

Relazione per il corso di Basi di Dati

A. A. 2020/2021

Progetto di basi di dati per la gestione di un negozio fisico

Studente: Crisante Davide (davide.crisante3@studio.unibo.it)

Matricola: 830155

INDICE	PAGINA
Capitolo 1 - Analisi	3
1.1 Analisi dei requisiti	3
1.2 Progettazione delle viste	5
Capitolo 2 - Progettazione concettuale	6
2.1 Schema Responsabile	6
2.2 Schema Dipendente	7
2.3 Schema Vendita	7
2.4 Schema Prodotto	8
2.5 Schema concettuale finale	9
Capitolo 3 - Progettazione logica	10
3.1 Stima del volume dei dati	10
3.2 Stima della frequenza delle operazioni principali	12
3.3 Schemi di navigazione e tabelle degli accessi	13
3.4 Raffinamento dello schema	17
3.5 Analisi delle ridondanze	19
3.6 Traduzione di entità e associazioni in relazioni	21
3.7 Schema relazionale finale	23
3.8 Costruzione delle tabelle del DB in linguaggio SQL	24
3.9 Traduzione delle operazioni in query SQL	29
Capitolo 4 - Progettazione dell'applicazione	36
4.1 Descrizione dell'architettura dell'applicazione realizzata	36

Capitolo 1 - Analisi

1.1 Analisi dei requisiti

Si vuole sviluppare un programma gestionale che consenta di gestire i dati: della cassa, del personale, dei clienti tesserati, dei prodotti, delle videocamere di sorveglianza, degli interessi dei clienti, delle entrate/uscite dell'attività.

Specifiche in linguaggio naturale

Il proprietario di un negozio fisico intende modernizzare il proprio approccio alle strategie di mercato e alla gestione dell'inventario e dei dipendenti.

Il negozio colleziona i dati di **dipendenti, stipendi, vendite, ricerche effettuate dagli utenti, video di sorveglianza, fornitori, clienti tesserati**.

I dipendenti non hanno dei turni fissi di lavoro quindi devono segnalare quando il turno inizia e finisce tramite il programma.

I dipendenti:

- devono:
 - loggarsi sul programma per indicare l'inizio del loro turno lavorativo;
- possono:
 - inizializzare il valore del fondo cassa;
 - effettuare ricerche di prodotti per nome o per categoria;
 - effettuare vendite;
 - cercare dettagli di vendite effettuate;
 - registrare nuovi clienti tesserati;
 - cercare clienti tesserati;

I responsabili hanno le stesse possibilità dei normali dipendenti ma in più possono:

- fare modifiche al personale
- visualizzare e rimuovere video di sorveglianza
- visualizzare le statistiche di ricerca/vendita
- inserire eventuali spese
- fare ordini a fornitori;
- registrare nuovi prodotti nel DB;

Di entrambe le categorie vogliamo memorizzare:

- nome
- cognome
- email
- città di residenza

Gli stipendi vengono erogati di norma il primo giorno di ogni mese e si vuole che sia lasciata la libertà ai responsabili di aumentare o diminuire lo stipendio rispetto allo stipendio base per ricompensare o meno un dipendente, i responsabili non possono modificare lo stipendio di loro stessi e degli altri responsabili.

Per ridurre la mole di lavoro del personale, il proprietario, ha deciso di lasciare un computer a disposizione della clientela, cosicché quest'ultima potesse controllare autonomamente prezzi e quantità in negozio dei prodotti.

I clienti possono:

- filtrare il catalogo per categoria di prodotto;
- cercare tramite delle parole chiave.

Le ricerche effettuate dai clienti sul computer apposito vengono salvate per fini statistici, utili per elaborare possibili strategie di marketing.

Delle vendite vanno immagazzinati:

- prodotti acquistati
- data di acquisto
- venditore
- numero di tessera del cliente, in caso di cliente tesserato
- codice scontrino

Inoltre vanno memorizzate anche informazioni riguardanti gli acquisti della merce dai fornitori:

- nome fornitore
- articolo acquistato
- quantità
- costo unitario dell'articolo
- data acquisto

I video di sorveglianza vengono già gestiti da videocamere connesse tramite una connessione LAN al computer dedicato a server per il nostro Sistema Informativo, non è necessario realizzare un applicativo apposito per esse in quanto già esistente, andrà solo adattato al nuovo SI.

I dipendenti ed i responsabili ricevono uno stipendio di cui vogliamo memorizzare importo, data, beneficiario ed assegnatore.

Come benefit tutti i dipendenti, presenti e passati, hanno anch'essi una tessera cliente per usufruire di eventuali benefici dedicati ai clienti tesserati, quindi sono a tutti gli effetti clienti tesserati.

Dei clienti tesserati dobbiamo memorizzare:

- nome
- cognome
- città di residenza
- email
- numero tessera

I dati personali di clienti tesserati e dipendenti possono essere mantenuti nel DB anche se un cliente/dipendente viene eliminato/licenziato.

Glossario dei Termini:

Termine	Descrizione	Sinonimi	Collegamenti
Dipendente	Persona che lavora nell'azienda. Può effettuare delle vendite.		Responsabile, Vendita, Turno
Responsabile	Persona che lavora nell'azienda. E' a tutti gli effetti un dipendente ma con più responsabilità.	Dirigente	Vendita, Dipendente, Turno
Turno	Orario in cui un dipendente sta lavorando	Orario di lavoro	Dipendente, Responsabile, Vendita

1.2 Progettazione delle Viste

Sono da considerare tre fasce di utenza del DB:

- Responsabili
- Dipendenti
- Utenti

Azioni necessarie per Responsabili:

1. Segnare inizio e fine del proprio turno lavorativo
2. Inizializzare il fondo cassa
3. Effettuare ricerche di prodotti per nome/categoria
4. Effettuare vendite
5. Visualizzazione dello stato di attività dei Dipendenti
6. Visualizzare e rimuovere Video di sorveglianza
7. Visualizzare le statistiche ricavate dai dati di ricerche e vendite
8. Visualizzare i profitti
9. Inserire costi (affitto, ecc)
10. Inserire l'importo dello stipendio per ogni dipendente
11. Modificare i dati riguardanti il personale (dati personali o aggiungere/rimuovere dipendenti)
12. Effettuare acquisti dai fornitori
13. Aggiungere, modificare o rimuovere informazioni riguardanti i Clienti Tesserati

Azioni necessarie per Dipendenti:

1. Segnalare inizio e fine del proprio turno lavorativo
2. Inizializzare il fondo cassa
3. Effettuare ricerche di prodotti per nome/categoria
4. Effettuare vendite
5. Aggiungere, modificare o rimuovere informazioni riguardanti i Clienti Tesserati

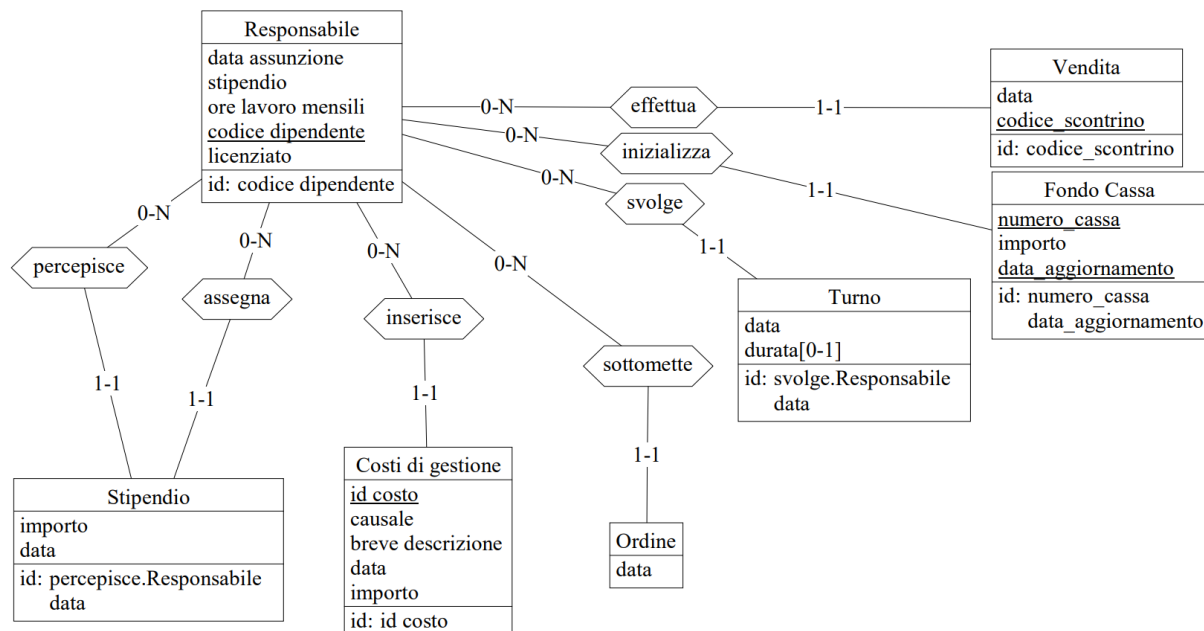
Azioni necessarie per Utenti presso il computer pubblico del negozio:

1. effettuare ricerche di prodotti disponibili in magazzino per nome/categoria

Capitolo 2 - Progettazione concettuale

2.1 Schema Responsabile

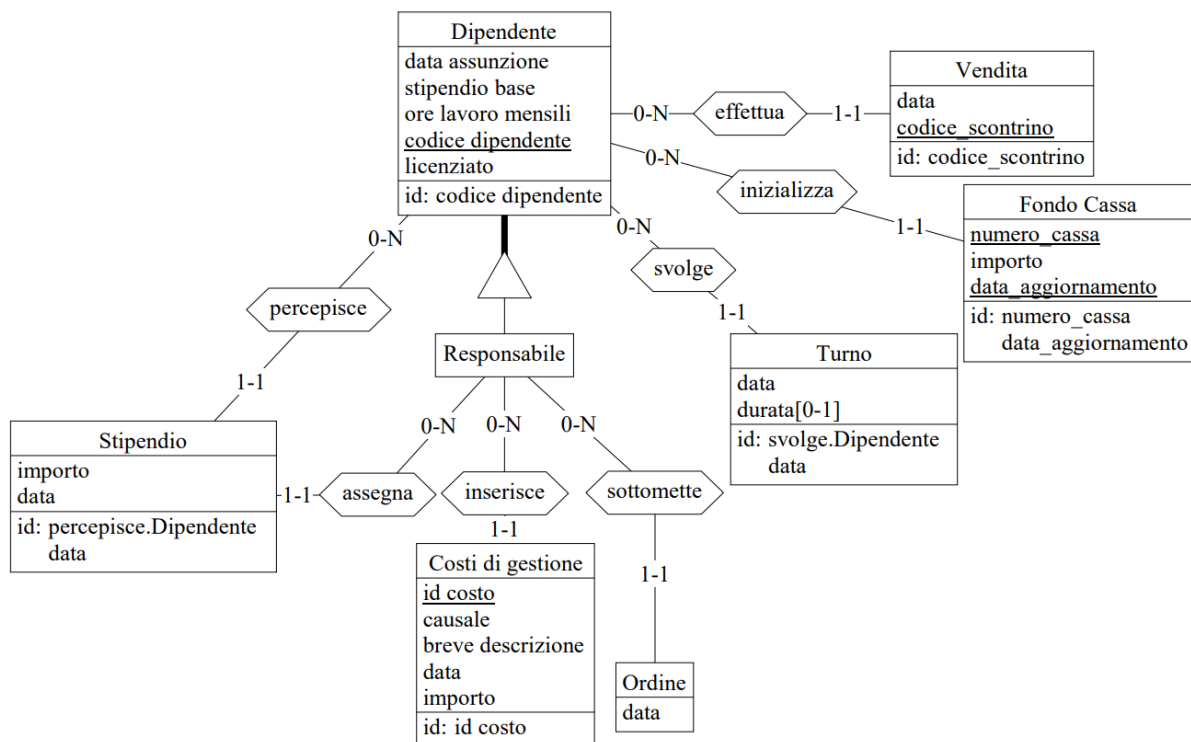
2.1.1 schema scheletro



2.1.2 raffinamenti proposti

Responsabile è può essere rappresentato come sottoclasse di Dipendente

2.1.3 schema concettuale parziale

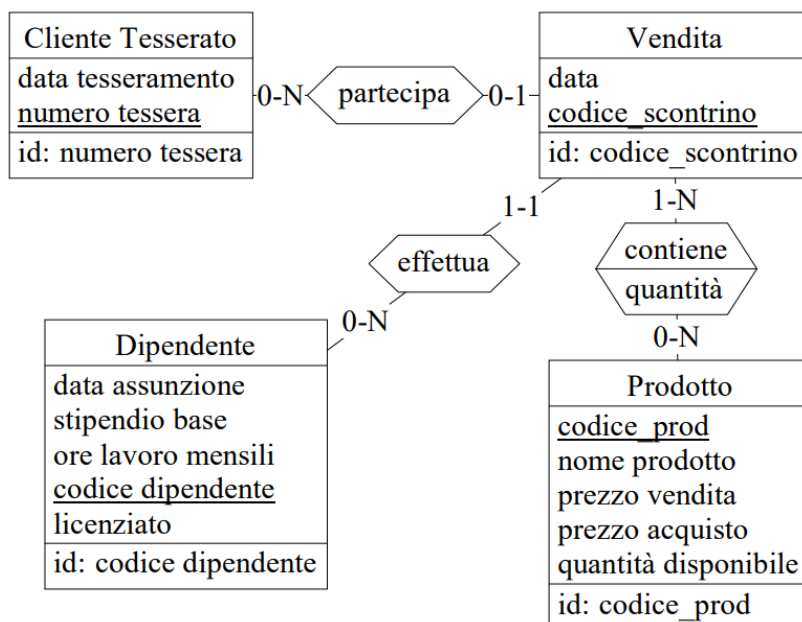


2.2 Schema Dipendente

Descritto implicitamente in 2.1.3 .

2.3 Schema Vendita

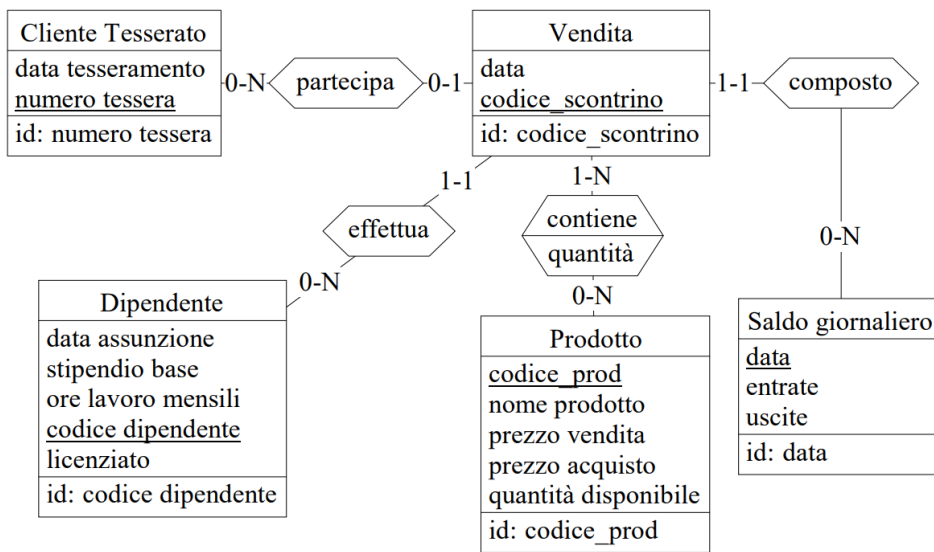
2.3.1 schema scheletro



2.3.2 raffinamenti proposti

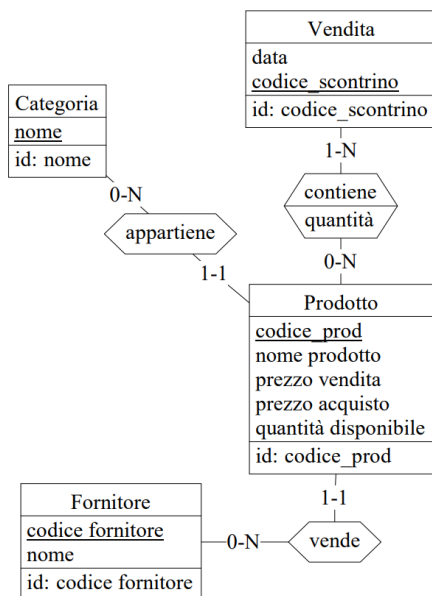
In vista delle operazioni richieste riguardanti il bilancio economico viene scelto di introdurre l'entità "Saldo giornaliero" questa scelta verrà analizzata in seguito.

