UNIVERSITÀ DEGLI STUDI DI SALERNO

PROGETTO BASI DI DATI

Sistema di acquisizione, gestione ed analisi di un supermercato.

Christian Di Maio

Vittorio Graziano

Giuseppe Giordano

Giuseppe De Filippo

Cdl Ingegneria Informatica

Prof. Ing. Antonio d'Acierno

Sommario

Descrizione della realtà di interesse

Si vuole gestire il sistema di acquisizione, gestione ed analisi di un supermercato. La realtà cui si prefigge la gestione, prevede l' esistenza di 3 macro-realtà su cui sviluppare l' analisi :

1) Realtà Supermercato
2) Realtà CEDI (Centro di Distribuzione)
3) Realtà Cliente

Supermercato

Il Supermercato è il nucleo della Base di Dati, prevede la gestione del personale del supermercato, la gestione dei turni del personale, la gestione della fatturazione degli scontrini in cassa, la gestione dei vari reparti del supermercato, la gestione delle fidelity-card per i clienti, la gestione degli articoli presenti nel supermercato ed infine la gestione degli ordini verso il CEDI.

CEDI

I CEDI sono le unità autonome che hanno il compito di rifornire il supermercato, gestiscono le informazioni sugli articoli (ogni CEDI rifornisce il supermercato con i propri articoli) e per ognuno di esso è necessario tenere traccia della sua posizione geografica.

Cliente

Il Cliente è colui che sottoscrive un contratto con il supermercato, il committente utilizza una carta fedeltà con la quale gestisce un concorso a premi basato sul punteggio di ogni cliente.

Specifiche

Dopo una prima analisi, si riporta su carta la descrizione informale della realtà, concordata con il cliente.

Supermercato

Reparto
Il reparto rappresenta l' area alla quale fa parte ogni articolo presente all'interno del supermercato. Ex "Freschi"-"Banco"-"Salumi". Per ogni reparto si tiene traccia del tipo, di una breve descrizione e del responsabile.

Turno
Il Turno rappresenta, come la parola lascia intendere, la fascia oraria (inizio e fine) che tutto il personale deve rispettare (il cliente non permette ai suoi dipendenti straordinari). Per esso bisogna tenere traccia dell' ora inizio turno, della ora fine turno e della data del turno. Ogni turno è della durata di 4 ore.

Personale
Il Personale rappresenta il collettivo di persone che lavora all'interno del supermercato. Il cliente preferisce gestire i suoi dipendenti consentendo loro di svolgere qualunque attività attinente alla gestione delle casse e all' inventario, l'unica distinzione che viene fatta all'interno del personale è che vi sono i responsabili di reparto ed gli operatori semplici che fanno riferimento ai rispettivi responsabili. Qualunque sia la mansione del dipendente per esso vengono raccolte le seguenti informazioni: Nome, Cognome, Anno di Nascita, Codice Fiscale(CF) e Sesso, tranne il Numero di telefono che viene salvato solo per i dipendenti.

Tessera
La tessera o fidelity-card viene gestita attraverso l'acquisizione del : Numero tessera e la data di sottoscrizione del contratto. Un cliente possiede una sola tessera.

Ordine CEDI
L' Ordine CEDI nasce nel momento in cui il supermercato ha bisogno di acquisire merce dal fornitore, per l'ordine è necessario tenere traccia dell' articolo da ordinare(il suo EAN), il numero identificativo del CEDI, la quantità da ordinare e se l'ordine è stato ricevuto ed evaso dal cedi.

Articolo Supermercato
L'articolo, visto dal lato supermercato, è l'unione delle informazioni fornite dal cedi (EAN, NOME, DESCRIZIONE, PREZZO) più le informazioni gestite internamente dal supermercato : quantità, prezzo di vendita, reparto in cui si trova l'articolo, informazioni se l'articolo è in offerta ed a quanto ammonta l'offerta. Il supermercato non ha la possibilità di registrare nuovi articoli che non sono forniti da alcun CEDI ed inoltre non può modificare informazioni circa l'EAN, il nome e la descrizione, i quali sono informazioni che conservano i CEDI per i rispettivi articoli.

Scontrino
Lo scontrino della spesa riporta con sé un ID univoco, la data di fatturazione, il totale della spesa ed il punteggio che il cliente ha ottenuto per quella spesa.

CEDI

CEDI
Per ogni CEDI vi è la necessità di tenere traccia delle seguenti informazioni: ID del CEDI, Nome, Posizione Geografica(Citta, Via, Numero Civico), numero p. iva.

Articolo CEDI
Articolo cedi raccoglie tutti gli articoli che il CEDI mette a disposizione del supermercato, vengono raccolte informazioni circa : il nome dell' articolo, la descrizione, l'ean ed il prezzo all' ingrosso

CLIENTE

Cliente
Del cliente il committente vuole gestire le informazioni circa: Il nome, Il cognome, il CF, la data di nascita e la quantità di punti a cui il cliente è arrivato.

SPECIFICHE UTENZA

Il cliente decide di mettere a disposizioni 5 tipologie di utenza che lavora sul database

- **CEDI** : non ha nessuno controllo sul supermercato, può solo gestire le due realtà ad esso collegate "CEDI" e "Articolo CEDI", nello specifico per articolo cedi non può effettuare aggiornamenti sugli articoli.

- **Titolare** : ha il controllo su tutta la realtà tranne che sulle realtà del CEDI, infatti per esse può solo visualizzarne le informazioni.

- **Dipendente** : ha la possibilità di visualizzare le turnazioni mensili, non può effettuare ordini verso il cedi, non ha accesso alle informazioni dei cedi, può visualizzare dettagli sui reparti.

- **Responsabile** : può gestire tutta la realtà del supermercato, ma non può effetturare ordini.

- **Cliente** : non ha accesso a nessuna informazione interna del supermercato, può solo visualizzare le sue informazioni, lo stato della tessara, gli scontrino e le righe ad esso collegato.

Lista Operazioni principali

Operazione 1: Inserire un nuovo ordine per i CEDI

Operazione 2: Generare scontrino della spesa

Operazione 3: Aggiungere un nuovo cliente

Operazione 4: Aggiornare o Inserire un nuovo articolo nell'anagrafica del supermercato

Operazione 5: Gestire i turni del personale

Ipotesi Aggiuntive

1) Un supermercato nasce senza articoli.

2) Un cedi per i propri articoli non ha limiti sulle quantità erogabili, poiché egli è colui che rifornisce. Si suppone inoltre che per il cedi non si rende necessario tenere traccia della quantità di ogni articolo (per il motivo di cui sopra).
3) Per fornire una più generica soluzione al cliente si decide di prevedere anche turni notturni (EX. supermercato 7/24)

4) Il supermercato dispone di 12 reparti, ognuno di essi prende come nominativo "Reparto X" dove la X sta per il numero

5) Ogni CEDI non commercia articoli di altri CEDI

6) Ogni articolo si differenzia tramite EAN

7) Il CEDI ogni qualvolta genera un nuovo articolo per la sua anagrafica non può più modificarne le informazioni, se ha bisogno di effettuare delle modifiche (ES. edizione speciale Coca-Cola) dovrà aggiungere alla propria anagrafica un nuovo articolo, con ean diverso.

8) La stessa ipotesi aggiuntiva si propaga lato supermercato, egli non può effettuare aggiornamenti o modifiche sui campi acquisiti dal CEDI.

1.5 Ipotesi Volumi*:

*Questa ipotesi sui volumi è sviluppata su una prima rassegna probabilistica delle entità che saranno coinvolte nella base dati, dopo lo sviluppo conclusivo del Modello E-R si avrà una panoramica generale e più precisa sui reali volumi della Base di Dati e ne verrà riproposta una più completa.

Supermercato

Entità Rilevate*	Dimensioni	Off-Set
Reparto	30	~ 5%
Turno	6	~ 0%
Personale	100	~ 15%
Tessera	10.000	~ 10%
Ordine CEDI	100.000	~ 30%
Articolo_Supermercato	10.000	~ 30%
Scontrino	1.000.000	~ 10%
Chiusura	10.000	~ 10%

CEDI

Entità Rilevate*	Dimensioni	Off-Set
CEDI	1.000	~5%
Articolo_CEDI	100.000	~10%

Cliente

Entità Rilevate*	Dimensioni	Off-Set
Cliente	10.000	~ 30%

Legenda

Probabilità di errore sulla dimensione minima

Probabilità di errore sulla dimensione media

Probabilità di errore sulla dimensione alta

2.1 Progettazione Concettuale

Lo scopo della progettazione concettuale è quello di rappresentare le specifiche informali della realtà di interesse in termini di una descrizione formale e completa, ma indipendentemente dai criteri di rappresentazione utilizzati nei sistemi di gestione di basi di dati. Il prodotto di questa fase viene chiamato schema concettuale dei dati e fa riferimento a un modello concettuale dei dati che ci consente di descrivere l'organizzazione dei dati ad un alto livello di astrazione. In particolare, utilizzeremo il modello E-R che da modello concettuale fornisce una serie di strutture dette costrutti atte a descrivere la realtà di interesse. Questa fase produce in uscita lo schema E-R, da noi realizzato con il supporto dello strumento C.A.S.E. Power Designer, e la documentazione di supporto di seguito riportati.

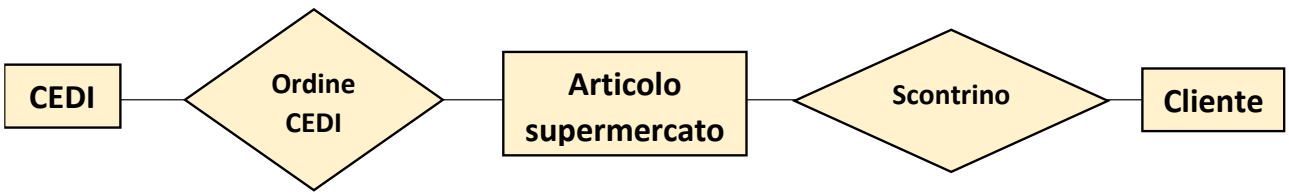
Modello E-R

Per la costruzione del modello E-R si è deciso di utilizzare la strategia mista; partendo quindi da uno "schema scheletro" (in formalismo carta e penna) si è passato al raffinamento delle entità fondamentali (Power Designer).

