Report Database

Federico Diotallevi

2023/2024

Contents

1 Analisi dei requisiti

Si vuole realizzare un database a supporto dell'automazione della gestione di un esercizio commerciale attivo nella ristorazione (bar/ristoranti). Il databse dovrà quindi memorizzare le informazioni riguardanti i tavoli con le relative comande, prodotti e scontrini. I dipendenti del bar potrenno visualizzare le comande associate ai tavoli e compilare gli scontrini, mentre l'amministratore potrà visualizzare i dati relativi al fatturato.

1.1 Intervista

Un primo testo ottenuto dall'intervista è il seguente:

Si vuole tenere traccia delle comande e della gestione del magazzino di un bar/ristorante memorizzandone i prodotti e le comande. Ogni prodotto è caratterizzato da un nome, una descrizione, un prezzo di vendita o, eventualmente, un prezzo di acquisto per la gestione del magazzino.

Al momento della creazione di una comanda, vengono registrati i prodotti associati, la quantità di ciascun prodotto e il tavolo a cui la comanda è destinata. Ai tavoli è possibile attribuire un nome scelto al momento (ad esempio "Tavolo Sala 1" o "Tavolo Pippo").

Il sistema permette di redigere uno o più scontrini per ogni tavolo, consentendo di dividere la spesa tra più clienti se necessario. Gli scontrini riportano il prezzo di ogni prodotto e il totale della spesa. Si tenga presente che uno scontrino non si riferisce necessariamente ad un tavolo, un ordine può anche essere fettuato al tavolo e in quel caso necessita il pagamento al momento dell'ordine.

Il database mantiene uno storico di tutti i pagamenti riscossi, consentendo la generazione di grafici per monitorare il fatturato sia annuale che mensile, o l'andamento delle vendite di ciascun prodotto.

Il bar/ristorante dispone di uno o più amministratori con la possibilità di aggiungere nuovi utenti e visualizzare i dati sul fatturato. Gli utenti possono essere suddivisi nei seguenti ruoli:

- Camerieri: possono redigere le comande per i tavoli.
- Baristi/Cuochi: possono visualizzare le comande in arrivo e segnarle come pronte una volta preparate.
- Magazzinieri: possono stilare la lista dei prodotti presenti in magazzino, annotare i prodotti da ricomprare e gestire il loro prezzo di acquisto.

Ogni comanda viene registrata con la data e l'ora in cui è stata creata e può essere modificata solo da utenti autorizzati fino a quando non viene chiusa con l'emissione di uno scontrino.

Gli amministratori possono anche visualizzare report dettagliati che comprendono:

- Elenco dei prodotti più venduti.
- Analisi del fatturato su base giornaliera, settimanale, mensile e annuale.
- Analisi dei servizi effettuati: persone e tavoli serviti

In sintesi, il compito dello staff del bar/ristorante è assicurarsi che le comande vengano redatte e gestite correttamente, che i prodotti siano sempre disponibili in magazzino, e che le vendite e il fatturato vengano monitorati e analizzati per migliorare la gestione complessiva del locale.

1.2 Estrazione dei concetti principali

Termine	Breve descrizione
Prodotto	Sono gli articoli venduti dall'esercizio, ognuno con un proprio costo
Tavolo	Oggetto a cui è possibile attribuire un nome e una data, rappresenta il gruppo dei clienti a cui si riferiscono comande e scontrini
Comanda	Una comanda è un ordine effettuato in un qualsiasi momento dai clienti contenente uno o più prodotti
Scontrino	Oggetto in cui viene riportata una spesa effettuata da un tavolo, possono esserci uno o più scontrini associati ad un tavolo
Cameriere	Colui che compila comande e scontrini
Preparatore	Colui che segna le comande come completate
Magazziniere	Colui che segna la disponibilità dei prodotti nel magazzino
Amministrator	Colui che gestisce visualizza dati finanziari e modifica i dipendenti

1.3 Elaborazione dei concetti principali

A seguito della lettura e comprensione dei requisiti si redige un testo che ne riassuma tutti i concetti, con particolare attenzione a quelli principali ed eliminando le ambiguità:

> Per ogni ***prodotto***[1] nel menù vengono memorizzati *nome, descrizione, prezzo di vendita*, inoltre ogni prodotto possiede un *codice* univoco assegnatogli quando viene aggiunto al menù. Se il prodotto è un semplice prodotto da magazzino allora deve riportare anche il proprio *costo*. Per ogni prodotto è possibile memorizzare una *ricetta* composta da *ingredienti* e *quantità* necessarie per la preparazione. Ogni ***comanda*** possiede *data e ora*, può essere in preparazione o completata ed ha un *codice univoco*, le comande possono riguardare *ordini al banco* oppure *ordini al tavolo*. Ad ogni comanda sono associati uno o più prodotti insieme alle *quantità* richieste. Deve essere possibile verificare che tutti i prodotti appartenenti ad una comanda sono stati pagati. Ogni prodotto, che sia ordinato al tavolo o al banco, può presentare una o più *richieste aggiuntive* del cliente, che potrebbero comportare un *costo aggiuntivo*. Un ***tavolo*** è caratterizzato da: *un codice univoco, un nome (scelto al momento della creazione), un numero di persone* sedute al tavolo e *una data di creazione*.
>
> Uno ***scontrino*** rappresenta la somma dei costi di uno o più prodotti presenti in una comanda associata ad una tavolo, l'insieme degli scontrini erogati in un tavolo rappresenta la spesa totale. Ogni scontrino ha un *codice univoco* indipendente dal tavolo, *una data e un orario*, *una spesa* e *una modalità di pagamento*[2], per ogni scontrino deve essere possibile elencare i prodotti che lo componevano e le relative quantità. Un ***ordine di magazzino*** è caratterizzato da *un codice univoco e una data e orario*. Per ogni ordine si tiene conto del *dipendente* che lo ha compilato e dei prodotti che lo compongono. Per ogni prodotto è possibile scegliere un ***fornitore*** (identificato attraverso un *nome*) avente ognuno un *costo* potenzialmente differente per ogni prodotto. Lo staff è composto da:

[1] Un prodotto può essenzialmente essere un prodotto preparato (es: un drink), un prodotto da magazzino (es: farina) oppure entrambi (es: bottiglia di vino).

[2] i.e. contante, bancomat...

- ***amministratori***: hanno la possibilità di aggiungere membri dello staff e visualizzare info sulle vendite
- ***preparatori***: preparano le comande e le segnano come completate e aggiornano i prodotti in menu.
- ***magazzinieri***: ordinano prodotti del magazzino e ne segnano la disponibilità.
- ***camerieri***: aggiungono tavoli e compilano comande e scontrini.

Segue un elenco delle principali azioni richieste:

1. *Aggiornare prodotti*
2. *Aggiungere tavoli*
3. *Compilare comande al tavolo*
4. *Mostrare le comande filtrate per stato e numerate per giornata*
5. *Mostare il numero di tavoli e clienti serviti in un dato periodo*
6. *Visualizzare prodotti e relative variazioni appartenenti ad una comanda*
7. *Ricerca prodotti per similitudine di nome, categoria e sottocategoria*
8. *Visualizzare prodotti non pagati in un tavolo*
9. *Compilare scontrini di un tavolo*
10. *Visualizzare i guadagni in un dato periodo*
11. *Compilare ordini del magazzino*
12. *Visualizzare le spese per i rifornimenti in un dato periodo*
13. *Visualizzare le prenotazioni per un dato giorno*
14. *Visualizzare la ricetta di un prodotto*
15. *Aggiornare staff del locale*

2 Progettazione concettuale

2.1 Schema Scheletro

L'entità ***prodotto*** viene identificata tramite un codice univoco, dall'analisi si evince che ne esistono due tipologie:

- prodotti da magazzino
- prodotti in menu

queste tipologie sono specializzazioni *sovrapponibili* dell'entità prodotto.

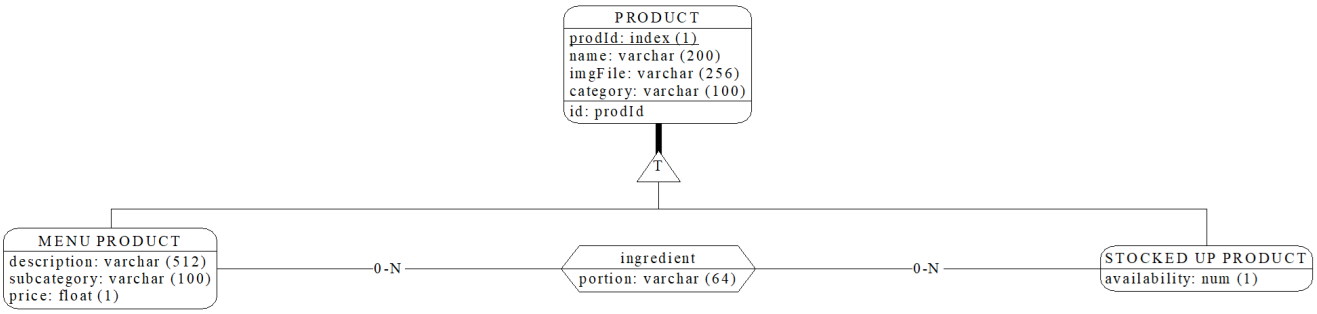


Figure 1: Schema parziale sulle specializzazioni di prodotto

Ogni comanda si riferisce ad un ***tavolo***, se ordinata da esso, altrimenti allo ***scontrino*** tramite cui è stata pagata, anch'essi identificati tramite un *codice univoco*. Ad ogni tavolo è possibile assegnare un *nome* e un *numero di clienti* seduti.