2.3.3 schema concettuale parziale



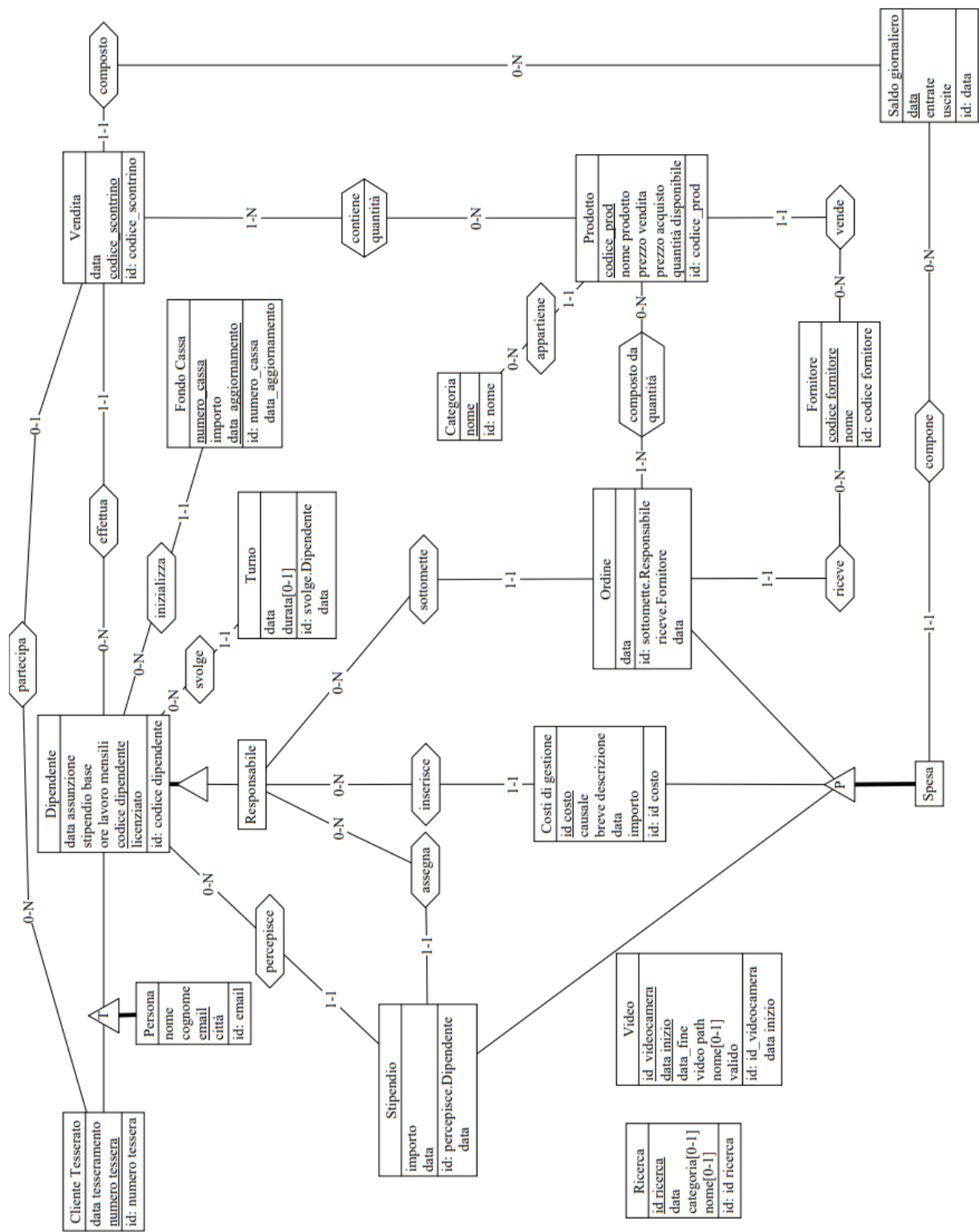
2.4 Schema Prodotto

2.4.1 schema scheletro



2.5 Schema concettuale finale

La pagina seguente è riservata allo schema concettuale finale:



Capitolo 3 - Progettazione logica

3.1 Stima del volume dei dati

Concetto	Tipo (Entity / Relationship)	Volume
Persona	E	550
Cliente Tesserato	E	500
partecipa	R	1000
Dipendente	E	20
Responsabile	E	3
percepisce	R	250
Stipendio	E	250
assegna	R	250
inserisce	R	300
Costi di gestione	E	300
sottomette	R	40
Ordine	E	40
svolge	R	900
Turno	E	900
effettua	R	35.000
Vendita	E	35.000
inizializza	R	300
Fondo Cassa	E	300
contiene	R	140.000
Prodotto	E	200
appartiene	R	200
Categoria	E	10

composto da	R	800
riceve	R	40
vende	R	40
Fornitore	E	5
Spesa	E	590
compone	R	360
Saldo giornaliero	E	360
composto	E	35.000
Ricerca	E	14.400
Video	E	1000

3.2 Stima della frequenza delle operazioni principali

Tabella delle operazioni:

Codice	Operazione	Frequenza media
R1	Visualizzazione dello stato di attività dei <u>Dipendenti</u>	2 / giorno
R2	Visualizzare le <u>ricerche</u> svolte	2 / mese
R3	Totale vendite giornaliere	1 / giorno
R4	Totale vendite settimanali	4 / mese
R5	Totale vendite mensili	2 / mese
R6	Saldo mensile (guadagni mese - spese mese)	5 / mese
R7	Saldo annuale (guadagni anno - spese anno)	5 / anno
R8	Inserire costi di gestione (affitto, bollette, assicurazioni, ecc)	5 / mese
R9	Inserire l'importo dello stipendio mensile per un dipendente	12 / mese * 20 dipendenti
R10	Modificare i dati riguardanti il personale	2 / anno
R11	Aggiungere personale	5 / anno
R12	Sottomettere ordini ai fornitori	4 / mese * 5 fornitori
R13	Visualizzare <u>Video</u> di sorveglianza	5 / mese
R14	Rimuovere <u>Video</u> di sorveglianza	5 / anno
R15	Eliminare personale	10 / anno
D1	Ricerca vendite effettuate	5/mese
D2	Registrare <u>Clienti Tesserati</u>	50 / mese
D3	Modificare informazioni riguardanti i <u>Clienti Tesserati</u>	5 / anno
D4	Segnare inizio e fine del proprio turno lavorativo	10 / giorno
D5	Inizializzare il fondo cassa	1 / giorno
D6	Effettuare vendite	100 / giorno

C1	Effettuare ricerche di prodotti per nome	40 / giorno
C2	Effettuare ricerche di prodotti per categoria	40 / giorno

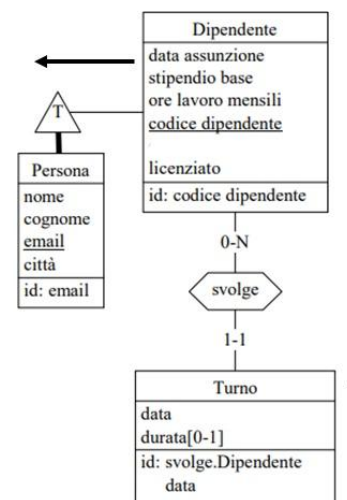
3.3 Schemi di navigazione e tabelle degli accessi

Dopo aver elencato le principali operazioni richieste associate alle rispettive frequenze, esprimeremo gli schemi di navigazione per le operazioni che non sono ritenute banali.

R1) Visualizzazione dello stato di attività dei Dipendenti

Visualizzare l'elenco dei dipendenti specificando quanto tempo hanno lavorato nell'ultimo mese e se stanno attualmente lavorando.

Si presume che un dipendente stia lavorando se nel SI è noto l'orario di inizio turno e non l'orario di fine turno.



Concetto	Costrutto	Accessi	Tipo
Dipendente	E	20	L
Persona	E	20	L
svolge	R	45	L
Turno	E	45	L

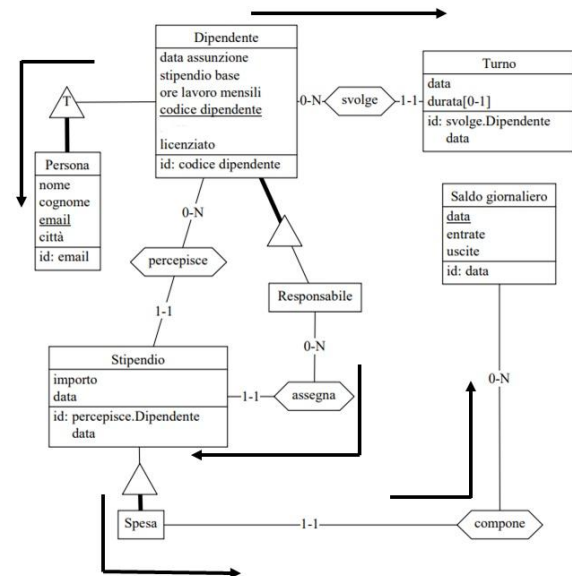
Tot : 130L Frequenza : 2/giorno

Costo totale : 260L *al giorno*

R9) Inserire l'importo dello stipendio mensile per un dipendente

Mostrare quanto il dipendente ha lavorato durante l'ultimo mese, i suoi dati personali e lavorativi.

Dopo aver immesso l'importo dello stipendio va aggiornato anche il saldo giornaliero.



Concetto	Costrutto	Accessi	Tipo
Persona	E	1	L
Dipendente	E	1	L
svolge	R	45	L
Turno	E	45	L
percepisce	R	1	S
Stipendio	E	1	S
assegna	R	1	S
Spesa	E	1	L
compone	R	1	L
Saldo giornaliero	E	1	L + S

Tot : 95L + 4s

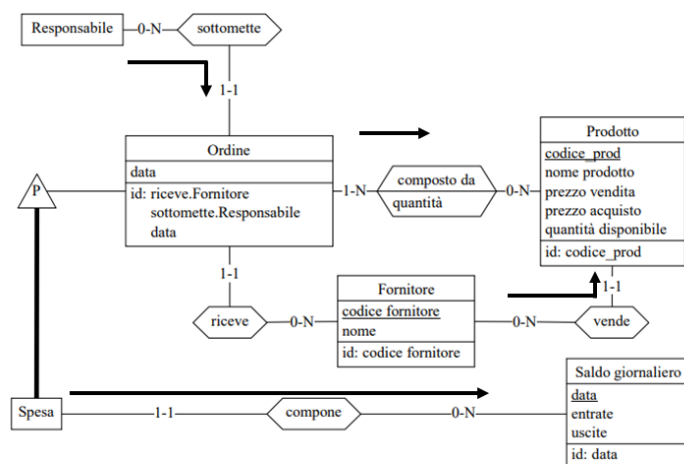
Frequenza : 12/anno * 20 dipendenti

Costo totale : 22.800L + 960S *all'anno*

R12) Sottomettere ordini ai fornitori

Salva l'ordine sottomesso ad un dato fornitore aggiornando le quantità disponibili in negozio dei prodotti acquistati dal fornitore ed aggiornando il saldo giornaliero.

Vanno prima letti tutti i fornitori.