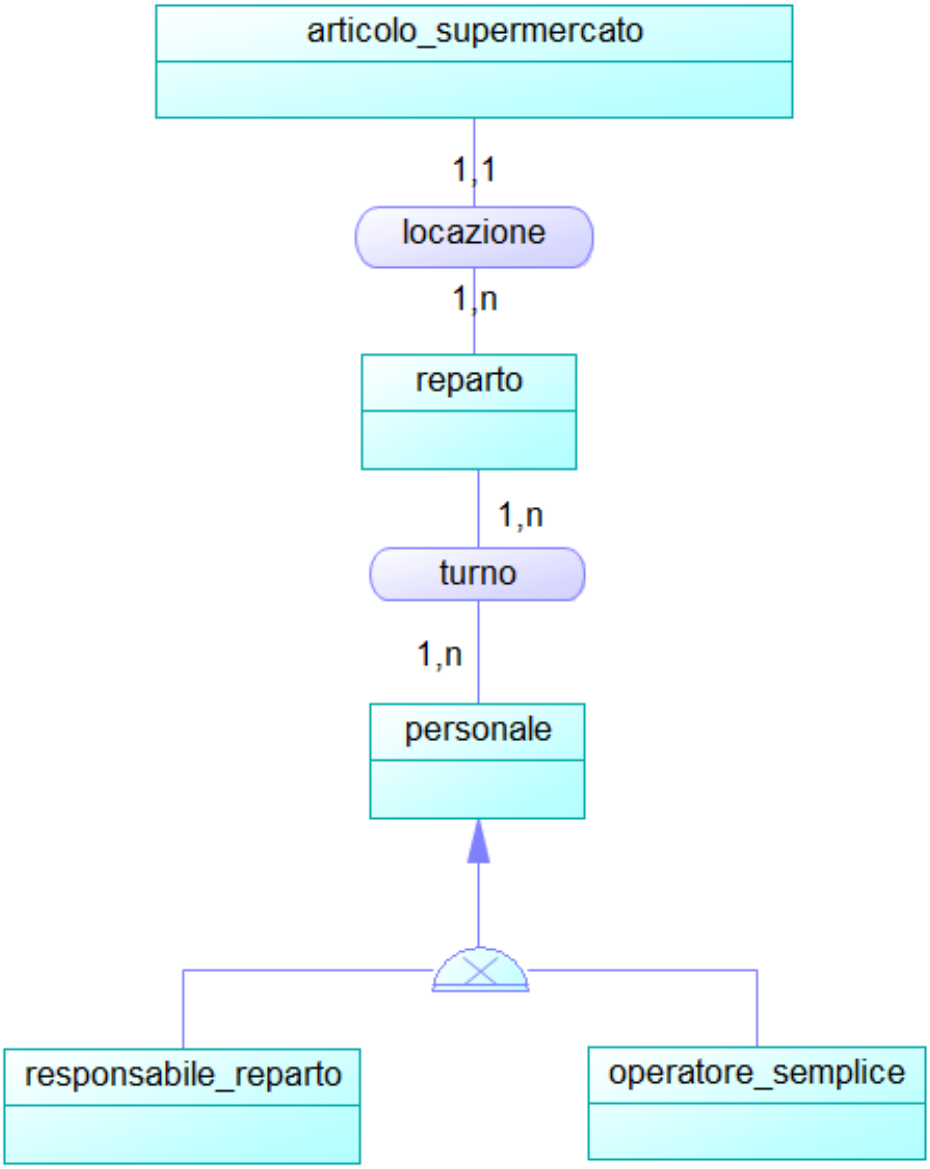
Strategia Mista

Questo schema fornisce una visione unitaria dell'intero progetto e favorisce le successive fasi di sviluppo. Per la nostra realtà sono stati individuati 3 concetti principali : CEDI, Articolo supermercato e Cliente

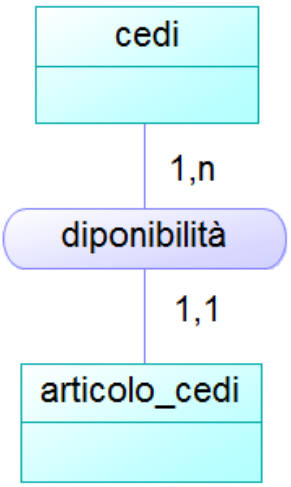
Schema scheletro



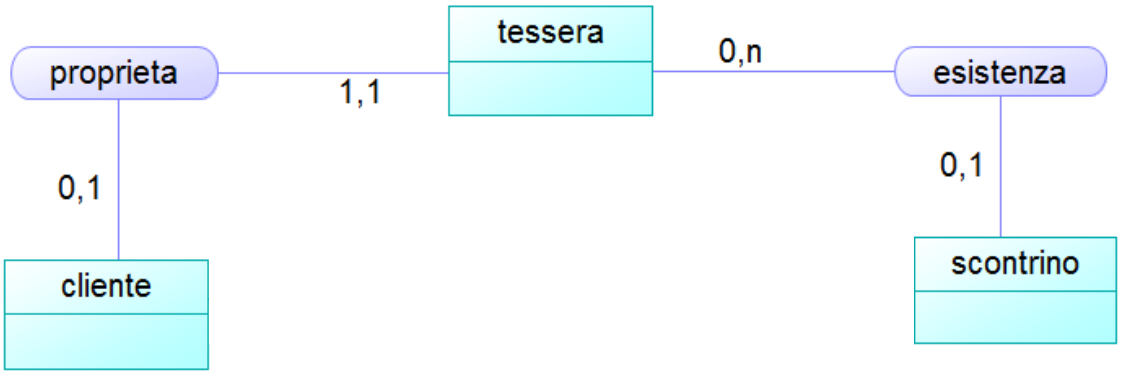
Raffinamento dell'entità "articolo_supermercato"



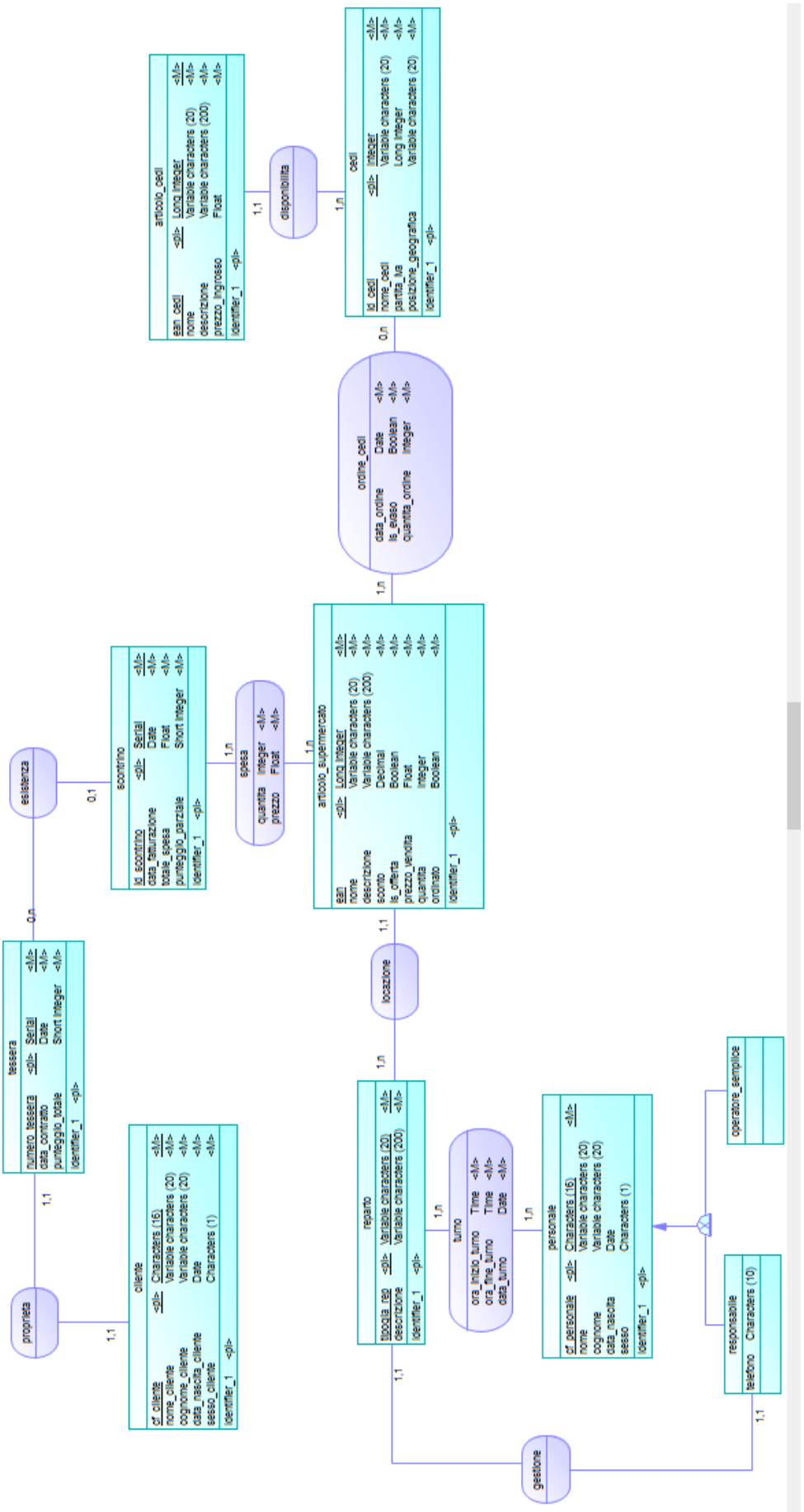
Raffinamento dell'entità "CEDI"



Raffinamento dell'entità "Cliente"



Integrazione – Schema ER completo



Documentazione

Dizionario dei dati per le **entità** più importanti:

Entità	Descrizione	Attributi	Identificatore
Reparto	Il reparto rappresenta il reparto nel quale è situato un articolo del supermercato. Ex "Freschi"-"Banco"-"Salumi". Per ogni reparto bisogna tenere traccia della tipogia, di una breve descrizione e del codice fiscale del responsabile del reparto.	tipologia_rep, descrizione	tipologia_rep
Personale	Il Personale rappresenta il collettivo di persone che lavora all'interno del supermercato. Si preferisce gestire i dipendenti consentendo loro di svolgere qualunque attività attinente alla gestione delle casse e all' inventario, l'unica distinzione che viene fatta all'interno del personale è che vi sono i responsabili di reparto e gli operatori semplici che fanno riferimento ai rispettivi responsabili. Qualunque sia la mansione del dipendente per esso vengono raccolte le seguenti informazioni: nome, cognome, data e luogo di nascita, codice fiscale(CF), sesso.	cf_personale, nome, cognome, data_nascita, sesso, luogo_nascita	cf_personale
Tessera	La tessera o fidelity-card viene gestita attraverso l'acquisizione del : Numero tessera e la data di sottoscrizione del contratto.	numero_tessera, data_contratto, punteggio	numero_tessera

Articolo_ supermercato	L'articolo, visto dal lato supermercato, è l'insieme delle informazioni fornite dal cedi (EAN, NOME, DESCRIZIONE, PREZZO) più le informazioni gestite internamente dal supermercato : quantità, prezzo di vendita, reparto in cui si trova l'articolo, se l'articolo è in offerta e se in quel momento è attivo un ordine per quell'articolo(ordinato). Il supermercato ha solo la possibilità di inserire o aggiornare i dati che sono utili per la sua realtà e non quelli che assimila dal CEDI.	ean, nome, descrizione, sconto, is_offerta, prezzo_vendita, ordinato, disponibilità	ean
Scontrino	Lo scontrino della spesa riporta con sé un ID univoco Nazionale, la data di fatturazione, il totale della spesa e il numero e punteggio della tessera.	id_scontrino, data_fatturazione, totale_spesa, punteggio	id_scontrino
CEDI	Per ogni CEDI vi è la necessità di tenere traccia delle seguenti informazioni: ID del CEDI, Nome, Posizione Geografica(Città,Via,numero_civico), numero p. iva.	id_cedi, nome, partita_iva, posizione_geografica	id_cedi
Articolo_CEDI	Articolo cedi raccoglie tutti gli articoli che il CEDI mette a disposizione del supermercato, vengono raccolte informazioni circa : il nome dell' articolo, la descrizione, l'ean, il prezzo all' ingrosso e l'identificativo del CEDI a cui appartiene.	ean, nome, descrizione, prezzo_ingrosso	ean
Cliente	Del cliente si vogliono gestire le informazioni circa: Il nome, Il cognome, il CF, la data di nascita, il luogo di nascita e il sesso.	cf_cliente, nome, cognome, data_nascita, luogo_nascita, sesso	cf_cliente

Dizionario dei dati per le **associazioni** più importanti:

Associazione	Descrizione	Attributi	Entità coinvolte
Ordine_ CEDI	L' Ordine CEDI nasce nel momento in cui il supermercato ha bisogno di acquisire merce dai fornitori, per l'ordine è necessario tenere traccia: l' articolo che voglio ordinare(il suo EAN), il numero identificativo del fornitore, la quantità da ordinare, il costo totale dell'ordine e se l'ordine è stato ricevuto ed evaso dal cedi.	data_ordine, quantità, is_evaso	articolo_supermercato (1,N) cedi (0,N)
Turno	Il turno rappresenta la fascia oraria (ora inizio e fine) che tutto il personale deve rispettare. Per esso bisogna tenere traccia del codice fiscale dell'addetto, dell'ora di inizio turno, dell'ora di fine turno, della data del turno ed infine del reparto in cui l'addetto svolge il turno. Il personale compie un solo turno lavorativo al giorno. Le informazioni dei turni del personale vengono mantenuti per la durata di 30 giorni.	ora_inizio_turno ora_fine_turno, data_turno,	reparto (1,N) personale (1,N)
Disponibilità	Disponibilità costituisce l'associazione tra l'articolo del CEDI ed il CEDI che dispone di quel determinato articolo.	NESSUNO	articolo_cedi (1,1) cedi (1,N)
Spesa	Spesa rappresenta la singola riga che viene stampata sullo scontrino	quantità, prezzo	scontrino (1,N) articolo_supermercato (1,N)

PROGETTAZIONE LOGICA

CARICO APPLICATIVO

Il carico applicativo è da supporto alla progettazione logica nella fase di ottimizzazione del progetto, di seguito si riporta: la tavola dei volumi (in riferimento al p. 1.2), le operazioni principali e la tavola degli accessi per ogni operazione.