Uno stesso ***prodotto*** può presentarsi nella stessa ***comanda*** con ***variazioni*** e quantità differenti (si pensi ad esempio ad una stessa bevanda, nello stesso tavolo, ordinata allo stesso momento, una con ghiaccio, una senza). E' quindi necessario reificare le entità ***prodotto al tavolo*** e ***prodotto al banco***, identificate tramite un *id univoci* e riportanti la **quantità ordinata.

Ad ogni prodotto al tavolo è possibile associare una variazione ed una quantità associata alla comanda. Ogni prodotto del tavolo può essere presente una sola volta nella stessa comanda, perciò è possibile creare un'associazione con la comanda e l'attributo *quantità*. Tale associazione esprime il vincolo per il quale le stesse istanze di ordine al tavolo e prodotto non possono presentarsi con molteplici quantità.

I prodotti al banco sono invece prodotti ordinati senza riferimento ad un tavolo, non hanno perciò associazioni con esso, inoltre non hanno bisogno di un attributo *quantità* nell'associazione con comanda, in quanto per un prodotto al banco la quantità in comanda corrisponde alla quantità ordinata.

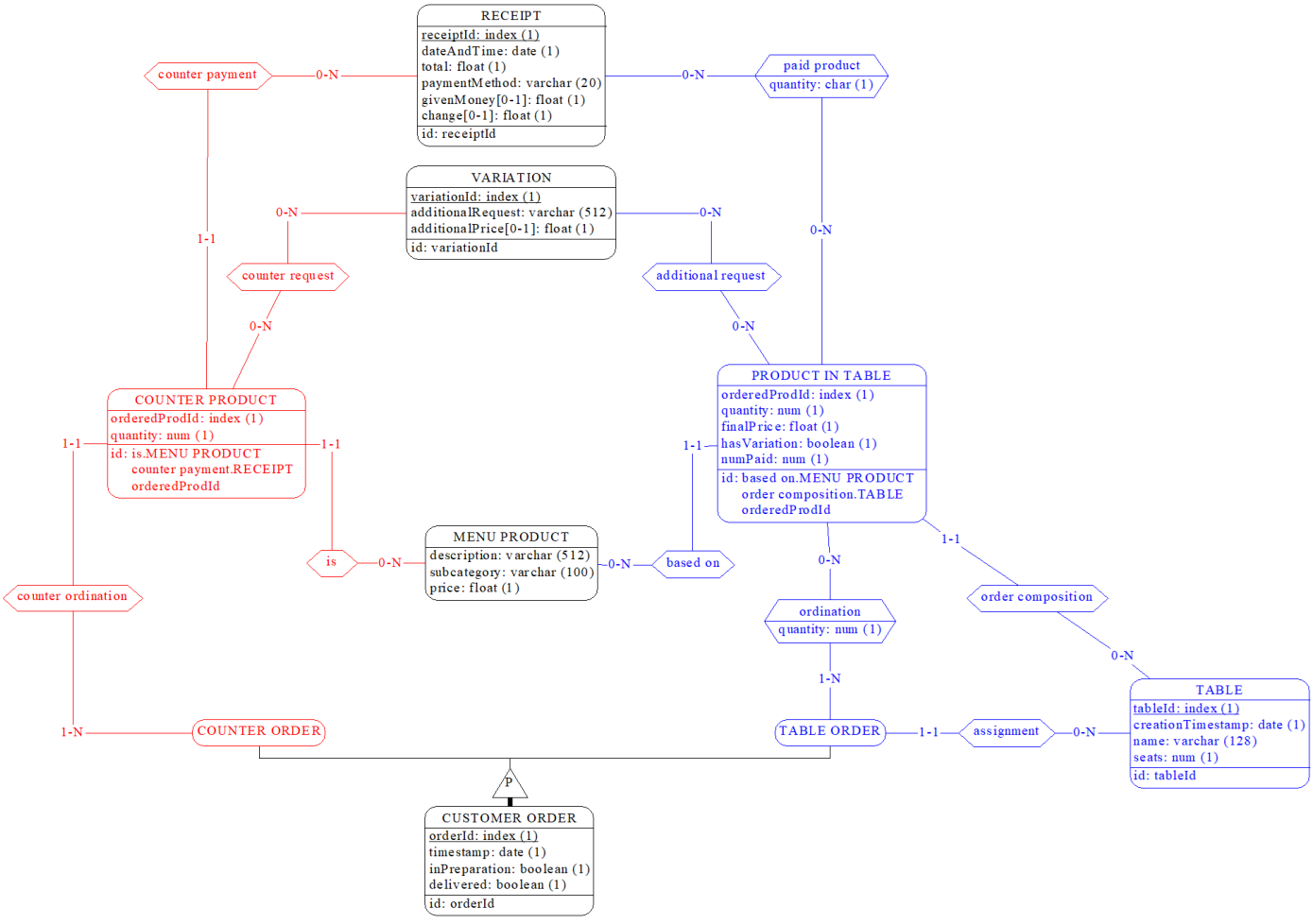


Figure 2: Schema parziale sulle relazioni tra tavoli, comande e prodotti e veriazioni: in blu ordni al tavolo, in rosso ordini al banco

Le entità ***cameriere, preparatore, amministratore*** e ***magazziniere*** non sono altro che specializzazioni dell'entità ***dipendente***, ognuna identificata tramite *id univoco*. Di ogni dipendente si vuole mantenere nel database il ruolo all'interno del locale per conoscere le operazioni che possono effettuare (ad esempio la visualizzazione dei dati sul fatturato è un'operazione che spetta esclusivamente ad un amministratore). In particolare per ogni comanda e per ogni ordine di magazzino si vuole tenere nel database rispettivamente il cameriere e l'addetto al magazzino che li hanno compilati.

Uno stesso ***prodotto*** può essere incluso in uno stesso ***ordine di magazzino***, identificato tramite un *codice univoco*, più volte, ordinato ad esempio da ***fornitori*** diversi, ognuno con un *prezzo* potenzialmente diverso per lo stesso prodotto. E' quindi necessario reificare l'entità di ***prodotto in ordine***, avente come attributi la *quantità ordinata, un'id univoco nell'ordine* e un'associazione al fornitore scelto.

Dell'ordine di magazzino è inoltre necessario tenere traccia nel database della *data e ora* dell'ordine, se l'ordine è già stato inviato e il dipendente che lo ha compilato.