Concetto	Costrutto	Accessi	Tipo
sottomette	R	1	S
Ordine	E	1	S
composto da	R	20	S
Prodotto	E	40	L
Prodotto	E	20	S
riceve	R	1	S
Fornitore	E	5	L
vende	R	40	L
Spesa	E	1	L
compone	R	1	L
Saldo giornaliero	E	1	L + S

Tot : 88L + 44s

Frequenza : 4/mese * 5 fornitori

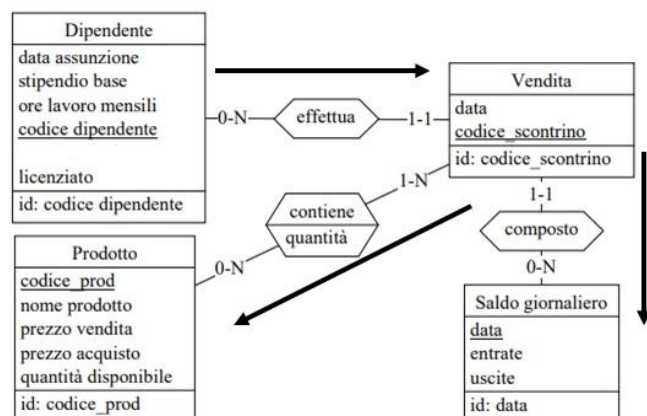
Costo totale : 1.760L + 880S *al mese*

D6) Effettuare vendite

Salva i prodotti, le quantità ed i prezzi a cui sono stati venduti i prodotti. Se il cliente è un cliente tesserato è possibile anche associare la vendita al cliente.

Vanno aggiornati:

- la quantità disponibile dei prodotti venduti
- il saldo giornaliero



Concetto	Costrutto	Accessi	Tipo
effettua	R	1	S
Vendita	E	1	S
contiene	R	4	S
Prodotto	E	4	L + S
composto	R	1	L
Saldo giornaliero	E	1	L + S

Tot : 6L + 11S

Frequenza : 100/giorno

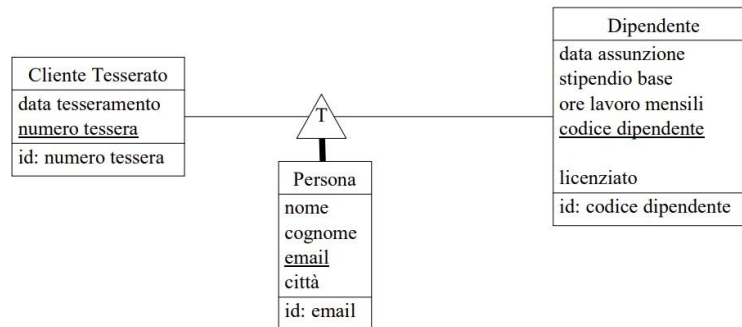
Costo totale : 600L + 1.100S *al giorno*

3.4 Raffinamento dello schema

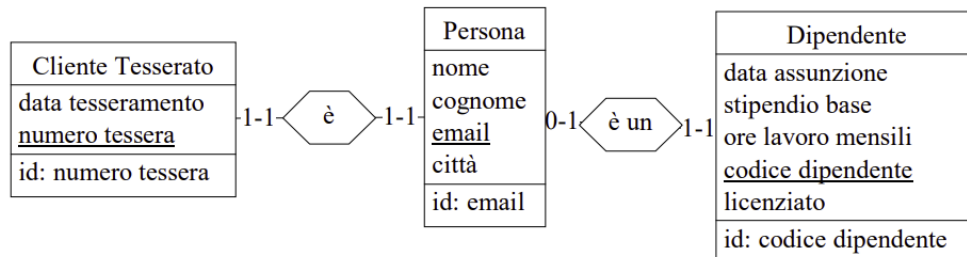
Eliminazione delle gerarchie :

Nello schema concettuale sopra presentato sono presenti tre gerarchie:

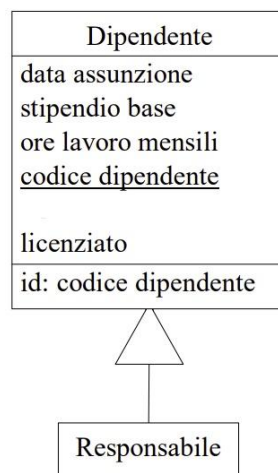
- **Persona** (Dipendente, Cliente Tesserato):



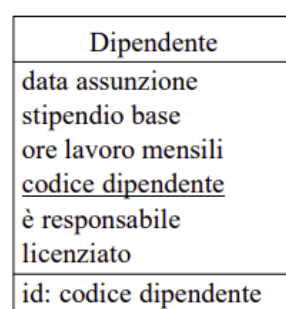
si sceglie di sostituire la generalizzazione con delle associazioni:



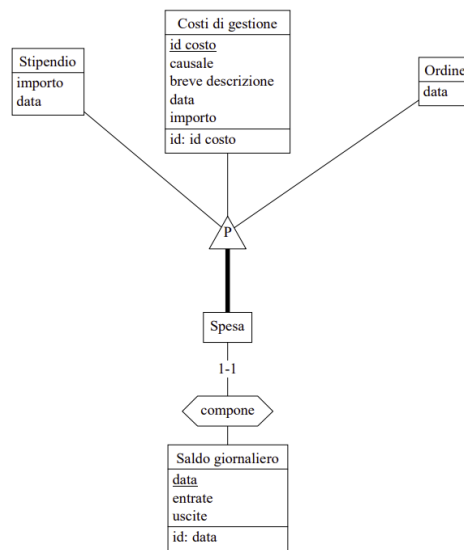
- **Responsabile** (Dipendente):



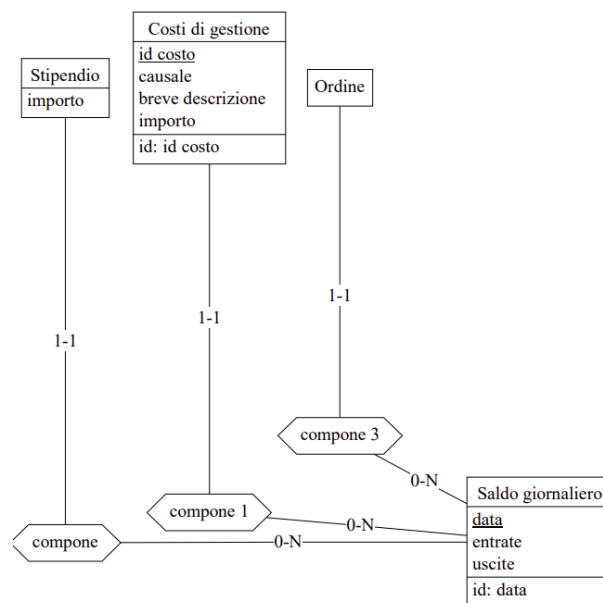
si sceglie di svolgere un collasso verso l'alto, quindi è stato aggiunto un attributo che svolge il compito di selettore:



- **Spesa** (Stipendio, Costi di gestione, Ordine):



la gerarchia è stata completamente rimossa:



3.5 Analisi delle ridondanze :

Nello schema concettuale sopra riportato è presente l'entità Saldo giornaliero che introduce la ridondanza. Lo scopo principale di questa scelta è quello di alleggerire di molto i costi di lettura per tutte le query riguardanti entrate/uscite economiche. Senza questa entità sarebbe stato necessario leggere il valore di tutte le entrate di prodotto in vendita per ogni Vendita compiuta in un dato intervallo di tempo, inoltre sarebbe stato necessario leggere ogni importo di ogni Spesa.

Il costo di scrittura che questa entità aggiunge è molto basso.

Di seguito vi è riportato come esempio il paragone tra l'esecuzione dell'operazione R7 con e senza ridondanza.

R7) Saldo annuale (guadagni anno - spese anno):

- **CON RIDONDANZA:**

Concetto	Costrutto	Accessi	Tipo
Saldo giornaliero	E	365	L

Totale : 365 L

- **SENZA RIDONDANZA:**

Concetto	Costrutto	Accessi	Tipo
Vendita	E	35.000	L
prodotto in vendita	R	140.000	L
Prodotto	E	200	L
Ordine	E	40	L
composto da	R	800	L
Costi di gestione	E	300	L
Stipendio	E	250	L

Totale: 176.590L

Eliminazione di attributi composti:

Non ci sono attributi composti.

Scelta delle chiavi primarie:

Nello schema E/R riportato sono già illustrate le chiavi primarie.

Chiavi esterne:

Per semplicità il nome delle chiavi esterne è lo stesso della chiave primaria a cui fanno riferimento tranne in cinque casi in cui è stato preferito usare un nome diverso.