TAVOLA DEI VOLUMI

CONCETTO	TIPO	VOLUME
Cliente	E	15.000
Tessera	E	10.000
Scontrino	E	1.000.000
Reparto(il supermercato ha 12 reparti)	E	12
Articolo supermercato	E	10.000
Cedi	E	1.000
Articolo cedi	E	100.000
Personale	E	100
Proprietà	R	10.000
Esistenza	R	20.000
Appartenenza	R	1.000.000
Spesa In media ogni scontrino presenta 3 articoli, ogni giorno evade in media 300 scontrini*	R	90.000
Locazione	R	10.000
Ordine cedi Si assume il 20% in più del numero di articoli presenti nel supermercato*	R	12.000
Disponibilità	R	100.000
Turno In media in ogni reparto lavorano 3 addetti.*	R	30.000
*Considerazioni effettuate per stimare i volumi.		

TAVOLA DELLE OPERAZIONI:

OPERAZIONE	*DEESCRIZIONE*	*TIPO*	*FREQUENZA*
Op. 1	Inserire un nuovo ordine per il CEDI	B	100 volte al giorno
Op. 2	Generare scontrino della spesa, e gestire i punti della relativa "carta fedeltà"	I	300 volte al giorno
Op. 3	Aggiungere un nuovo cliente	I	4 volte al giorno
Op. 4	Inserire un nuovo articolo o/e la relativa quantità nell'anagrafica del supermercato	I	10 volte al giorno
Op. 5	Gestire i turni del personale	B	1 volta al mese

Tavola degli accessi per l'operazione 1.

Concetto	*Costrutto*	*Accessi*	*Tipo*
Articolo_cedi	E	1	L
Ordine_cedi	A	1	S

Tavola degli accessi per l'operazione 2.

Concetto	*Costrutto*	*Accessi*	*Tipo*
Spesa	A	N* x 1	S
Scontrino	E	1	S
Tessera	E	1	S

**N sta ad indicare il numero medio di articoli per scontrino, che abbiamo supposto a 3*

Tavola degli accessi per l'operazione 3.

Concetto	*Costrutto*	*Accessi*	*Tipo*
Cliente	A	1	S
Tessera	E	1	S

Tavola degli accessi per l'operazione 4.

Concetto	***Costrutto***	***Accessi***	***Tipo***
Ordine_cedi	A	1	L
Articolo_supermercato	E	1	S

Tavola degli accessi per l'operazione 5.

Concetto	***Costrutto***	***Accessi***	***Tipo***
Personale	A	N*	L
Reparto	A	12	L
Turno	E	N* x 30	S

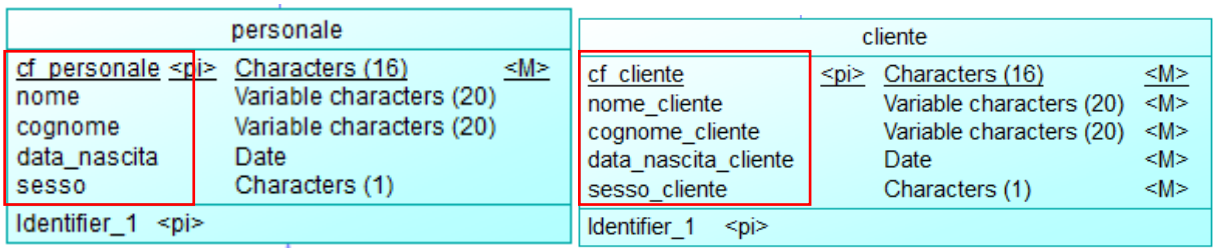
**N sta ad indicare il numero medio del personale*

Ristrutturazione

ANALISI DELLE RIDONDANZE

Lo schema E-R in ingresso a questa fase progettuale, presenta le seguenti ridondanze:

Ridondanza n. 1



DESCRIZIONE:

L' entità "personale" e "cliente" hanno tutti gli attributi in comune poiché entrambe rappresentano delle informazioni circa la persona.

ANALISI:

Eliminare questa ridondanza significherebbe aggiungere una nuova entità : "Cliente_Personale"; nel caso in questione però il numero di dipendenti è minimo rispetto al numero di clienti, in particolare si sprecherebbe una quantità di memoria al massimo pari ad:

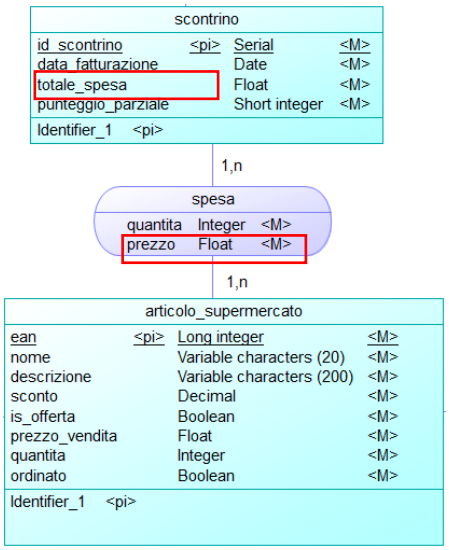
- dimensione occorrenza * numero occorrenze, quindi avremo: 60 byte *100= 6 KB.

In seconda battuta è plausibile pensare che essendo una realtà dinamica, le entità personale o cliente possano gestire in futuro nuove informazioni, questo ci porta a realizzare una soluzione quanto più generica possibile tenendo le due entità separate.

Conclusione:

Questa quantità è trascurabile rispetto alle dimensioni complessive, quindi si sceglie di non eliminare la ridondanza.

Ridondanza n. 2



DESCRIZIONE:

L'attributo totale_spesa nell' entità scontrino può essere ottenuto, con un relativo costo computazionale, sommando il prezzo delle singole tuple in spesa che afferiscono allo scontrino.

ANALISI:

Operazioni coinvolte:

- Generare scontrino della spesa, e gestire i punti della relativa "carta fedeltà" (Op2).

Memoria aggiunta rispetto all' assenza di ridondanza:

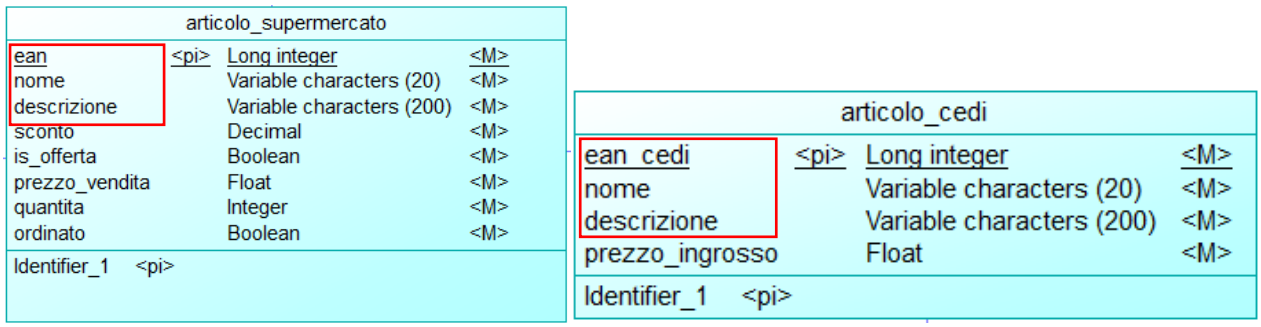
- 4 Byte(dimensione float) * 1.000.000(n. occorrenze scontrino) = 4.000.000 byte= 4MB

Conclusione:

Questa quantità è irrisoria rispetto alla dimensione della base dati inoltre, porta al vantaggio di avere il dato pronto in caso di operazioni su di esso.

Quindi si decide di non eliminare la ridondanza.

Ridondanza n. 3



DESCRIZIONE:

Date le specifiche fornite dal cliente, articolo_supermercato ed articolo_cedi portano informazioni ripetute su: ean,nome e descrizione. Ean poiché per legge lo stesso articolo deve avere ean uguale indipendentemente dalla locazione del suddetto(supermercato o cedi), idem per le altre due informazioni

ANALISI:

Memoria aggiunta in presenza di ridondanza (assumendo come caso peggiore che il supermercato abbia tutti gli articoli di tutti i cedi)

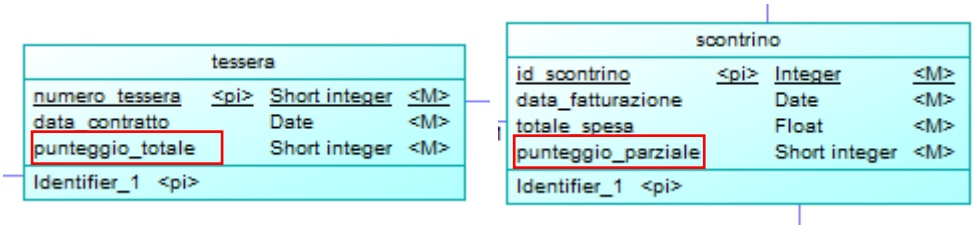
- 228 byte(dimensione occorrenza) * 100.000(numero occorrenze)=22.800.000 byte=22,8 MB

In seconda battuta è plausibile pensare che essendo una realtà dinamica, il magazzino possa gestire in futuro nuove informazioni per i propri articoli (articolo_cedi), questo ci porta a realizzare una soluzione quanto più generica possibile tenendo le due entità separate.

Conclusione:

La quantità di memoria utilizzata è irrisoria, inoltre questi dati vengono utilizzati da alcune operazioni, quindi si sceglie di tenere la ridondanza a favore della disponibilità immediata dei dati.