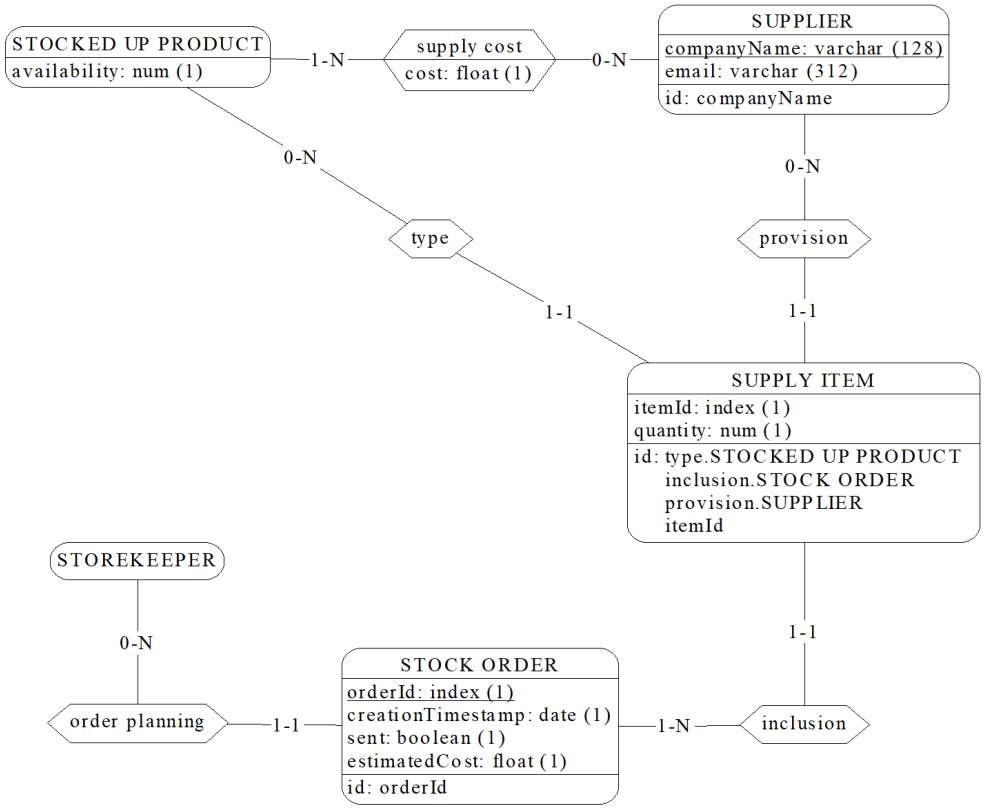


Figure 3: Schema parziale sugli ordini dei prodotti

Ogni ***prodotto al tavolo*** può essere pagato in più scontrini in quantità differenti e con *modalità di pagamento differenti*, tuttavia uno stesso prodotto al tavolo non può essere presente in più scontrini, è quindi sufficiente un'associazione tra prodotto al tavolo e scontrino, con l'attributo *quantità pagata*. Tale associazione esprime il vincolo per il quale le stesse istanze di prodotto al tavolo non possono presentarsi con moltepliche quantità pagate in uno stesso scontrino

Ogni ***prodotto al banco*** è un prodotto ordinato senza riferimento ad un tavolo, è identificato attraverso un *id* univoco all'interno dello scontrino cui si riferisce. Ogni prodotto al banco deve essere interamente pagato al momento dell'ordine, perciò è sufficiente un'associazione tra prodotto al banco e scontrino, senza necessità di un attributo quantità pagata.

Ogni ***scontrino*** è identificato da un *codice univoco*, ha una *data e un orario*, una *spesa* e una *modalità di pagamento*.