- relazione “partecipa” tra Cliente Tesserato e Vendita con importazione della chiave esterna “numero_cliente_tesserato” in Vendita;
- relazione “effettua” tra Dipendente e Vendita con importazione della chiave esterna “codice_dipendente” in Vendita;
- relazione “è” tra Persona e Cliente Tesserato con importazione della chiave esterna “email” in Cliente Tesserato;
- relazione “è un” tra Persona e Dipendente con importazione della chiave esterna “email” in Dipendente;
- relazione “percepisce” tra Dipendente e Stipendio con importazione della chiave esterna “codice_beneficiario” in Stipendio;
- relazione “percepisce” tra Dipendente e Stipendio con importazione della chiave esterna “codice_assegnatore” in Stipendio;
- relazione “inserisce” tra Costi di gestione e Dipendente con importazione della chiave esterna “codice_dipendente” in Costi di gestione;
- relazione “sottomette” tra Ordine e Dipendente con importazione della chiave esterna “codice_dipendente” in Ordine;
- relazione “svolge” tra Turno e Dipendente con importazione della chiave esterna “codice_dipendente” in Turno;
- relazione “inizializza” tra Fondo Cassa e Dipendente con importazione della chiave esterna “codice_dipendente” in Fondo Cassa;
- relazione “contiene” tra Vendita e Prodotto con creazione dell'entità “Prodotto_in_vendita” che ha come attributi:
 - “codice_scontrino” importato da Vendita;
 - “codice_prod” importato da Prodotto;
 - “quantità”
 - “prezzo_quando_venduto”
- relazione “appartiene” tra Prodotto e Categoria con importazione della chiave esterna “categoria” in Prodotto;
- relazione “vende” tra Prodotto e Fornitore con creazione dell'entità “prodotto_di_fornitore” che ha come attributi:
 - “codice_prod” importato da Prodotto;
 - “codice_fornitore” importato da Fornitore;
- relazione “riceve” tra Ordine e Fornitore con importazione della chiave esterna “codice_fornitore” in Ordine;

- relazione “composto da” tra Prodotto e Ordine con creazione dell'entità “Composizione” che ha come attributi:
 - “codice_prod” importato da Prodotto;
 - “quantità”;
 - “codice_dipendente”, “giorno_saldo”, “codice_fornitore” importati da Ordine
- relazione “compone” tra Stipendio e Saldo giornaliero con importazione della chiave esterna “giorno_saldo” in Stipendio;
- relazione “compone 1” tra Costi di gestione e Saldo giornaliero con importazione della chiave esterna “giorno_saldo” in Costi di gestione;
- relazione “compone 2” tra Ordine e Saldo giornaliero con importazione della chiave esterna “giorno_saldo” in Ordine;
- relazione “compone 3” tra Vendita e Saldo giornaliero con importazione della chiave esterna “giorno_saldo” in Vendita;

3.6 Traduzione di entità e associazioni in relazioni

Cliente_Tesserato(numero_tessera, data_tesseramento, email)

FK : email REFERENCES **Persona**

Persona(email, nome, cognome, città)

Dipendente(codice_dipendente, data_assunzione, stipendio_base, ore_lavoro_mensili, email, è_responsabile, licenziato)

FK : email REFERENCES **Persona**

Fondo_Cassa(numero_cassa, data_aggiornamento, importo, codice_dipendente)

FK : codice_dipendente REFERENCES **Dipendente**

Turno(codice_dipendente, data, durata[0-1])

FK : codice_dipendente REFERENCES **Dipendente**

Vendita(codice_scontrino, giorno_saldo, codice_dipendente, numero_cliente_tesserato[0-1])

FK : giorno_saldo REFERENCES **Saldo_giornaliero**

FK : codice_dipendente REFERENCES **Dipendente**

FK : numero_cliente_tesserato REFERENCES **Cliente_Tesserato**

Prodotto_in_Vendita(codice_prod, codice_scontrino, quantità, prezzo_quando_venduto)

FK : codice_prod REFERENCES **Prodotto**

FK : codice_scontrino REFERENCES **Vendita**

Prodotto(codice_prod, nome_prodotto, prezzo_vendita, prezzo_acquisto, quantità_disponibile, categoria)

FK : categoria REFERENCES **Categoria**

Categoria(nome)

Composizione(codice_prod, codice_dipendente, giorno_saldo, codice_fornitore,
quantità)

FK : codice_prod REFERENCES **Prodotto**

FK : codice_dipendente REFERENCES **Ordine**

FK : giorno_saldo REFERENCES **Ordine**

FK : codice_fornitore REFERENCES **Ordine**

Ordine(codice_dipendente, giorno_saldo, codice_fornitore)

FK : codice_dipendente REFERENCES **Dipendente**

FK : giorno_saldo REFERENCES **Saldo_giornaliero**

FK : codice_fornitore REFERENCES **Fornitore**

prodotto_di_fornitore(codice_prod, codice_fornitore)

FK : codice_prod REFERENCES **Prodotto**

FK : codice_fornitore REFERENCES **Fornitore**

Fornitore(codice_fornitore, nome)

Saldo_giornaliero(data, entrate, uscite)

Stipendio(codice_beneficiario, giorno_saldo, importo, codice_assegnatore)

FK : codice_assegnatore REFERENCES **Dipendente**

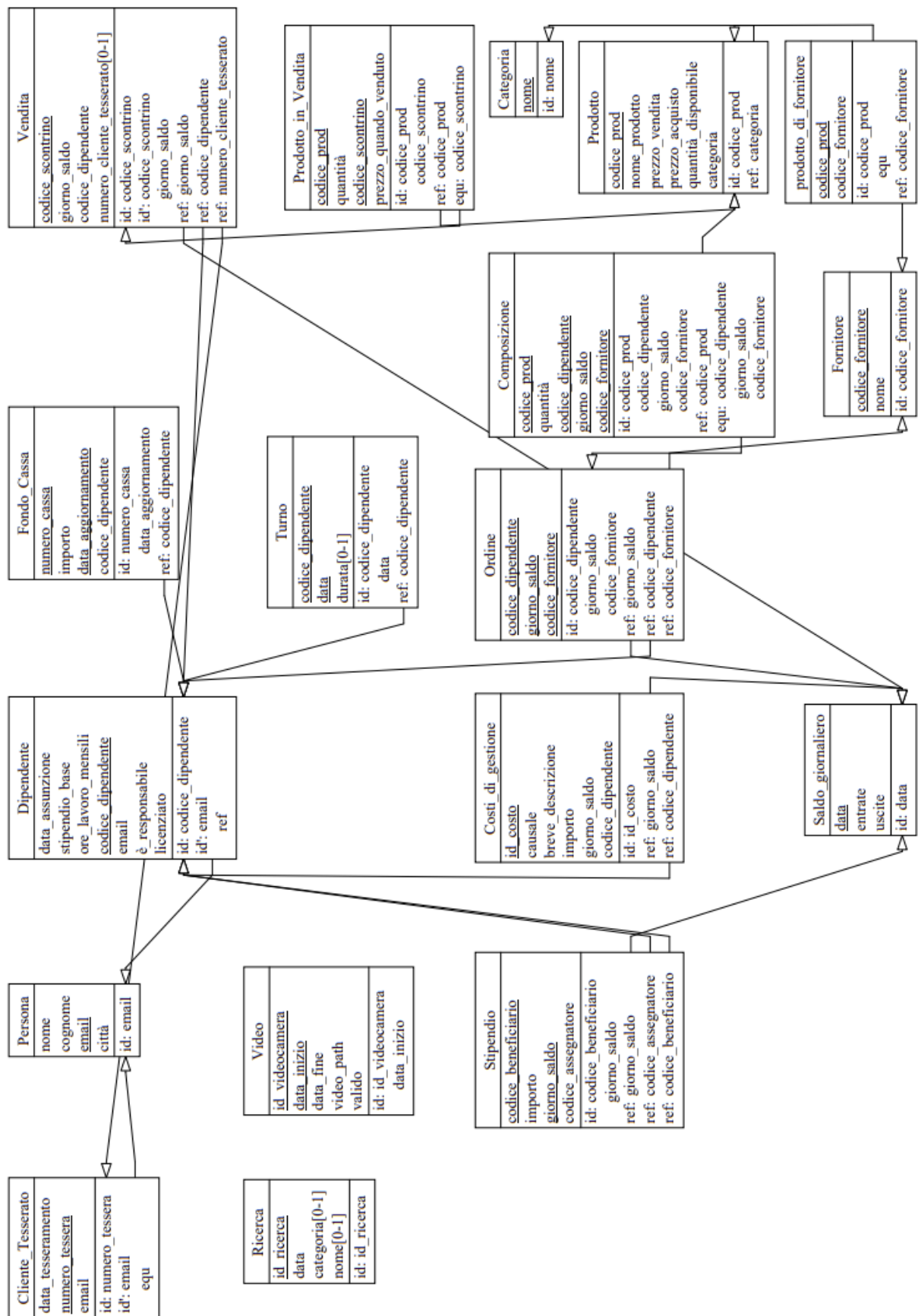
FK : codice_beneficiario REFERENCES **Dipendente**

FK : giorno_saldo REFERENCES **Saldo_giornaliero**

Video(id_videocamera, data_inizio, data_fine, video_path, valido)

Ricerca(id_ricerca, data, categoria[0-1], nome[0-1])