Ridondanza n. 4



DESCRIZIONE:

L'attributo punteggio_totale dell'entità tessera può essere ottenuto aggiornando, ad ogni generazione dello scontrino cui fa riferimento la tessera, il suo contenuto sommando se stesso al punteggio generato per quello scontrino. Per poter tenere traccia del punteggio che il cliente ottiene ad ogni scontrino si adotta l'attributo punteggio_parziale nell'entità scontrino

ANALISI:

Memoria aggiunta in presenza di ridondanza

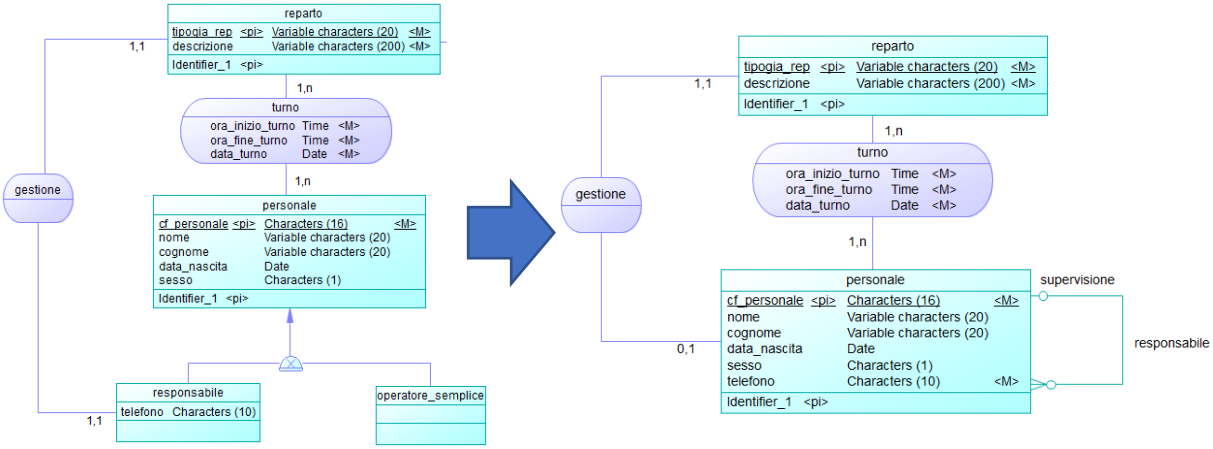
- 2 bytes(dimensione occorrenza) * 1.000.000(numero occorrenze)=2.000.000 byte=2,2 MB

Conclusione:

La quantità di memoria utilizzata è irrisoria, inoltre questi dati vengono utilizzati da alcune operazioni, quindi si sceglie di tenere la ridondanza a favore della disponibilità immediata dei dati.

ELIMINAZIONE DELLE GERARCHIE

Lo schema analizzato presenta la seguente gerarchia:



DESCRIZIONE:
Si tratta di una generalizzazione totale ed esclusiva, atta a descrivere la specializzazione dell' entità "personale" in: "responsabile reparto" ed "operatore semplice".

ANALISI:
In questo caso visto che la realtà non prevede di collezionare informazioni specifiche per le entità figlie si decide di procedere in questo modo: viene scelto di accorpare le entità figlie nell' entità padre, per consentire questo accorpamento si introduce una nuova associazione tra personale e sé stesso che porta l'informazione su la distinzione tra i responsabili e i dipendenti. Tale associazione prevede cardinalità (0,N) "responsabile" → "operatore semplice" in riferimento al fatto che un responsabile possa avere più di un subordinato o che non né possa avere (un responsabile non ha superiori), cardinalità (0,1) "operatore semplice" → "responsabile" ad indicare che un operatore semplice possa avere al massimo un responsabile o non averne perché egli stesso è un responsabile. Per quanto concerne l'associazione "gestione", essa và trasferita all'entità personale poiché la l'informazione "responsabile" viene gestita lì. L'unica modifica da apportare riguarda la cardinalità che parte da personale verso reparto, essa diventa (0,1) poiché la specializzazione responsabile diventa un sottoinsieme dell'entità padre personale

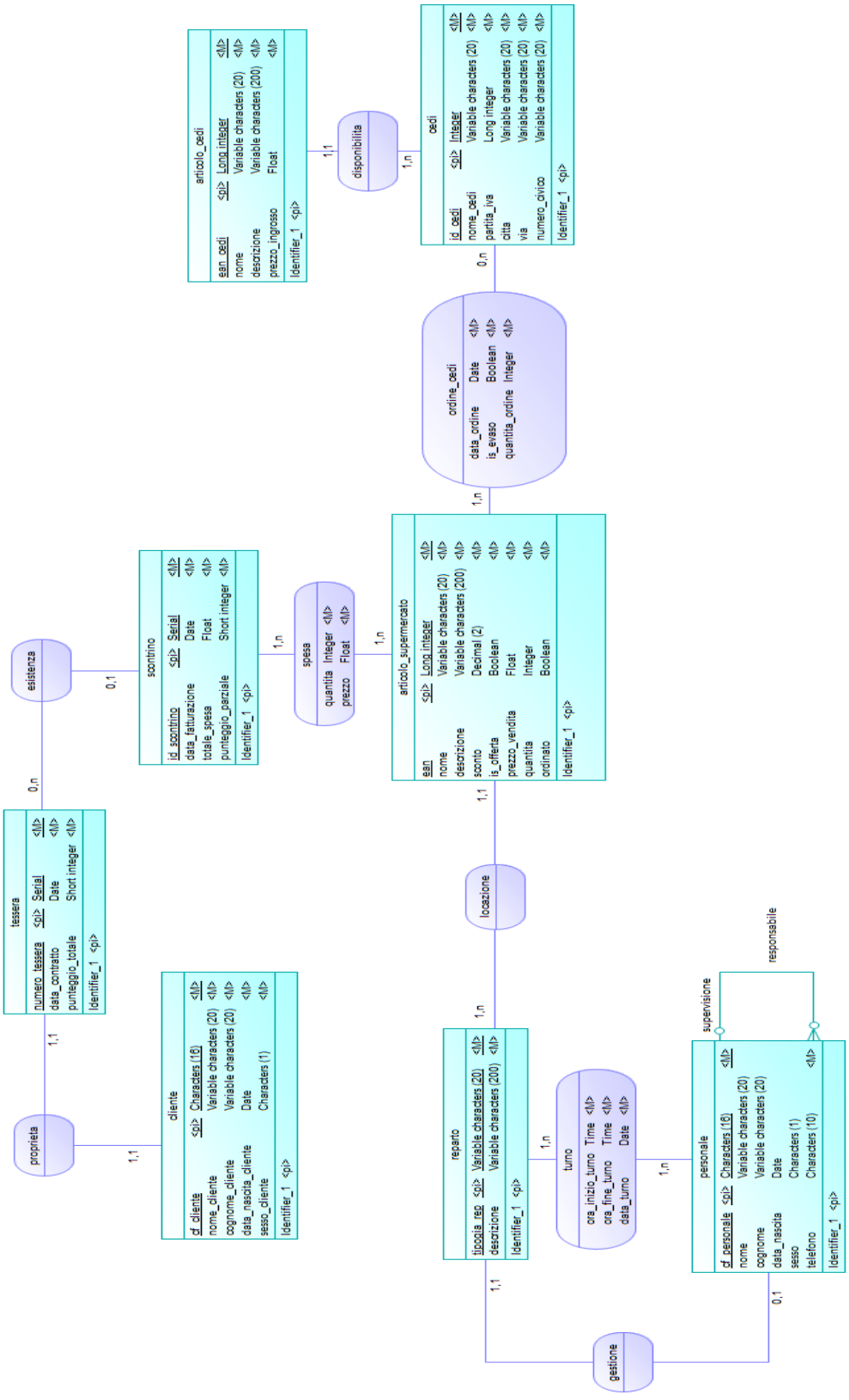
ELIMINAZIONE ATTRIBUTI COMPOSTI/MULTIVALORE

Questa fase della progettazione logica viene anticipata dallo strumento C.A.S.E. utilizzato nella realizzazione dello schema E-R(Power Designer).
Questo tool di sviluppo implementa delle soluzioni atte ad anticipare alcune fasi della progettazione logica. Ad ogni modo l'unica modifica apportata è stata quella di esplodere l'attributo composto "posizione_geografica" della relazione CEDI in Città,Via e Numero Civico

SCELTA IDENTIFICATORI PRINCIPALI

Anche questa fase è anticipata dal tool per i motivi sopra citati.

Schema ER finale



TRADUZIONE

tessera(numero_tessera , data_contratto, punteggio_totale,cliente:cliente.cf_cliente)

cedi(id_cedi,nome_cedi,partita_iva,città,via,numero_civico)

articolo_cedi(ean_cedi,nome,descrizione,prezzo_ingrosso,fornitore:cedi.id_cedi)

scontrino (id_scontrino, data_fatturazione, totale_spesa, tessera:tessera.numero_tessera, punteggio_parziale)

cliente (cf_cliente, nome_cliente, cognome_cliente, data_nascita_cliente, luogo_nascita_cliente, sesso_cliente).

articolo_supermercato(ean,nome,descrizione,sconto,is_offerta,prezzo_vendita, disponibilità,ordinato,reparto:tipologia.rep)

spesa(articolo:articolo_supermercato.ean, scontrino:scontrino.id_scontrino, quantità, prezzo),

reparto(tipologia_rep,descrizione,personale:personale.cf_personale),

personale(cf_personale ,nome, cognome, data_nascita, luogo_nascita, sesso, telefono, personale:personale.cf_personale)

turno(personale:personale.cf_personale,reparto:reparto.tipo, data_turno ,ora_inizio,ora_fine)

ordine_cedi(articolo_ordinato:articolo_supermercato.ean, cedi:cedi.id_cedi, data_ordine,quantità_ordine, is_evaso)

Vincoli aggiuntivi :

Tessera:

- Il riferimento all' entità cliente deve essere unico per ogni tupla.
- Sé si elimina un cliente si elimina la tessera, se si aggiorna il cf di un cliente si aggiorna anche il campo dedicato in tessera.

CEDI :

- L'attributo partita_iva deve essere unico per ogni tupla.
- L'attributo partita_iva deve avere 11 cifre.
- Cittò,via e numero civico sono informazioni opzionali.

Articolo_cedi :

- L' attributo prezzo_ingrosso deve contenere 3 cifre decimali.
- Sé si elimina un cedi si eliminano anche tutti gli articoli ad esso collegati, se si aggiorna l'identificativo del cedi la modifica deve essere fatta anche nell'attributo dedicato in articolo_cedi.

Articolo_supermercato:

- L'attributo sconto deve essere 0% se non c'è un offerta per quell' articolo, altrimenti deve essere maggiore del 10% e minore del 70%.
- Il prezzo di vendita deve essere maggiore o uguale a zero.
- Il prezzo di vendita del supermercato è espresso con 2 cifre decimali.
- L'attributo ordinato è attivo solo nel momento in cui non c'è disponibilità per quell'articolo.
- L'attributo reparto si aggiorna seguendo la politica di reazione di "reparto", è possibile cancellare un articolo ma esso non deve essere presente tra gli ordini.

Spesa:

- Sé cancello lo scontrino posso cancellare tutte le righe della spesa.
- Non posso cancellare un articolo se ho ancora una riga della spesa associato ad esso.
- Non posso avere nelle righe della spesa di uno scontrino due articoli uguali, devo sommare le quantità..

Reparto:

- Non posso eliminare un responsabile se esso è ancora identificato come tale nella relazione reparto
- Il riferimento all'attributo responsabile deve essere univoco per ogni tupla.

Personale:

- Il codice fiscale del dipendente deve essere diverso da quello del responsabile, sé egli è un responsabile non deve avere inizializzato l'attributo che identifica *il responsabile del dipendente.*
- Per eliminare un responsabile, devo prima eliminare il riferimento al responsabile per ogni dipendente.

Turno:

- Per cancellare un dipendente devo prima cancellare tutti i suoi turni.
- Per cancellare il reparto devo cancellare tutti i turni assegnati per quel reparto.

Ordine Cedi:

- Non posso cancellare un articolo, se ho ancora un ordine per il suddetto.
- Non posso cancellare un CEDI, se ho ancora un ordine per esso.
- La quantità di default per gli ordini deve essere di 1000, o quanto meno maggiore di 0.
- Non posso fare ordine di uno stesso articolo nello stesso giorno.

Progettazione fisica

Funzioni Randomiche

Vengono di seguito riportate tutte le funzioni usate per la generazione di informazioni randomiche, atte a dare supporto durante la fase di popolamento della base di dati.

