrazioni

Le immagini allegate nella relazione sono state elaborate tramite https://www.draw.io, un software gratuito utilizzabile direttamente nel browser.
Tutti i file degli schemi e delle varie rappresentazioni sono presenti nella cartella “Immagini_schemi”

7 Ruoli

7.1 Daniele La Rosa

- Elaborazione in gruppo della traccia
- Elaborazione in gruppo dello schema logico
- Elaborazione in gruppo delle operazioni (query.sql)
- Controllo sulla correttezza della relazione e del codice MySQL
- Traduzione di MySQL in Oracle SQL

7.2 Maria Zonfrilli

- Elaborazione delle operazioni (query.sql)
- Controllo sulla correttezza della relazione e delle illustrazioni
- Collaborazione parziale nella progettazione dei Trigger

7.3 Ernesto Di Mambro

- Elaborazione dello schema EER in workbench
- Controllo della correttezza del codice MySQL
- Collaborazione parziale nella progettazione delle operazioni (Operations.sql)

7.4 Marco Pontone

- Elaborazione in gruppo della traccia
- Elaborazione in gruppo degli schemi rappresentativi
- Elaborazione dei trigger (trigger.sql)
- Analisi delle prestazioni del Database
- Controllo della correttezza dei vari codici MySQL (Operations.sql, Views.sql, Insert.sql)
- Controllo generale sulla correttezza e della funzionalità del progetto