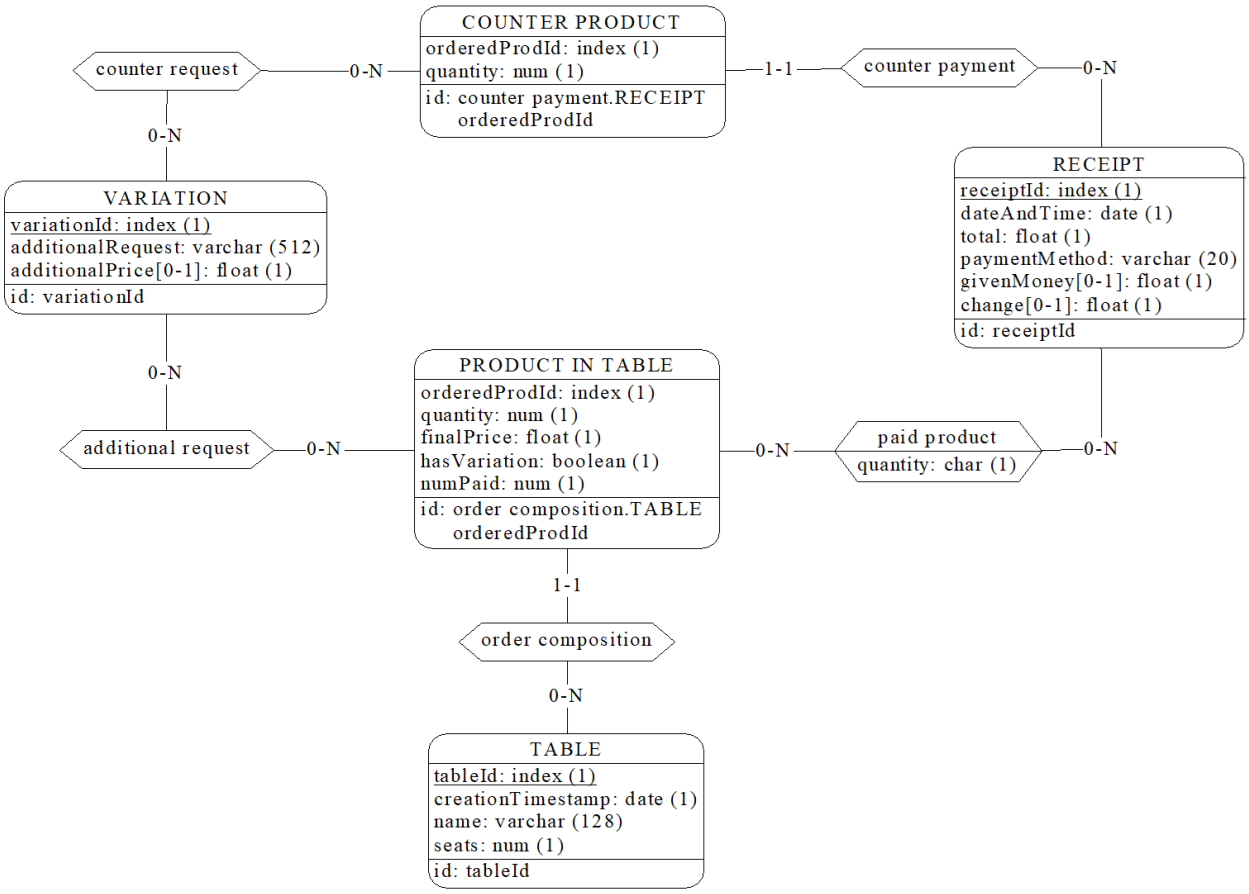
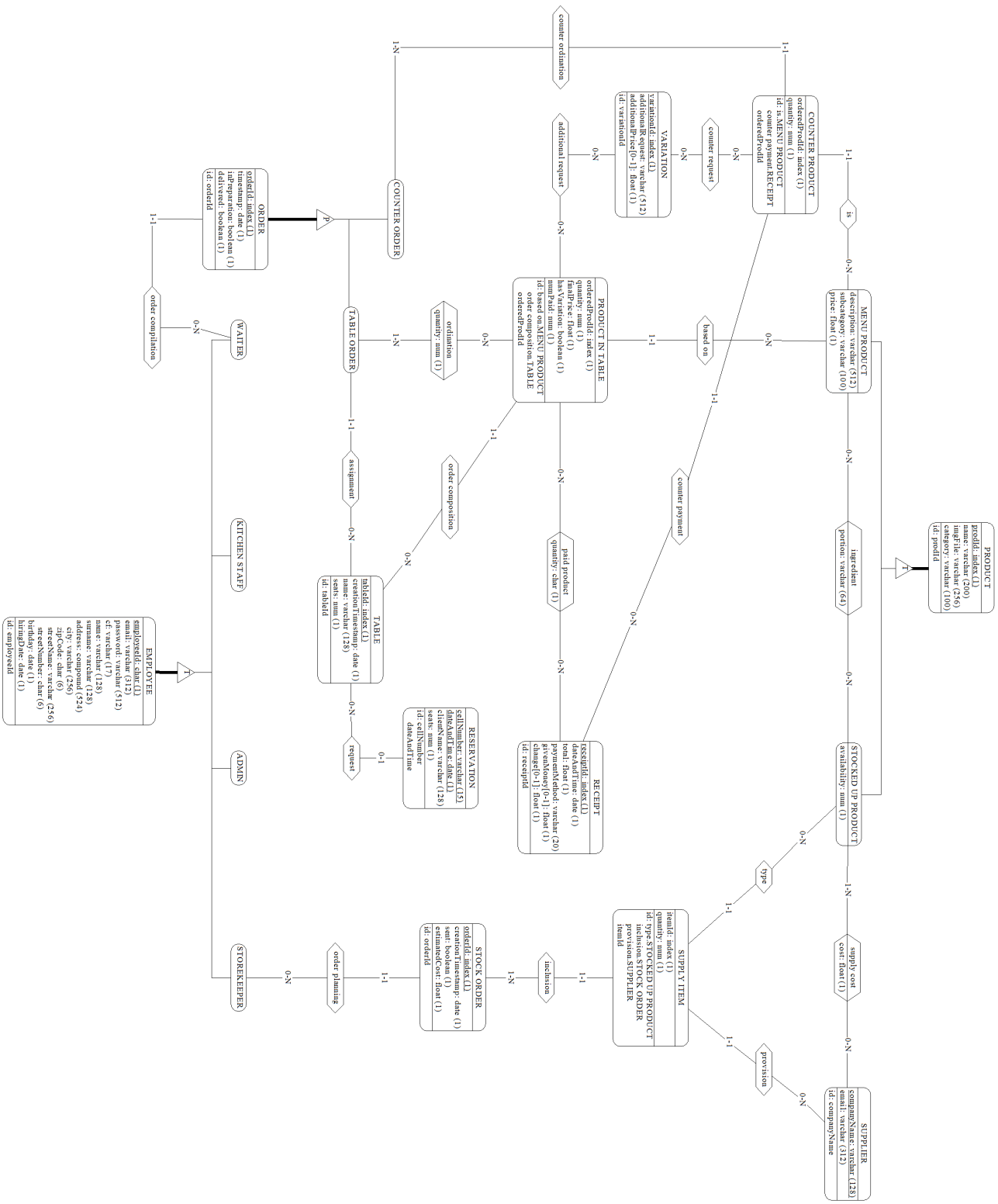


Figure 4: Schema parziale sulle relazioni tra prodotti al tavolo e scontrini

2.2 Schema ER completo

Di seguito si allega lo schema ER nel suo complesso, costruito con DB-Main.