Funzione di utilità che genera randomicamente un numero compreso tra <low> e <max>

```
create or replace function random_utili_numero(low int,max_n int) returns int as
$$
    begin
    return floor(random() * (max_n - low + 1) + low)::int;
    end;
$$ language plpgsql;
```

Funzione di utilità che genera randomicamente una data compresa tra <min> e <max>

```
create or replace function random_utili_data(min date,max date) returns date as
$$
declare
  born date ;
begin
  born = CURRENT_DATE - (random() * (CURRENT_DATE - $1))::int;
  return born;
end;
$$ language plpgsql;
```

Funzione che genera randomicamente il cognome di una persona

```
create or replace function random_persona_cognome() returns text as
$$
    declare
    chars text[] := '{Rossi,Russo,Ferrari,Esposito,Bianchi,Romano,
    Colombo,Ricci,Marino,Greco,Bruno,Gallo,Conti,De Luca,Mancini,
    Costa,Giordano,Rizzo,Lombardi,Moretti}';
    j integer;
    begin
    j = cast ((random() * (array_length(chars, 1)-1) + 1) as int);
    return chars[j];
    end;
$$ language plpgsql;
```

Funzione che genera randomicamente il nome di una persona in base al sesso

```
create or replace function random_persona_nome(sex char) returns text as
$$
    declare
    man text[] := '{Francesco,
                    Leonardo,Alessandro,Lorenzo,Mattia,Andrea,Gabriele,Riccardo,Matteo,Tommaso,Edoardo,Federico,Giuseppe,Antonio,Diego,Davide,Christian,
                    Nicolò,Giovanni,Samuele}';
    girl text[] :=  '{Sofia,Giulia,Aurora,Alice,Ginevra,Emma,Giorgia,Greta,Martina,Beatrice,Anna,Chiara,Sara,Nicole,Ludovica,Gaia,Matilde,Vittoria,Noemi,Francesca}';
    j integer;
    begin
    if sex = 'M' then
        j = cast ((random() * (array_length(man, 1)-1) + 1) as int);
        return man[j];
    else
        j = cast ((random() * (array_length(girl, 1)-1) + 1) as int);
        return girl[j];
    end if;
    end;
$$ language plpgsql;
```

Funzione che genera randomicamente la data di nascita

```
CREATE OR REPLACE FUNCTION random_utili_datanascita(min date,max date) RETURNS date AS
$$
declare
  born date ;
begin
  born = CURRENT_DATE - (random() * (CURRENT_DATE - $1))::int;
  if born > max then
    born = born - interval '9000 day';
  end if;
  return born;
end;
$$ LANGUAGE plpgsql;
```

Funzione che genera randomicamente il codice fiscale di una persona

```
create or replace function random_persona_cf(
 name_s varchar(20),
 surname varchar(20),
 birthdate varchar(20),
 location_born varchar(20),
 sex varchar(1)
 ) returns character(16) AS
 $$
 declare
            cf_name varchar(20) = upper(substring( $1 from 1 for 2));
            cf_surname varchar(3) = upper(substring( $2 from 1 for 2));
            cf_bd_d varchar(20) = upper(substring( $3 from 1 for 2));
            cf_bd_y varchar(20) = upper(substring( $3 from 9 for 2));
            cf_lb varchar(3) = upper(substring( $4 from 1 for 2));
            cf_sex int;
            temp_cf varchar;
begin
   if sex = 'M' then
       cf_sex = 0;
   else
       cf_sex = 1;
   end if;
   temp_cf = cf_name ||cast( (floor(random()*(9-1+1))+1 ) as varchar(1))|| cf_bd_d || cf_surname ||cast( (floor(random()*(9-1+1))+1 ) as
varchar(1))|| cf_sex || cf_lb || cf_bd_y || cast( (floor(random()*(9-1+1))+10 ) as varchar(1)) || upper(substring($4 from 3 for 2));
--In caso di un eventuale collisione, non siamo l' anagrafe!
--questo è necessario perchè l'agenzia delle entrate può gestire gli omocodici attraverso l'uso di un algoritmo, usando noi funzioni random
aumentando il volume c'è il
--grosso rischio di incorrere in un errore di duplicazione chiave
     if (select count(*) from cliente where cf_cliente=temp_cf) > 0 then
            temp_cf = cf_bd_d ||cast( (floor(random()*(9-1+1))+1 ) as varchar(1))|| cf_name || cf_sex ||cast( (floor(random()*(9-1+1))+1 ) as
varchar(1))|| cf_surname || cf_lb || cf_bd_y || cast( (floor(random()*(9-1+1))+10 ) as varchar(1)) || upper(substring($4 from 3 for 2));
   end if;
   return temp_cf;
end;
$$ language plpgsql;
```

Funzione che genera randomicamente una città

```
create or replace function random_utili_citta() returns varchar AS
$$
declare chars text[] := '{  Agrigento,Alessandria,Ancona,Aosta,Arezzo,
                            Ascoli Piceno,Asti,Avellino,Bari,Barletta,Andria,Trani,Belluno,Benevento,Bergamo,Biella,Bologna,Bolzano,
                            Brescia,Brindisi,Cagliari,Caltanissetta,Campobasso,Carbonia-Iglesias,Caserta,Catania,Catanzaro,Chieti,Como}';
j integer;
begin
    j = cast ((random() * (array_length(chars, 1)-1) + 1) as int);
    return chars[j];
end;
$$ language plpgsql;
```

Funzione che genera randomicamente il sesso di una persona

```
create or replace function random_persona_sesso() returns char as
$$
declare
    n integer;
begin
    n = floor(random() * (2-1+1)) +1;
    if n=1 then
     return 'M';
    else
     return 'F';
    end if;
end;
$$ language plpgsql;
```

Funzione che genera randomicamente il numero di telefono fisso/cellulare

```
create or replace function random_persona_telefono() returns varchar as
$$
declare
    prefisso text[] := '{334,338,327,352,0823}';
    c integer;
    telefono varchar;
begin
    c = floor(random() * (5-1+1))+1;
    if length(prefisso[c]) > 3 then
        telefono = prefisso[c] || cast(floor(random() * (999999-100000+1))+1 as varchar);
    else
        telefono = prefisso[c] || cast(floor(random() * (9999999-100000+1))+1 as varchar);
    end if;
    if length(telefono) <> 10 then
       Loop
       EXIT when length(telefono) = 10;
       telefono = telefono || '0';
       end loop;
    end if;
    return telefono;
end;
$$ language plpgsql;
```

Blocco funzioni per la generazione di informazioni circa l'inizio e la fine di un turno

```
create or replace function random_inizio_turno() returns time as
$$
declare
ore text[] := '{00,04,08,12,16,20}';
minuti text[] := '{00,30}';
n integer;
minu integer;
begin
 n = random_utili_numero(1,5);
 minu = random_utili_numero(1,2);
 return ore[n]||':'||minuti[minu]||':'||'00';
end;
$$ language plpgsql;

create or replace function random_fine_turno(inizio time ) returns time as
$$
begin
    return inizio + interval '4 hours';
end;
$$ language plpgsql;
```

Blocco funzioni per la generazione di informazioni relative al CEDI

```
create or replace function random_cedi_nome() returns varchar as
$$
declare
    azienda text[] := '{Crai,Sigma,Famila,Deco,Penny,Md,San Benedetto,Lete,Milka,Mulino Bianco,Barilla,Voiello,Schar,Santo Stefano,Coca Cola}';
    numero_sede int := floor(random() * (5-1+1))+1;
begin
    return azienda[(floor(random() * (15-1+1))+1)] || cast(numero_sede as varchar);
end;
$$ language plpgsql;

create or replace function random_cedi_piva() returns bigint as
$$
declare
    piva bigint := floor(random() * (99999999999-10000000000+1))+1;
begin
    return piva;
end;
$$ language plpgsql;
```

Funzione che genera randomicamente un ean

```
create or replace function random_articolo_ean() returns bigint as
$$
declare
    pre_elab varchar;
    post_elab bigint;
begin
    pre_elab =  floor(random() * (1000000000000 - 10000000 + 1) + 10000000)::varchar;
    if length(pre_elab) <> 13 then
        Loop
        EXIT when length(pre_elab) = 13;
        pre_elab = pre_elab || random_utili_numero(0,9);
        end loop;
    end if;
    post_elab = cast(pre_elab as bigint);

    return post_elab;
end;
$$ language plpgsql;
```

Funzione che genera randomicamente la descrizione per un articolo

```
create or replace function random_articolo_descrizione() returns varchar as
$$
declare
    generics text[] := '{alimento da banco , alimento da frigo, generico food, prodotto commestibile,
                         Alimento di originale Animale, Alimento di origine Vegetale, Bevande,
                         Alimento dietetico per persone sane,Alimento per persone con particolari condizioni patologiche,
                         alimento addizionato con ingredienti di origine naturale che possiedono particolari funzioni preventive e/o protettive,
                         Prodotto di 1° Gamma: conservato attraverso trattamento termico,Prodotto di 2° Gamma: congelato,
                         alimento biologico}';

   j integer;
begin
   j = cast ((random() * (array_length(generics, 1)-1) + 1) as int);
   return generics[j];
end;
$$ language plpgsql;
```

Funzione che genera randomicamente il nome di un articolo

```
create or replace function random_articolo_nome(type_food varchar) returns varchar as
$$
declare
    banco text[] := '{kitkat,pasta de cecco,pasta barilla,marmellata,carta igienica,fazzoletti,shampoo,bagno schiuma,detergente intimo,amica chips,milka,kinder fetta a latte,
                      nutella,fonzies}';
    frigo text[] := '{salame,prosciutto,mozzarella,latte,formaggio,grana padana,gorgonzola,wurstel,pasta sfoglia,yougurt,lievito,jocca,ricotta,mortadella,philadelphia}';
    bevande text[] := '{coca cola, sprite, fanta, chinotto, gatorade, powerade, red bull,vino, birra, acqua minerale, acqua gassata }';
    senza_glutine text[] := '{pane senza glutine,pizza senza glutine,pasta senza glutine}';
    generics text[] := '{tonno,angus,bistecca,melanzane,pomodori,insalata,zucchine,finocchio,scottona,vitello,girello,pollo,beef,carne kobe,spinaci,spinacine}';
    returned_name varchar;
    i integer;
begin
    CASE type_food
        WHEN 'alimento da banco' THEN
            i = cast ((random() * (array_length(banco, 1)-1) + 1) as int);
            returned_name = banco[i];
        WHEN 'alimento da frigo' THEN
            i = cast ((random() * (array_length(frigo, 1)-1) + 1) as int);
            returned_name = frigo[i];
        WHEN 'Bevande' THEN
            i = cast ((random() * (array_length(bevande, 1)-1) + 1) as int);
            returned_name = bevande[i];
        when 'Alimento per persone con particolari condizioni patologiche' then
            i = cast ((random() * (array_length(senza_glutine, 1)-1) + 1) as int);
            returned_name = senza_glutine[i];
        ELSE
            i = cast ((random() * (array_length(generics, 1)-1) + 1) as int);
            returned_name = generics[i];
 END CASE;
 return returned_name;
end;
$$ language plpgsql;
```

Blocco di funzioni genera randomicamente un prezzo (approssimato su 3 cifre) ed un reparto

```
create or replace function random_articolo_prezzo() returns decimal as
$$
declare
    pre_elab decimal;
begin
    pre_elab = random() * (9.999-0.999)+0.999;
    return trunc(pre_elab,3);
end;
$$ language plpgsql;


create or replace function random_utili_reparto() returns varchar as
$$
begin
   return (select tipologia_rep from reparto order by random() limit 1);
end;
$$ language plpgsql;
```