3.7 Schema relazionale finale



3.8 Costruzione delle tabelle del DB in linguaggio SQL

```
create table Categoria (  
    nome varchar(40) not null,  
    constraint IDCategoria primary key (nome));  
  
create table Cliente_Tesserato (  
    data_tesseramento date not null,  
    numero_tessera int not null auto_increment,  
    email varchar(40) not null,  
    constraint IDCliente_Tesserato primary key (numero_tessera),  
    constraint FKcliente_mail_ID unique (email));  
  
create table Composizione (  
    codice_prod int not null,  
    quantità int not null check (quantità > 0),  
    codice_dipendente int not null,  
    giorno_saldo date not null,  
    codice_fornitore int not null,  
    constraint IDComposizione primary key (codice_prod, codice_dipendente,  
giorno_saldo, codice_fornitore));  
  
create table Costi_di_gestione (  
    id_costo int not null auto_increment,  
    causale varchar(50) not null,  
    breve_descrizione varchar(200) not null,  
    importo DECIMAL(6, 2) not null check (importo > 0),  
    giorno_saldo date not null,  
    codice_dipendente int not null,  
    constraint IDCosti_di_gestione primary key (id_costo));  
  
create table Dipendente (  
    data_assunzione date not null,  
    stipendio_base DECIMAL(6, 2) not null,  
    ore_lavoro_mensili float(1) not null,  
    codice_dipendente int(10) not null auto_increment,  
    email varchar(40) not null,  
    è_responsabile bool not null,  
    licenziato bool not null default false,  
    password varchar(45),  
    constraint IDDipendente primary key (codice_dipendente),  
    constraint FKdipendente_mail_ID unique (email));  
  
create table Fondo_Cassa (  
    numero_cassa int not null auto_increment,
```



```

        importo DECIMAL(6,2) not null,
        data_aggiornamento datetime not null,
        codice_dipendente int not null,
        constraint IDFondo_Cassa primary key (numero_cassa,
data_aggiornamento));

create table Fornitore (
    codice_fornitore int(10) not null auto_increment,
    nome varchar(30) not null,
    constraint IDFornitore primary key (codice_fornitore));

create table Ordine (
    codice_dipendente int not null,
    giorno_saldo date not null,
    codice_fornitore int not null,
    constraint IDOrdine_ID primary key (codice_dipendente, giorno_saldo,
codice_fornitore));

create table Persona (
    nome varchar(15) not null,
    cognome varchar(15) not null,
    email varchar(40) not null,
    città varchar(40) not null,
    constraint IDPersona_ID primary key (email));

create table Prodotto (
    codice_prod int(10) not null,
    nome_prodotto varchar(40) not null,
    prezzo_vendita DECIMAL(6, 2) not null,
    prezzo_acquisto DECIMAL(6, 2) not null,
    quantità_disponibile int not null,
    categoria varchar(40) not null,
    constraint IDProdotto_ID primary key (codice_prod));

create table prodotto_di_fornitore (
    codice_prod int(10) not null,
    codice_fornitore int(10) not null,
    constraint FKè_un_ID primary key (codice_prod));

create table Prodotto_in_Vendita (
    codice_prod int(10) not null,
    quantità int not null,
    codice_scontrino int not null,
    prezzo_quando_venduto DECIMAL(6,2) not null,
    constraint IDProdotto_in_Vendita primary key (codice_prod,
codice_scontrino));

```

```

create table Ricerca (
    id_ricerca int not null auto_increment,
    data date not null,
    categoria varchar(40),
    nome varchar(100),
    constraint IDRicerca primary key (id_ricerca));

create table Saldo_giornaliero (
    data date not null,
    entrate DECIMAL(6, 2) not null,
    uscite DECIMAL(6, 2) not null,
    constraint IDSaldo_giornaliero primary key (data));

create table Stipendio (
    codice_beneficiario int not null auto_increment,
    importo DECIMAL(6, 2) not null,
    giorno_saldo date not null,
    codice_assegnatore int not null,
    constraint IDStipendio primary key (codice_beneficiario,
giorno_saldo));

create table Turno (
    codice_dipendente int not null auto_increment,
    data datetime not null,
    durata time check (durata > 0),
    constraint IDTurno primary key (codice_dipendente, data));

create table Vendita (
    codice_scontrino int not null auto_increment,
    giorno_saldo date not null,
    codice_dipendente int(10) not null,
    numero_cliente_tesserato int,
    constraint IDVendita_ID primary key (codice_scontrino),
    constraint IDVendita_2 unique (codice_scontrino, giorno_saldo));

create table Video (
    id_videocamera int not null,
    data_inizio datetime not null,
    data_fine datetime not null,
    video_path varchar(100) not null,
    valido boolean default true not null,
    constraint IDVideo primary key (id_videocamera, data_inizio));

alter table Cliente_Tesserato add constraint FKcliente_mail_FK
    foreign key (email)

```

```

        references Persona (email)
        on update cascade
        on delete cascade;

alter table Composizione add constraint FKprodotto
    foreign key (codice_prod)
    references Prodotto (codice_prod);

alter table Composizione add constraint FKcomposto
    foreign key (codice_dipendente, giorno_saldo, codice_fornitore)
    references Ordine (codice_dipendente, giorno_saldo, codice_fornitore);

alter table Costi_di_gestione add constraint FKgestione_compone
    foreign key (giorno_saldo)
    references Saldo_giornaliero (data);

alter table Costi_di_gestione add constraint FKgestione_inserisce
    foreign key (codice_dipendente)
    references Dipendente (codice_dipendente);

alter table Dipendente add constraint FKdipendente_mail_FK
    foreign key (email)
    references Persona (email);

alter table Fondo_Cassa add constraint FKinit
    foreign key (codice_dipendente)
    references Dipendente (codice_dipendente);

alter table Ordine add constraint FKordine_compone
    foreign key (giorno_saldo)
    references Saldo_giornaliero (data);

alter table Ordine add constraint FKordine_sottomette
    foreign key (codice_dipendente)
    references Dipendente (codice_dipendente);

alter table Ordine add constraint FKriceve
    foreign key (codice_fornitore)
    references Fornitore (codice_fornitore);

alter table Prodotto add constraint FKappartiene
    foreign key (categoria)
    references Categoria (nome);

alter table Prodotto add constraint is_positive
    CHECK (quantità_disponibile >= 0);

```

```

alter table prodotto_di_fornitore add constraint FKdispone
    foreign key (codice_fornitore)
    references Fornitore (codice_fornitore);

alter table prodotto_di_fornitore add constraint FKè_un_FK
    foreign key (codice_prod)
    references Prodotto (codice_prod);

alter table Prodotto_in_Vendita add constraint FKprod_in_vendita_è
    foreign key (codice_prod)
    references Prodotto (codice_prod);

alter table Prodotto_in_Vendita add constraint FKcontiene
    foreign key (codice_scontrino)
    references Vendita (codice_scontrino);

alter table Stipendio add constraint FKstipendio_compone
    foreign key (giorno_saldo)
    references Saldo_giornaliero (data);

alter table Stipendio add constraint FKstipendio_assegna
    foreign key (codice_assegnatore)
    references Dipendente (codice_dipendente);

alter table Stipendio add constraint FKstipendio_percepisce
    foreign key (codice_beneficiario)
    references Dipendente (codice_dipendente);

alter table Turno add constraint FKsvolge
    foreign key (codice_dipendente)
    references Dipendente (codice_dipendente);

alter table Vendita add constraint FKvendita_compone
    foreign key (giorno_saldo)
    references Saldo_giornaliero (data);

alter table Vendita add constraint FKvendita_effettua
    foreign key (codice_dipendente)
    references Dipendente (codice_dipendente);

alter table Vendita add constraint FKvendita_partecipa
    foreign key (numero_cliente_tesserato)
    references Cliente_Tesserato (numero_tessera);

```

3.9 Traduzione delle operazioni in query SQL

Per snellire la comunicazione tra applicativo e DBMS sono stati inseriti dei trigger:

```
DELIMITER $$
CREATE TRIGGER dopo_nuovo_prodotto_in_vendita
AFTER INSERT
ON prodotto_in_vendita FOR EACH ROW
BEGIN
    UPDATE prodotto set quantità_disponibile = quantità_disponibile -
    NEW.quantità where prodotto.codice_prod = NEW.codice_prod;
END$$
DELIMITER ;
```

```
DELIMITER $$
CREATE TRIGGER nuova_vendita
BEFORE INSERT
ON vendita FOR EACH ROW
BEGIN
    IF NEW.giorno_saldo NOT IN (SELECT data FROM saldo_giornaliero) THEN
        INSERT INTO saldo_giornaliero(data, entrate, uscite) VALUES
        (new.giorno_saldo, 0, 0);
    END IF;
END$$
DELIMITER ;
```

```
DELIMITER $$
CREATE TRIGGER nuovo_ordine
BEFORE INSERT
ON ordine FOR EACH ROW
BEGIN
    IF NEW.giorno_saldo NOT IN (SELECT data FROM saldo_giornaliero) THEN
        INSERT INTO saldo_giornaliero(data, entrate, uscite) VALUES
        (new.giorno_saldo, 0, 0);
    END IF;
END$$
DELIMITER ;
```

```
DELIMITER $$
CREATE TRIGGER nuovo_costo_di_gestione
BEFORE INSERT
ON costi_di_gestione FOR EACH ROW
BEGIN
    IF NEW.giorno_saldo NOT IN (SELECT data FROM saldo_giornaliero) THEN
```

```

            INSERT INTO saldo_giornaliero(data, entrate, uscite) VALUES
(new.giorno_saldo, 0, 0);
        END IF;
    END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER nuovo_stipendio
BEFORE INSERT
ON stipendio FOR EACH ROW
BEGIN
    IF NEW.giorno_saldo NOT IN (SELECT data FROM saldo_giornaliero) THEN
        INSERT INTO saldo_giornaliero(data, entrate, uscite) VALUES
(new.giorno_saldo, 0, 0);
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER stipendio_update_saldo
AFTER INSERT
ON stipendio FOR EACH ROW
BEGIN
    UPDATE saldo_giornaliero set uscite = uscite + new.importo where
saldo_giornaliero.data = new.giorno_saldo;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER costo_di_gestione_update_saldo
AFTER INSERT
ON costi_di_gestione FOR EACH ROW
BEGIN
    UPDATE saldo_giornaliero set uscite = uscite + new.importo where
saldo_giornaliero.data = new.giorno_saldo;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER nuovo_prodotto
BEFORE INSERT
ON prodotto FOR EACH ROW
BEGIN
    IF NEW.categoria NOT IN (SELECT * FROM categoria) THEN
        INSERT INTO categoria(nome) VALUES (NEW.categoria);
    END IF;

```

```

END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER ordine_update_quantità_prodotto
AFTER INSERT
ON composizione FOR EACH ROW
BEGIN
    UPDATE prodotto set quantità_disponibile = quantità_disponibile +
new.quantità where codice_prod = new.codice_prod;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER ordine_update_saldo
AFTER INSERT
ON composizione FOR EACH ROW
BEGIN
    UPDATE saldo_giornaliero set uscite = uscite + ((SELECT
prezzo_acquisto FROM prodotto WHERE codice_prod = new.codice_prod) *
new.quantità) where saldo_giornaliero.data = new.giorno_saldo;
END$$
DELIMITER ;