3 Progettazione logica

3.1 Stima del volume di dati

Di seguito la stima dei volumi richiesti per entità e relazioni:

Concetto	Costrutto	Volume
Menu Product	E	80
Stocked Up Product	E	250
Ingredient	R	320[3]
Variation	E	500
counter request	E	750.000
additional request	R	750.000[4]

Concetto	Costrutto	Volume
Table	E	150.000
Product In Table	E	750.000[5]
Counter Product	E	750.000
counter ordination	E	750.000
Table Order	E	300.000[6]
Counter Order	E	300.000
ordination	R	750.000
based on	R	750.000
Receipt	E	600.000[7]
paid product	E	1.000.000[8]
counter payment	R	750.000
order composition	R	750.000
assignment	R	300.000
Reservation	E	5.000
request	R	5.000

Concetto	Costrutto	Volume
Supplier	E	10
supply cost	R	750
provision	R	10.000
type	R	10.000
Supply Item	E	10.000

[3] Considerando una media di 4 ingredienti per prodotto in menu
[4] Considerando circa 5 diversi prodotti ordinati in un tavolo
[5] Considerando circa 2 comande per tavolo
[6] Considerando che ogni prodotto ordinato presenta una variazione
[7] Considerando circa 2 scontrini per tavolo + 1 scontrino per ogni ordine al banco
[8] Considerando che per ogni prodotto al tavolo deve esistere almeno un corrispondente prodotto pagato e che alcuni potrebbero essere pagati in scontrini diversi

Concetto	Costrutto	Volume
inclusion	R	10.000
Stock Order	E	1.000

Concetto	Costrutto	Volume
Waiter	E	25
order compilation	R	300.000
Kitchen Staff	E	25
Admin	E	5
Storekeeper	E	10
order planning	R	1.000

3.2 Descrizione delle operazioni principali e stima della loro frequenza

Le operazioni da effettuare sono quelle precedentemente elencate nella fase di analisi. Di seguito vengono elencate insieme alla relativa frequenza:

Numero operazione	Operazione	Frequenza
1.	Aggiornare prodotti	50 all'anno
2.	Aggiungere tavoli	200 a settimana
3.	Compilare comande al tavolo	800 a settimana
4.	Mostrare comande filtrate per stato e numerate per giornata	30 al giorno
5.	Mostare numero tavoli e clienti serviti in un dato periodo	2 al giorno
6.	Visualizzare prodotti e variazioni nella comanda di un tavolo	100 al giorno
7.	Ricerca prodotti per pattern nome, categoria e sottocategoria	4000 a settimana
8.	Visualizzare prodotti non pagati in un tavolo	2400 a settimana
9.	Compilare scontrini di un tavolo	600 a settimana
10.	Visualizzare i guadagni in un dato periodo	10 al mese
11.	Compilare ordini del magazzino	3 a settimana
12.	Visualizzare spese per rifornimenti in un dato periodo	10 al mese
13.	Visualizzare le prenotazioni per un dato giorno	100 a settimana
14.	Visualizzare la ricetta di un prodotto	20 al giorno
15.	Aggiornare staff del locale	5 all'anno

3.3 Schemi di navigazione e tabelle degli accessi

Sono riportate in seguito le tabelle degli accessi delle operazioni elencate sopra. Per il calcolo dei costi le operazioni in scrittura sono considerate con peso doppio rispetto a quelle in lettura.

3.3.1 Operazione 1 - Aggiornare prodotti

L'aggiornamento di un *prodotto* comporta anche l'aggiornamento della ricetta, possiamo considerare che in media ogni prodotto ha 3 *ingredienti* e ogni *prodotto di magazzino* è *ingrediente* di 3 *prodotti da menù*.

Concetto	Costrutto	Accessi	Tipo
Product	E	1	L
Product	E	1	S
Ingredient	R	3	L
Ingredient	R	3	S
	Totale:	4S + 4L → 500 all'anno	

3.3.2 Operazione 2 - Aggiungere Tavoli

Concetto	Costrutto	Accessi	Tipo
Table	E	1	S
	Totale:	1S → 400 a settimana	

3.3.3 Operazione 3 - Compilare Comande al tavolo

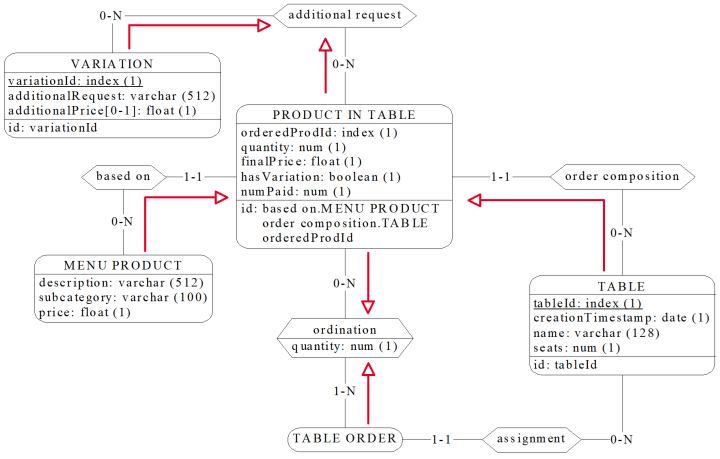


Figure 5: Schema di navigazione per l'operazione 3

La compilazione delle *comande* richiede delle ricerche di *prodotti di menu*, che in media sono 3 a comanda, successivamente occorrono degli accessi in lettura alle *variazioni*, degli aggiornamenti nei