Script creazione tabelle

```
drop table if exists tessera cascade;
drop table if exists cliente cascade;
drop table if exists scontrino cascade;
drop table if exists cedi cascade;
drop table if exists articolo_cedi cascade;
drop table if exists personale cascade;
drop table if exists reparto cascade;
drop table if exists turno cascade;
drop table if exists articolo_supermercato cascade;
drop table if exists ordine_cedi cascade;
drop table if exists spesa cascade;

create table cliente(
    cf_cliente character(16) check (length(cf_cliente) = 16) primary key ,
    nome_cliente varchar(20) not null,
    cognome_cliente varchar(20) not null,
    data_nascita_cliente date not null,
    luogo_nascita_cliente varchar(20) not null,
    sesso_cliente char not null
);

create table tessera(
     numero_tessera serial primary key,
     data_contratto date not null,
     punteggio_totale smallint default  0,
     codice_cliente character(16) check (length(codice_cliente) = 16) not null unique
);
ALTER TABLE tessera
    ADD CONSTRAINT fk_tessera_cliente FOREIGN KEY (codice_cliente)
    REFERENCES cliente (cf_cliente)
    ON UPDATE cascade
    ON DELETE cascade;

create table cedi(
    id_cedi serial primary key,
    nome_cedi varchar(20) not null,
    partita_iva bigint unique not null,
    citta varchar(20),
    via varchar(20),
    numero_civico varchar(20)
);
```

```
create table articolo_cedi(
    ean_cedi bigint primary key,
    nome varchar(20) not null,
    descrizione varchar(200) not null,
    prezzo_ingrosso float not null check(prezzo_ingrosso > 0),
    identificativo_cedi integer not null
);
alter table articolo_cedi
    ADD CONSTRAINT fk_articoloCedi_cedi FOREIGN KEY (identificativo_cedi)
    REFERENCES cedi (id_cedi)
    ON UPDATE cascade
    ON DELETE cascade;


create table scontrino(
 id_scontrino serial primary key,
 data_fatturazione date not null,
 totale_spesa float check (totale_spesa > 0) not null,
 numero_tessera int,
 punteggio_parziale smallint
);
ALTER TABLE scontrino
    ADD CONSTRAINT fk_scontrino_tessera FOREIGN KEY (numero_tessera)
    REFERENCES tessera (numero_tessera)
    ON UPDATE cascade
    ON DELETE set null;


create table personale(
    cf_personale character(16) primary key,
    nome varchar(20) not null,
    cognome varchar(20) not null,
    data_nascita date not null,
    luogo_nascita varchar(20) not null,
    sesso char not null,
    telefono char(10) not null,
    cf_responsabile character(16) check(cf_responsabile <> cf_personale)
);
alter table personale
    add constraint fk_personale_responsabile foreign key (cf_responsabile) references personale(cf_personale)  on delete restrict on update cascade;
alter table personale
    alter constraint fk_personale_responsabile DEFERRABLE INITIALLY IMMEDIATE;
```

```
create table reparto(
    tipologia_rep varchar(20) primary key,
    descrizione varchar(200) not null,
    cf_responsabile character(16) not null unique,
);
alter table reparto
    add constraint fk_reparto_personale
    foreign key (cf_responsabile)
    references personale(cf_personale) on update cascade on delete restrict ;


create table turno(
    cf_addetto character(16) references personale (cf_personale) on update cascade on delete no action,
    reparto varchar(20) references reparto (tipologia_rep) on update cascade on delete no action,
    ora_inizio_turno time without time zone,
    ora_fine_turno time without time zone,
    data_turno date,
    primary key(cf_addetto,reparto,data_turno)
);

create table articolo_supermercato(
    ean bigint primary key check(length(cast(ean as varchar)) = 13),
    nome varchar(20) not null,
    descrizione varchar(200) not null,
    sconto decimal check ((sconto = 0.00 and is_offerta = False) OR ((sconto > 0.10 AND sconto < 0.70) and is_offerta = True)),
    is_offerta boolean default False,
    prezzo_vendita float default 0.00 check (prezzo_vendita >= 0.00),
    disponibilità integer not null check (disponibilità >= 0),
    ordinato boolean not null check ((ordinato = True and disponibilità = 0) OR (ordinato = False and disponibilità > 0)),
    reparto varchar(20) references reparto (tipologia_rep) on update cascade on delete restrict
);
```

```
create table ordine_cedi(
    ean_art bigint check(length(cast(ean_art as varchar)) = 13) references articolo_supermercato (ean) on delete no action on update  no action,
    id_cedi integer references cedi (id_cedi) on delete no action on update cascade,
    data_ordine date,
    quantita_ordine integer default 1000 check(quantita_ordine > 0),
    is_evaso boolean default False,
    primary key(ean_art,data_ordine,id_cedi)
);
CREATE OR REPLACE Function insertarticolo()
  RETURNS trigger AS $BODY$
DECLARE
    prezzo_supermercato float;
    random_n float;
    random_sconto float := random();
    prezzo_scontato float;
BEGIN
   IF NEW.is_evaso is True THEN
      random_n = random()::float;
      select prezzo_ingrosso into prezzo_supermercato from articolo_cedi where ean_cedi = new.ean_art;
      if(random_n > 0.9) then
          random_sconto = trunc(random_sconto::numeric,2);
          if(random_sconto > 0.69) then
              random_sconto = random_sconto - 0.59;
          end if;
          if(random_sconto < 0.11) then
              random_sconto = random_sconto + 0.20;
          end if;
          prezzo_scontato = prezzo_supermercato - (trunc((prezzo_supermercato::numeric),2) * random_sconto);
          update articolo_supermercato
          set reparto = random_utili_reparto(),sconto = random_sconto, is_offerta = True,
              prezzo_vendita = prezzo_scontato,disponibilità = new.quantita_ordine,ordinato=false
          where ean = new.ean_art;
      else
          update articolo_supermercato
          set reparto = random_utili_reparto(),sconto = 0.00, is_offerta = False,
              prezzo_vendita = trunc((prezzo_supermercato::numeric),2),disponibilità = new.quantita_ordine,ordinato=false
          where ean = new.ean_art;
      end if;
    END IF;
    return null;
END;
$BODY$ language plpgsql;

create trigger inserisciArticoloSupermercato
after update of is_evaso on ordine_cedi
for each row
execute procedure insertarticolo();


create table spesa(
    ean_art bigint check(length(cast(ean_art as varchar)) = 13) references articolo_supermercato (ean) on delete no action on update no action,
    numero_scontrino int references scontrino(id_scontrino) on delete cascade on update cascade,
    quantita smallint not null,
    prezzo float not null,
    primary key(ean_art,numero_scontrino)
);
```

Script generazione utenza

Sù richiesta del cliente vengono generate le 5 rispettive utenze del database, seguendo le specifiche. Dove il cliente non ha specificato operazioni vengono concessi tutti i permessi.

```
create user cedi password 'mycedi';
create user titolare_supermercato password 'mysuper';
create user dipendente password 'dip';
create user responsabile password 'admin';
create user cliente password 'mycli';


--CEDI

revoke all privileges on articolo_cedi from cedi;
revoke all privileges on cedi from cedi;
revoke all privileges on tessera from cedi;
revoke all privileges on scontrino from cedi;
revoke all privileges on cliente from cedi;
revoke all privileges on articolo_supermercato from cedi;
revoke all privileges on spesa from cedi;
revoke all privileges on reparto from cedi;
revoke all privileges on personale from cedi;
revoke all privileges on turno from cedi;
revoke all privileges on ordine_cedi from cedi;
grant insert on articolo_cedi to cedi;
grant delete on articolo_cedi to cedi;
grant select on articolo_cedi to cedi;
grant select on cedi to cedi;
grant insert on cedi to cedi;
grant update on cedi to cedi;
grant delete on cedi to cedi;
grant update on ordine_cedi to cedi;
grant select on ordine_cedi to cedi;

--titolare_supermercato

revoke all privileges on articolo_cedi from titolare_supermercato;
revoke all privileges on cedi from titolare_supermercato;
grant select on articolo_cedi to titolare_supermercato;
grant select on cedi to titolare_supermercato;


--dipendente
revoke all privileges on articolo_cedi from dipendente;
revoke all privileges on ordine_cedi from dipendente;
revoke all privileges on cedi from dipendente;
revoke all privileges on reparto from dipendente;
revoke all privileges on turno from dipendente;
grant select on turno to dipendente;
grant select on reparto to dipendente;


--responsabile

revoke all privileges on articolo_cedi from responsabile;
revoke all privileges on cedi from responsabile;
revoke all privileges on ordine_cedi from responsabile;
grant select on articolo_supermercato to responsabile;
grant select on scontrino to responsabile;

--cliente

revoke all privileges on articolo_cedi from cliente;
revoke all privileges on cedi from cliente;
revoke all privileges on tessera from cliente;
revoke all privileges on scontrino from cliente;
revoke all privileges on cliente from cliente;
revoke all privileges on articolo_supermercato from cliente;
revoke all privileges on spesa from cliente;
revoke all privileges on reparto from cliente;
revoke all privileges on personale from cliente;
revoke all privileges on turno from cliente;
revoke all privileges on ordine_cedi from cliente;
grant select on tessera to cliente;
grant select on cliente to cliente;
grant select on scontrino to cliente;
grant select on spesa to cliente;
```

Script popolamento base di dati

Script popolamento della relazione cliente

```
DO $$
declare
 nom varchar;
 cogno varchar;
 data_n date;
 sex char;
 tel varchar;
 city varchar;
BEGIN
for i in 1..10000 LOOP
sex = random_persona_sesso();
nom = random_persona_nome(sex);
cogno = random_persona_cognome();
data_n = random_utili_datanascita('1960-01-01','2000-01-01') ;
tel = random_persona_telefono();
city = random_utili_citta();
insert into cliente(cf_cliente,nome_cliente,cognome_cliente,data_nascita_cliente,luogo_nascita_cliente,sesso_cliente)
                    values (random_persona_cf(nom,cogno,cast(data_n as varchar),city,sex),
                    nom,cogno,data_n,city,sex);
end loop;
end $$;
```

Script popolamento della relazione tessera

```
insert into tessera(data_contratto,punteggio,codice_cliente)
select random_utili_data('2019-01-01','2019-12-12'),random_utili_numero(10,1000),c.cf_cliente
from cliente as c
order by random()
limit 10000;
```

Script popolamento della relazione CEDI

```
DO $$
begin
For i in 1..30000 LOOP
    insert into cedi(nome_cedi,partita_iva,citta,via,numero_civico)
    values (random_cedi_nome(),random_cedi_piva(),random_utili_citta(),'','');
end loop;
end $$;
```

Script popolamento della relazione articolo_cedi

```
DO $$
declare
    descrizione varchar;

begin
For i in 1..100 LOOP
    descrizione = random_articolo_descrizione();
    insert into articolo_cedi(ean_cedi,nome,descrizione,prezzo_ingrosso,identificativo_cedi)
    select random_articolo_ean(),random_articolo_nome(descrizione),descrizione,random_articolo_prezzo(),c.id_cedi
    from cedi as c
    order by random()
    limit 400;
end loop;
end $$;
```

Script popolamento della relazione personale

```
DO $$
declare
 nom varchar;
 cogno varchar;
 data_n date;
 sex char;
 tel varchar;
 city varchar;
 cf character(16);
 responsable_cf character(16)[];
 num integer;
BEGIN
for i in 1..300 LOOP
sex = random_persona_sesso();
nom = random_persona_nome(sex);
cogno = random_persona_cognome();
data_n = random_utili_datanascita('1960-01-01','2000-01-01') ;
tel = random_persona_telefono();
city = random_utili_citta();
cf = random_persona_cf(nom,cogno,cast(data_n as varchar),city,sex);
insert into personale(cf_personale,nome,cognome,data_nascita,luogo_nascita,sesso,telefono,cf_responsabile)
                values (cf,nom,cogno,data_n,city,sex,random_persona_telefono(),NULL);
end loop;
--settaggio responsabili
  responsable_cf = ARRAY(select cf_personale from personale order by random() limit 12);
  update personale set cf_responsabile = responsable_cf[random_utili_numero(1,12)] where cf_personale <> ALL(responsable_cf) ;
end $$;
```