```

Di seguito sono elencate le query utilizzate per l'esecuzione delle operazioni richieste:

R1) Visualizzazione dello stato di attività dei Dipendenti

```

select nome, cognome, ore_lavoro_mensili, T.tot_ore_mensili,
Lavoro.sta_lavorando, D.email, D.stipendio_base, D.data_assunzione,
D.è_responsabile, P.città, D.codice_dipendente
from Persona P, dipendente D ,
    (select codice_dipendente, SEC_TO_TIME(SUM(TIME_TO_SEC(durata))) as
tot_ore_mensili from turno
where data between ? and ?
group by codice_dipendente) as T,
    (select codice_dipendente as CD, EXISTS (
        select *
        from turno
        where durata is null and CD = codice_dipendente
    ) as sta_lavorando
from turno
group by codice_dipendente) AS Lavoro
where P.email = D.email
AND D.codice_dipendente = T.codice_dipendente

```

```
AND D.codice_dipendente = Lavoro.CD  
AND D.licenziato = false  
group by D.codice_dipendente;
```

R2) Visualizzare le ricerche svolte

```
SELECT categoria, COUNT(*) FROM ricerca  
WHERE nome IS NULL AND data between ? and ?  
GROUP BY categoria having categoria not in ("Tutto");
```

R3) Totale vendite giornaliere

```
select ROUND(SUM(entrates), 2)  
from saldo_giornaliero  
where data between ? and ?;
```

R4) Totale vendite settimanali

Come R3.

R5) Totale vendite settimanali

Come R3.

R6) Saldo mensile

```
select ROUND(SUM(entrates) - SUM(uscite), 2)  
from saldo_giornaliero  
where data between ? and ?;
```

R7) Saldo annuale

Come R6.

R8) Inserire costi di gestione

```
INSERT INTO costi_di_gestione(causale, breve_descrizione, importo,  
giorno_saldo, codice_dipendente)  
VALUES (?, ?, ?, ?, ?);
```

R9) Inserire l'importo dello stipendio mensile per un dipendente

```
INSERT INTO stipendio(codice_beneficiario, importo, giorno_saldo,  
codice_assegnatore) VALUES ( ?, ?, ?, ?);
```

R10) Modificare i dati riguardanti il personale

```
UPDATE persona SET nome = ?, cognome = ?, città = ?  
WHERE email = ?;
```

R11) Aggiungere personale


```
INSERT INTO persona(nome, cognome, email, città)
VALUES (?, ?, ?, ?);
```

```
INSERT INTO dipendente(data_assunzione, stipendio_base, ore_lavoro_mensili,
codice_dipendente, email, è_responsabile, password) VALUES (?, ?, ?, ?, ?,
?, ?);
```

```
INSERT INTO cliente_tesserato(data_tesseramento, numero_tessera, email)
VALUES (?, ?, ?);
```

R12) Sottomettere ordini ai fornitori

```
INSERT INTO ordine(codice_dipendente, giorno_saldo, codice_fornitore)
VALUES (?, ?, ?);
```

```
INSERT INTO composizione(codice_prod, quantità, codice_dipendente,
giorno_saldo, codice_fornitore)
VALUES (?, ?, ?, ?, ?);
```

R13) Visualizzare Video di sorveglianza

```
select id_videocamera, data_inizio, timediff(data_fine, data_inizio) as
durata, video_path
from video
where valido = true;
```

R14) Rimuovere Video di sorveglianza

```
UPDATE video SET valido = false
WHERE id_videocamera = ? AND data_inizio = ?;
```

R15) Eliminare personale

```
UPDATE dipendente SET licenziato = true WHERE email = ?;
```

D1) Ricerca vendite effettuate

```
(SELECT PV.codice_prod, P.nome_prodotto, PV.prezzo_quando_venduto as
prezzo, PV.quantità, V.numero_cliente_tesserato, J.nome, J.cognome,
V.giorno_saldo FROM prodotto_in_vendita PV, vendita V, prodotto P,
( select nome, cognome
from persona P , cliente_tesserato CT, vendita V
WHERE P.email = CT.email
AND CT.numero_tessera = V.numero_cliente_tesserato AND
V.codice_scontrino = ?) as J
WHERE PV.codice_scontrino = V.codice_scontrino
AND PV.codice_prod = P.codice_prod
AND V.codice_scontrino = ?)
```

```

UNION
(SELECT PV.codice_prod, P.nome_prodotto, PV.prezzo_quando_venduto as
prezzo, PV.quantità, null, null, null, V.giorno_saldo
FROM prodotto_in_vendita PV, vendita V, prodotto P
WHERE PV.codice_scontrino = V.codice_scontrino
      AND PV.codice_prod = P.codice_prod
      AND V.codice_scontrino = ?)
LIMIT 1;

```

D2) Registrare Clienti Tesserati

```

INSERT INTO persona(nome, cognome, email, città)
VALUES ( ?, ?, ?, ?);

```

```

INSERT INTO cliente_tesserato(data_tesseramento, email)
VALUES (NOW(), ?);

```

D3) Modificare informazioni riguardanti i Clienti Tesserati

```

UPDATE persona SET
nome = ?
cognome = ?
città = ?
WHERE email = ?;

```

D4) Segnare inizio e fine del proprio turno lavorativo

```

UPDATE turno SET durata= ?
WHERE codice_dipendente = ?
AND durata IS NULL;

```

D5) Inizializzare il fondo cassa

```

INSERT INTO fondo_cassa (numero_cassa, importo, data_aggiornamento,
codice_dipendente) VALUES( ?, ?, ?, ?);

```

D6) Effettuare vendite

```

INSERT INTO vendita (codice_scontrino, giorno_saldo, codice_dipendente,
numero_cliente_tesserato) VALUES (NULL, ?, ?, ?);

```

```

INSERT INTO prodotto_in_vendita(codice_scontrino, codice_prod, quantità,
prezzo_quando_venduto) VALUES (?, ?, ?, ?);

```

C1) Effettuare ricerche di prodotti per nome

```

SELECT nome_prodotto AS NOME, categoria AS CATEGORIA, prezzo_vendita AS
PREZZO, quantità_disponibile as QTA

```

```
FROM prodotto
WHERE nome_prodotto LIKE '%?%';
```

C2) Effettuare ricerche di prodotti per categoria

```
SELECT nome_prodotto AS NOME, categoria AS CATEGORIA, prezzo_vendita AS
PREZZO, quantità_disponibile as QTA
FROM prodotto
WHERE CATEGORIA = ?;
```

Capitolo 4 - Progettazione dell'applicazione

4.1 Descrizione dell'architettura dell'applicazione realizzata

L'applicazione è stata sviluppata usando il linguaggio di programmazione Java. Nello specifico è stata usata la libreria JavaFX per ogni aspetto riguardante la GUI, JDBC per la comunicazione con il DB.

Il programma ha bisogno di essere lanciato da una directory contenente un file chiamato "config" che contiene le informazioni necessarie a connettersi al DB. Quindi se fosse necessario modificare indirizzo, nome utente o password del DB, quel file dovrà essere aggiornato di conseguenza.

Per garantire una installazione rapida ed user-friendly durante il primo avvio del programma è necessario effettuare il login utilizzando le credenziali :

- codice dipendente : 1
- password : adminpassword

Una volta eseguito il login presso questo account fittizio è possibile registrare altri dipendenti tramite la scheda "Alti Privilegi" > "Gestisci personale".

E' consigliato registrare almeno un nuovo account Responsabile, effettuare il logout, effettuare il login con le credenziali del nuovo account, andare nuovamente in "Alti Privilegi" > "Gestisci personale", selezionare la riga della tabella che non ha nè nome nè cognome, premere "Mostra info", premere su "Licenzia".

Nel caso si voglia provare l'applicazione con alcuni dati già inseriti nel database, è presente un file [fill DB.sql](#) che contiene alcune istruzioni per popolare il DB.

Una volta effettuato il login, nella sezione "Turno" è possibile cliccare su:

- "Inizia" per segnare l'inizio del proprio turno lavorativo;
- "Termina" per segnare la fine del proprio turno lavorativo;
- "Logout" per chiudere l'applicazione (NON termina il turno).

Schermata di LOGIN:

Nella schermata di login è possibile effettuare:

- il login se si è un dipendente
- l'accesso alle funzionalità riservate al computer lasciato a disposizione dei clienti

Schermata Cliente:

Possono essere effettuate ricerche per nome o per categoria.