Script popolamento della relazione reparto

```
DO $$
declare
    resp_cf character(16)[];
begin
 resp_cf = ARRAY(select cf_personale from personale where cf_responsabile is NULL);
insert into reparto(tipologia_rep,descrizione,cf_responsabile) values
('Reparto 1','reparto banchi',resp_cf[1]),
('Reparto 2','reparto salumi',resp_cf[2]),
('Reparto 3','reparto gastronomia',resp_cf[3]),
('Reparto 4','reparto carne',resp_cf[4]),
('Reparto 5','reparto pane',resp_cf[5]),
('Reparto 6','reparto gelato',resp_cf[6]),
('Reparto 7','reparto amici animali',resp_cf[7]),
('Reparto 8','reparto cura persona',resp_cf[8]),
('Reparto 9','reparto casa',resp_cf[9]),
('Reparto 10','reparto cura persona',resp_cf[10]),
('Reparto 11','reparto hobby',resp_cf[11]),
('Reparto 12','reparto frutta',resp_cf[12]);
end$$;
```

Script popolamento della relazione turno

```
do $$
declare
ora time ;
i integer := 0;
begin
LOOP
EXIT WHEN i = 3000;
ora = random_inizio_turno();
insert into turno(cf_addetto,reparto,ora_inizio_turno,ora_fine_turno,data_turno)
select personale.cf_personale,reparto.tipologia_rep,ora,random_fine_turno(ora),random_utili_data('2000-01-01','2019-12-12')
from personale,reparto
order by random()
limit 1;
i = i + 1;
end loop;
end $$;
```

Script popolamento della relazione articolo_supermercato ed ordine_cedi

```
insert into articolo_supermercato(ean,nome,descrizione,sconto,is_offerta,prezzo_vendita,disponibilità,ordinato,reparto)
select a.ean_cedi,a.nome,a.descrizione,0.00,False,0.00,0,True,random_utili_reparto()
from articolo_cedi as a
order by random()
limit 20000;


insert into ordine_cedi(ean_art,id_cedi,data_ordine,quantita_ordine,is_evaso)
select a.ean_cedi,a.identificativo_cedi,random_utili_data('2018-01-01','2019-01-01'),1000,False
from articolo_cedi as a,articolo_supermercato as s
where a.ean_cedi=s.ean and s.ordinato = True;


--Assumo che solo il 20% degli ordini sia ancora da evadere
update ordine_cedi set is_evaso = True where random() > 0.2;
```

Script popolamento della relazione scontrino e spesa

```
CREATE or replace procedure generazione_scontrino()
LANGUAGE plpgsql
AS $$
declare
    n_articoli integer := random_utili_numero(1,50);
    n_scontrino integer;
    temp_quantita integer;
    temp_tessera int;
    temp_punti integer;
    duplicato smallint;
    temp_disponibilita integer;
    temp_ean bigint[];
    random_n float;
BEGIN
    temp_tessera = (select numero_tessera from tessera order by random() limit 1);
    random_n = random()::float;
    if(random_n > 0.8) then
        insert into scontrino values (DEFAULT,random_utili_data('2019-01-01','2019-12-12'),0.1,temp_tessera,1);
    else
        insert into scontrino values (DEFAULT,random_utili_data('2019-01-01','2019-12-12'),0.1,NULL,0);
    end if;
    n_scontrino = (select max(id_scontrino) from scontrino);
    temp_ean = ARRAY(select ean  from articolo_supermercato where ordinato = false order by random() limit n_articoli);
    FOR i IN 1..n_articoli LOOP
            temp_quantita = random_utili_numero(1,10);
            temp_disponibilita = (select disponibilità from articolo_supermercato where ean = temp_ean[i]);
            if( temp_disponibilita <= temp_quantita) then
                RAISE NOTICE 'disponibilità insufficiente, non è possibile acquistare articolo, è stata informato il magazzino di inviare un ordine';
                update articolo_supermercato set ordinato = True where ean = temp_ean[i];
            else
                update articolo_supermercato set disponibilità = (temp_disponibilita - temp_quantita) where ean = temp_ean[i];
                insert into spesa
                select a.ean,n_scontrino,temp_quantita, temp_quantita*a.prezzo_vendita
                from articolo_supermercato as a
                where ean = temp_ean[i];
            end if;

    END LOOP;
    if(random_n > 0.8) then
            temp_punti = (select count(*) from spesa where numero_scontrino = n_scontrino);
            update scontrino set totale_spesa = trunc(((select sum(prezzo) from spesa where numero_scontrino = n_scontrino)::numeric),2),
                                punteggio_parziale = temp_punti
                where id_scontrino = n_scontrino;
            update tessera set punteggio_totale = punteggio_totale + temp_punti where numero_tessera = temp_tessera;
    else
        update scontrino set totale_spesa = trunc(((select sum(prezzo) from spesa where numero_scontrino = n_scontrino)::numeric),2)
                where id_scontrino = n_scontrino;
    end if;
END
```

Query

Query 1

SELEZIONARE REPARTO E NUMERO ARTICOLI ACQUISTATI DI QUEL REPARTO DA PARTE DEL PRIMO CLIENTE REGISTRATO NEL SUPERMERCATO.

```
create view v1 as
    select codice_cliente
    from tessera as t
    where numero_tessera = (select min(numero_tessera) from tessera);

create view v2 as
    select t.numero_tessera
    from tessera as t,v1
    where t.codice_cliente = v1.codice_cliente;

create view v3 as
    select s.id_scontrino
    from scontrino as s inner join v2 as v
    on  v.numero_tessera=s.numero_tessera;

create view v4 as
    select spesa.ean_art
    from v3 inner join spesa
    on v3.id_scontrino = spesa.numero_scontrino;

select art_sup.reparto,count(*)  as quantita
from v4 inner join articolo_supermercato as art_sup
on v4.ean_art = art_sup.ean
group by (art_sup.reparto)
order by(count(*)) asc;
```

Risultato

reparto character varying (20)	quantita bigint
Reparto 12	5
Reparto 1	6
Reparto 7	7
Reparto 10	7
Reparto 9	9
Reparto 3	9
Reparto 8	9
Reparto 5	10
Reparto 2	10
Reparto 11	10
Reparto 6	11
Reparto 4	18

Scelte di progettazione query :

- Cercare di strutturare la query su più viste di modo da renderla più leggibile e più ottimizzabile
- Cercare di scartare il prima possibile tutte le tuple non coinvolte nella nostra operazione
- Cercare quanto più possibile “push selection”

Query 2

CALCOLARE IL NUMERO DI ARTICOLI ACQUISTATI PER REPARTO A PARTIRA DA UNA DETERMINATA DATA

```
create view v1_query2 as
    select id_scontrino
    from scontrino
    where data_fatturazione < '2019-03-02';

create view v2_query2 as
    select spesa.ean_art as ean,sum(spesa.quantita) as quantita
    from v1_query2 inner join spesa
    on v1_query2.id_scontrino = spesa.numero_scontrino
    group by(spesa.ean_art);

select art_sup.reparto , sum(v2_query2.quantita) as articoli_acquistati
    from v2_query2 inner join articolo_supermercato as art_sup
    on v2_query2.ean = art_sup.ean
    group by(art_sup.reparto);
```

Risultato

reparto character varying (20)	articoli_acquistati numeric
Reparto 9	31584
Reparto 5	38733
Reparto 8	36342
Reparto 6	35982
Reparto 3	38759
Reparto 7	36018
Reparto 1	35093
Reparto 2	33800
Reparto 4	39025
Reparto 11	36678
Reparto 10	38011
Reparto 12	40228

Scelte di progettazione query :

- Cercare di strutturare la query su più viste di modo da renderla più leggibile e più ottimizzabile
- Cercare di scartare il prima possibile tutte le tuple non coinvolte nella nostra operazione

Query 3

VISUALIZZARE IL PERSONALE PRESENTE NELLA GIORNATA CON PIÙ INCASSI

```
create view v1_query3 as
    select data_fatturazione,sum(totale_spesa) as totale
    from scontrino
    group by(data_fatturazione);

create view v2_query3 as
    select data_fatturazione,totale as totale
    from v1_query3
    where totale >= ALL(select totale from v1_query3);

create view v3_query3 as
    select cf_addetto
    from  turno inner join v2_query3 --posso farlo perchè da quella vista esce solo una riga
    on v2_query3.data_fatturazione = data_turno;

select personale.nome,personale.cognome
from personale inner join v3_query3
on personale.cf_personale = v3_query3.cf_addetto;
```

Risultato

nome character varying (20)	cognome character varying (20)
Giovanni	Mancini

Trigger

Il trigger nasce da una specifica della realtà analizzata “il supermercato non può attingere autonomamente agli articoli dei cedi”, il meccanismo di rifornimento articoli da parte del supermercato è articolato in più fasi:

1) Generazione ordine da parte del supermercato per un articolo al CEDI di interesse
2) Attesa evasione ordine
3) Ordine evaso dal CEDI
4) Fine rifornimento

Per poter implementare questo meccanismo si è pensato di creare un trigger che osserva la relazione ordine_cedi e viene scatenato ad ogni update di tupla, controlla lo stato di “is_evaso” e sé risulta true carica le informazioni inerenti all’articolo che ha generato l’evento in “articolo_supermercato”, non tutte le informazioni poiché per scelta progettuale, in fase di popolamento vengono precaricati ean,nome,descrizione su “articolo_supermercato” per permettere al dbms di generare già un indice sulla chiave. Chiaramente in partenza articolo_supermercato ed ordine_cedi sono cloni (vedere popolamento).

```
CREATE OR REPLACE Function insertarticolo()
  RETURNS trigger AS $BODY$
DECLARE
    prezzo_supermercato float;
    random_n float;
    random_sconto float := random();
    prezzo_scontato float;
BEGIN
   IF NEW.is_evaso is True THEN
      random_n = random()::float;
      select prezzo_ingrosso into prezzo_supermercato from articolo_cedi where ean_cedi = new.ean_art;
      if(random_n > 0.9) then
          random_sconto = trunc(random_sconto::numeric,2);
          if(random_sconto > 0.69) then
              random_sconto = random_sconto - 0.59;
          end if;
          if(random_sconto < 0.11) then
              random_sconto = random_sconto + 0.20;
          end if;
          prezzo_scontato = prezzo_supermercato - (trunc((prezzo_supermercato::numeric),2) * random_sconto);
         update articolo_supermercato
         set reparto = random_utili_reparto(),sconto = random_sconto, is_offerta = True,
            prezzo_vendita = prezzo_scontato,disponibilità = new.quantita_ordine,ordinato=false
          where ean = new.ean_art;
      else
          update articolo_supermercato
          set reparto = random_utili_reparto(),sconto = 0.00, is_offerta = False,
            prezzo_vendita = trunc((prezzo_supermercato::numeric),2),disponibilità = new.quantita_ordine,ordinato=false
          where ean = new.ean_art;
      end if;
    END IF;
    return null;
END;
$BODY$ language plpgsql;

create trigger inserisciArticoloSupermercato
after update of is_evaso on ordine_cedi
for each row
execute procedure insertarticolo();
```

Tuning PostGresql

Visto il carico applicativo della base di dati ed le operazioni su di essa svolte, si progetta di ottimizzare la configurazione di Postgresql attraverso la modifica di alcuni paramentri nel file di configurazione ***postgresql.conf***

Configurazione Server :

- **intel i7-5550u**
- **SSD 256GB**
- **12GB Ram**
- **OS Windows**
- **Numero Connesioni al DB : al più 20**
- **Versione DB : 11**

Si decide di andare a focalizzarsi sui seguenti parametri

max_connections, *numero connessioni massimo accettato*

shared_buffers, *rappresenta quanta RAM il database può usare per il buffer*

maintenance_work_mem, rappresenta quanta memoria và destinata per le operazioni di manutenzione

wal_buffers, *rappresenta quanta memoria condivisa usare per i dati WAL ancora non scritti sul disco*

random_page_cost, rappresenta il costo stimato per recuperare una pagina non sequenzialmente

work_mem, Specifica la quantità di memoria che deve essere utilizzata dalle operazioni di ordinamento interno e dalle tabelle hash

min_wal_size, Finché l'utilizzo del disco WAL rimane al di sotto di questa impostazione, i vecchi file WAL vengono sempre riciclati per l'utilizzo futuro su un checkpoint, anziché essere rimossi.

max_wal_size, rappresenta la dimensione massima di crescita tra due WAL checkpoints

effective_io_concurrency, Imposta il numero di operazioni di I / O su disco simultanee che PostgreSQL si aspetta possano essere eseguite simultaneamente.

PRIMA

max_connections = 100
shared_buffers = 128MB
maintenance_work_mem= 64MB
wal_buffers = -1 (automatico)
random_page_cost = 4.0
work_mem = 4MB
min_wal_size = 80MB
max_wal_size = 1GB
effective_io_concurrency = 0

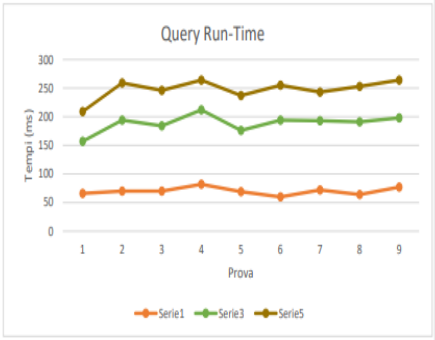
DOPO

max_connections = 20
shared_buffers = 512MB
maintenance_work_mem= 768MB
wal_buffers = 16MB
random_page_cost = 1.1
work_mem = 100MB
min_wal_size = 2GB
max_wal_size = 1GB
effective_io_concurrency = 100MB

Statistiche

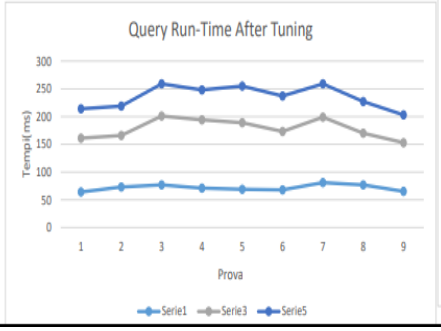
Tempi di attesa prima del tuning (ms)

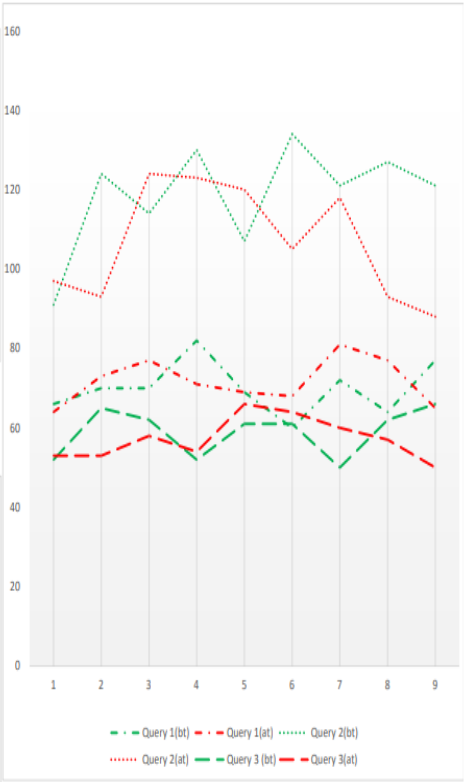
query 1	query 2	query 3
66	91	52
70	124	65
70	114	62
82	130	52
69	107	61
60	134	61
72	121	50
64	127	62
77	121	66



Tempi di attesa dopo il tuning (ms)

query 1	query 2	query 3
64	97	53
73	93	53
77	124	58
71	123	54
69	120	66
68	105	64
81	118	60
77	93	57
65	88	50





Primo applicativo java:

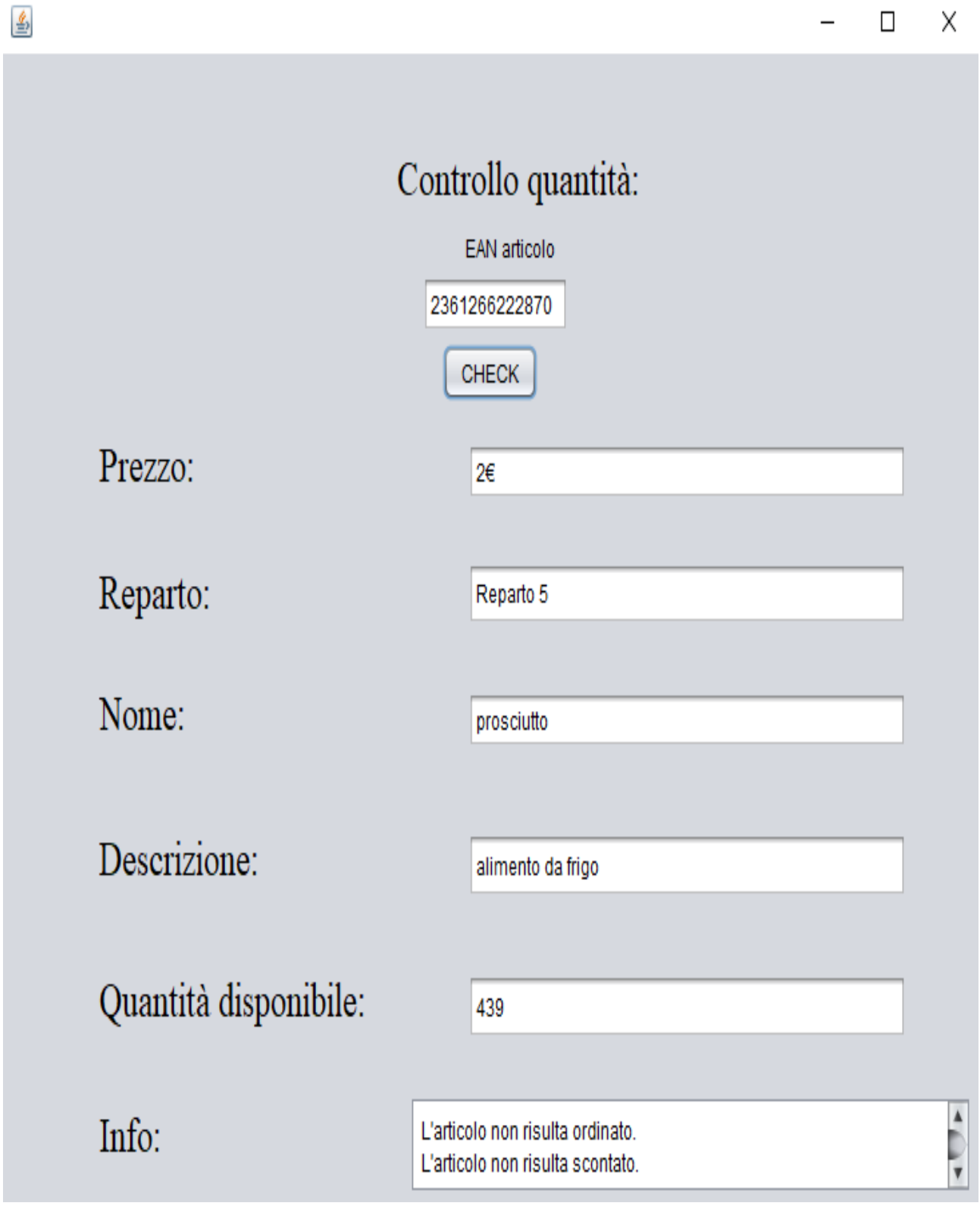
Descrizione:

Si vuole realizzare un applicativo java che dato in ingresso un codice EAN è in grado di restituire informazioni riguardanti l'articolo corrispondete al codice, in particolare:

il prezzo, la descrizione, la disponibilità, il reparto, il nome e altre info utili.

Codice java:

```
{
   BigInteger id_letto = new BigInteger(idText.getText());
   String url = "jdbc:postgresql://localhost:5432/project";
   String username = "responsabile";
   String pass = "admin";
   String query = "Select * from articolo_supermercato where ean = \'" + id_letto + "\'";
   String a = new String();
   String b= new String ();
    try {
                Connection conn = DriverManager.getConnection(url, username, pass);
                Statement stm = conn.createStatement();
                ResultSet rst = stm.executeQuery(query);
                while (rst.next()) {
                            quantitaText.setText(rst.getString("disponibilità") + " ");
                            nomeField.setText(rst.getString("nome") + " ");
                            descrizioneField.setText(rst.getString("descrizione") + " ");
                            repartoField.setText(rst.getString("reparto")+" ");
                            prezzoField.setText(rst.getInt("prezzo_vendita")+"€");
                            if (rst.getBoolean("ordinato")== true){
                              a= ("L'articolo  risulta ordinato.");
                            } else {
                              a= ("L'articolo non risulta ordinato.");
                            }
                            if(rst.getBoolean ("is_offerta")==true){
                                b=("L'articolo  risulta scontato del "+ rst.getInt("sconto");
                            }else{
                              b=("L'articolo non risulta scontato.");
                            }
                           infoArea.setText(a+"\n"+ b);
                     }
    conn.close();
    } catch (java.sql.SQLException ex) {
       System.out.println(ex.getMessage());
    }
```


Controllo quantità:
EAN articolo
2361266222870
CHECK
Prezzo:
2€
Reparto:
Reparto 5
Nome:
prosciutto
Descrizione:
alimento da frigo
Quantità disponibile:
439
Info:
L'articolo non risulta ordinato.
L'articolo non risulta scontato.

Secondo applicativo java:

Descrizione:

Si vuole realizzare un applicativo che prendendo in input una data di inizio e una data di fine periodo è in grado di darci in output informazioni riguardanti gli scontrini nel periodo tra le due date, in particolare:

Somma di tutti gli scontrini, numero di scontrino e media per scontrino.

```
String url = "jdbc:postgresql://localhost:5432/project";
String username = "responsabile";
String pass = "admin";
  try {
       DateFormat df = new SimpleDateFormat("dd/MM/yyyy");
       String date1InString = data1Field.getText();
       String date2InString = data2Field.getText();
       java.util.Date date1 =  df.parse(date1InString);
       java.util.Date date2 =  df.parse(date2InString);
      String query = "Select SUM(totale_spesa) as tot, count(*) as cont from scontrino where data_fatturazione between \'" +
      df.format(date1)+ "\'and \'" +df.format(date2)+"\'";
      System.out.println(query);
      Connection conn = DriverManager.getConnection(url, username, pass);
      Statement stm = conn.createStatement();
      ResultSet rst = stm.executeQuery(query);
      while (rst.next()) {
                  Double a = new Double(rst.getString("tot"));
                  Double b= new Double(rst.getString("cont"));
                  String a1= String.format("%.2f", a);
                  totField.setText(a1+ "€");
                  nField.setText(rst.getString("cont"));
                  Double c= new Double(a/b);
                  String c1= String.format("%.2f", c);
                  String d=  new String();
                  mediaField.setText(c1+"€");
       }
      conn.close();
} catch (java.sql.SQLException ex) {
   System.out.println(ex.getMessage());
} catch (ParseException e) {
   e.printStackTrace();
}
```


Totale importo scontrini:
Da:
fino a:
10/04/2019
14/04/2019
Formata data: gg/mm/aaaa
check
Tot. importo scontrini:
30852,36€
N. scontrini:
31
Media scontrino:
995,24€