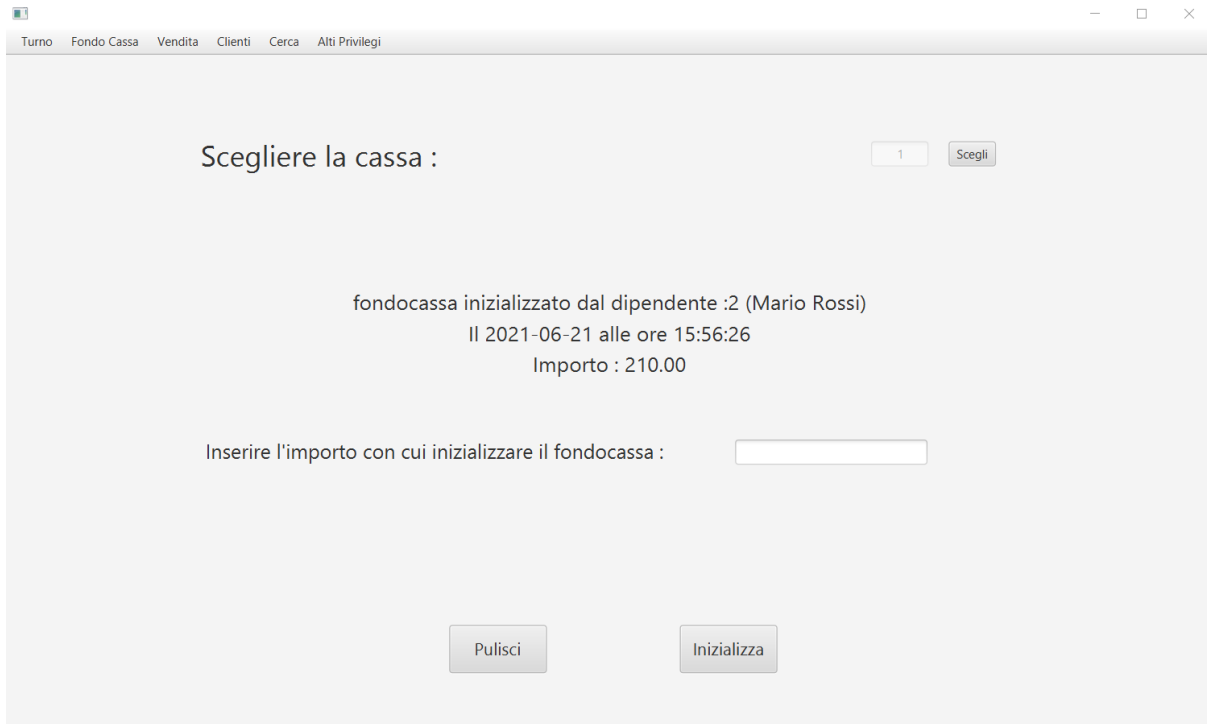
L'applicazione mostra prezzo e disponibilità dei prodotti.

Inoltre le ricerche vengono salvate nel DB.

[illegible]

Schermata Inizializza Fondo Cassa:

Per ogni cassa è possibile dichiarare a quanto ammonta il fondo cassa.



Scegliere la cassa :

1 Scegli

fondocassa inizializzato dal dipendente :2 (Mario Rossi)
Il 2021-06-21 alle ore 15:56:26
Importo : 210.00

Inserire l'importo con cui inizializzare il fondocassa :

Pulisci Inizializza

Schermata Effettua Vendita:

In vista dell'uso di schermi touchscreen la digitazione del codice del prodotto va svolta premendo i tasti del tastierino numerico a schermo.

Il tasto "d" cancella l'ultimo carattere inserito.

il tasto # va inserito dopo il codice del prodotto da vendere, seguito dal numero di articoli del medesimo prodotto da voler acquistare (in caso di assenza di questo carattere, la quantità assunta sarà considerata pari ad una unità).

"Enter" aggiunge il prodotto nella lista presente sulla sinistra.

Il campo "Codice Cliente" permette di inserire il codice tessera di un cliente tesserato.

"Termina" salva la vendita nel DB.

"Cancella" resetta l'intera schermata.

Inserendo il codice scontrino vengono visualizzati i dati della vendita a cui il codice scontrino fa riferimento.

Data : 2021-06-21

Schermata Cerca Clienti:

In questa schermata è possibile:

- cercare un cliente tramite il suo codice tessera;
- cercare un cliente tramite la sua email;
- cercare i dati di una qualsiasi Persona presente nel DB inserendo l'email associata;
- registrare nuovi clienti;
- aggiornare dati dei clienti, è possibile inserire i dati a mano o effettuare prima la ricerca tramite la casella di testo per poi modificare i campi che richiedono un aggiornamento.
- eliminare un Cliente Tesserato.

Schermata Cerca Prodotto:

Identica alla Schermata Cliente.

Schermata Gestisci Personale:

Qui è possibile visualizzare:

- lo stato di attività dei dipendenti;
- quanto tempo hanno lavorato nell'ultimo mese
- bilancio, entrate, uscite in un certo lasso di tempo
- selezionando una riga nella tabella contenente i dipendenti e premendo poi "Mostra Info", i campi di testo a destra vengono riempiti con i dati del dipendente scelto.

[illegible]

Qui è possibile inserire spese di tre tipi:

- Turno

Fondo Cassa

Vendita

Clienti

Cerca

Alti Privilegi

Nome spesa

Descrizione

Dipendente :

Mario Rossi

Assunto da :

2020-10-01

Ore contratto :

80.0

Ore effettive :

00:00:00

Stipendio base :

1200.00

E' un responsabile :

SI

Data ultimo stipendio :

2021-06-01

Importo ultimo stipendio :

1300.00

-

1200.00

+

Salva

Aggiorna

Effettua

Giammarco Sars SRLS
Thomann SRLS

nome	nome_prodotto	prezzo_acquisto
Thomann SRLS	Bic JelPen 0.2	1.20
Thomann SRLS	Duracell Batteria 9V	2.50

Prodotto :
Duracell Batteria 9V

Quantità

-10

-

1

+

+10

Aggiungi al carrello

Fornitore	Prodotto	Prezzo	Quantità
Thomann SRLS	Bic JelPen 0.2	1.20	21

Totale : 25.2

Annulla

Acquista

Schermata Aggiungi Prodotto:

Qui è possibile inserire nuovi prodotti nel catalogo.

Vi sono dei suggerimenti per quanto riguarda categoria e fornitore del prodotto che si sta aggiungendo, tuttavia è possibile inserire autonomamente entrambi e, nel caso di inserimento di una nuova categoria/nuovo fornitore, questo viene aggiunto alla tabella Categoria/Fornitore.

Se viene inserito il codice prodotto di un prodotto già presente nel DB, le informazioni di quest'ultimo vengono aggiornate.

Turno Fondo Cassa Vendita Clienti Cerca Alti Privilegi

Lista fornitori registrati :

Giammarco Sars SRLS

Thomann SRLS

Lista categorie registrate :

cancelleria

elettronica domestica

telefonia

Aggiorna

Fornitore :

Nome prodotto :

Prezzo acquisto :

Prezzo vendita :

Categoria :

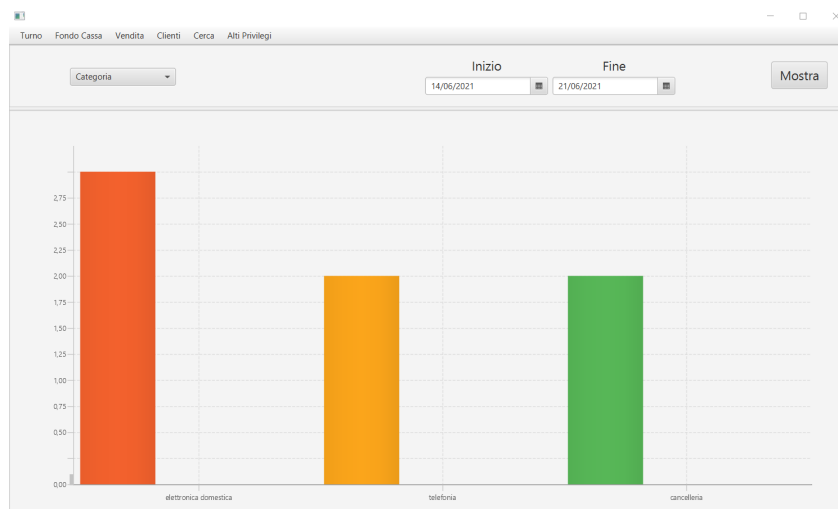
Codice Prodotto :

Pulisci Aggiungi

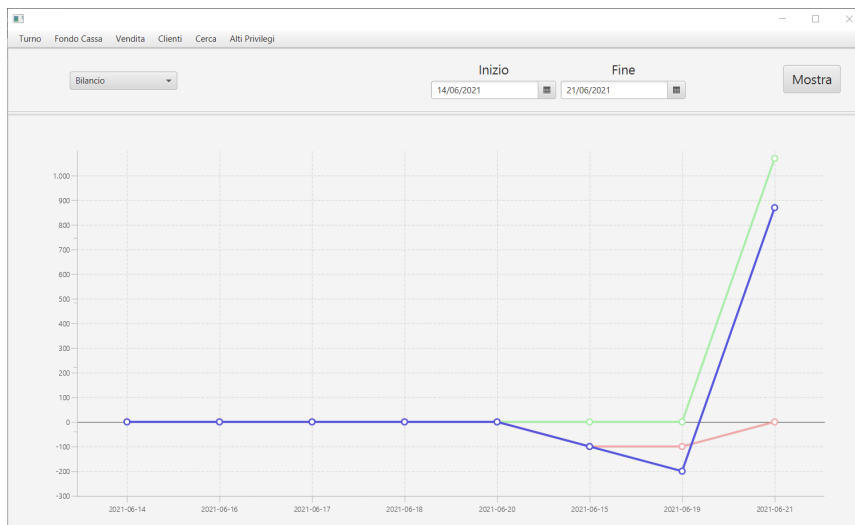
Schermata Mostra Statistiche:

Mostra due tipi di statistiche:

- statistiche sulle categorie più cercate;



- grafico delle fluttuazioni del bilancio in un dato periodo.



Schermata Video di Sicurezza:

Una volta selezionata la riga riguardante il video di sicurezza è possibile:

- riprodurre il video;
- aggiornare la lista dei video;
- rimuovere il video;

[illegible]