prodotti al tavolo e degli accessi in scrittura nelle *richieste aggiuntive*. Consideriamo inoltre 1 sola scrittura di *order composition*, in quanto gli altri prodotti potrebbero già essere scritti.

Concetto	Costrutto	Accessi	Tipo
Menu Product	E	3	L
based on	R	3	L
Variation	E	3	L
additional request	R	3	S
Product In Table	E	3	L
Product In Table	E	3	S
order composition	R	3	L
order composition	R	1	S
ordination	R	3	S
Table Order	E	1	S
assignment	R	1	S
	Totale:	12S + 15L → 31.200 a settimana	

3.3.4 Operazione 4 - Mostrare le comande filtrare per stato e numerate per giornata

Considerando 60 comande per giornata:

Concetto	Costrutto	Accessi	Tipo
Table Order	E	60	L
	Totale:	60L → 1800 al giorno	

3.3.5 Operazione 5 - Mostrare il numero di tavoli e clienti serviti in un dato periodo

Considerando 180 tavoli serviti a settimana avremmo:

Concetto	Costrutto	Accessi	Tipo
Table	E	180	L
	Totale:	180L → 360 al giorno	

3.3.6 Operazione 6 - Visualizzare prodotti e relative variazioni nella comanda di un tavolo

Per visualizzare i prodotti di una comanda e le relative variazioni, è necessario accedere alla comanda, ai prodotti ordinati (in media 3 per comanda) e alle eventuali variazioni associate.

Concetto	Costrutto	Accessi	Tipo
Customer Order	E	1	L
ordination	R	3	L
Product In Table	E	3	L
based on	R	3	L

Concetto	Costrutto	Accessi	Tipo
Menu Product	E	3	L
additional request	R	3	L
Variation	E	3	L
	Totale:	19L → 1900 al giorno	

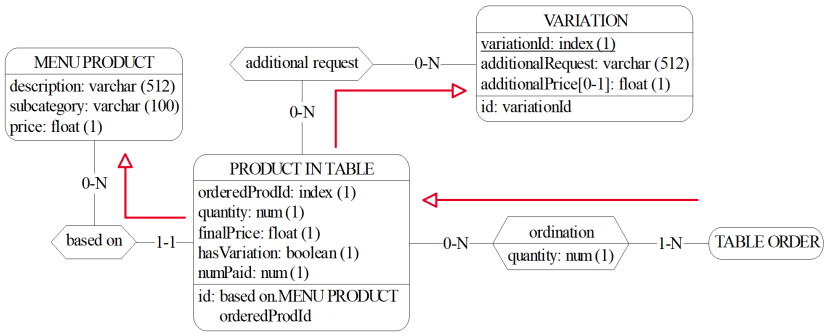


Figure 6: Schema di navigazione per l'operazione 6

3.3.7 Operazione 7 - Ricerca prodotti per pattern nome, categorie e sottocategorie

Considerando 5 prodotti ricercati:

Concetto	Costrutto	Accessi	Tipo
Menu Product	E	5	L
Stocked Up Product	E	5	L
	Totale:	10L → 40000 a settimana	

3.3.8 Operazione 8 - Visualizzare prodotti non pagati in un tavolo

Per visualizzare i prodotti non pagati è sufficiente accedere prima al *tavolo* e successivamente ai *prodotti nel tavolo*, che in media in un *tavolo* sono 5.

Concetto	Costrutto	Accessi	Tipo
Table	E	1	L
Product In Table	E	5	L
order composition	R	5	L
	Totale:	11L → 26.400 a settimana	

3.3.9 Operazione 9 - Compilare scontrini di un tavolo

Per compilare uno *scontrino* è necessario innazitutto una visualizzazione dei prodotti non pagati in un tavolo. Successivamente bisogna aggiungere dei *prodotti pagati*, in media 3 per *scontrino*, e aggiornare l'attributo *numPaid* nei relativi *prodotti ordinati*.

Concetto	Costrutto	Accessi	Tipo
Receipt	E	1	S
Table	E	1	L
Product In Table	E	5	L
order composition	R	5	L
paid product	R	3	S
Product In Table	E	3	S
	Totale:	7S + 11L → 15.000 a settimana	

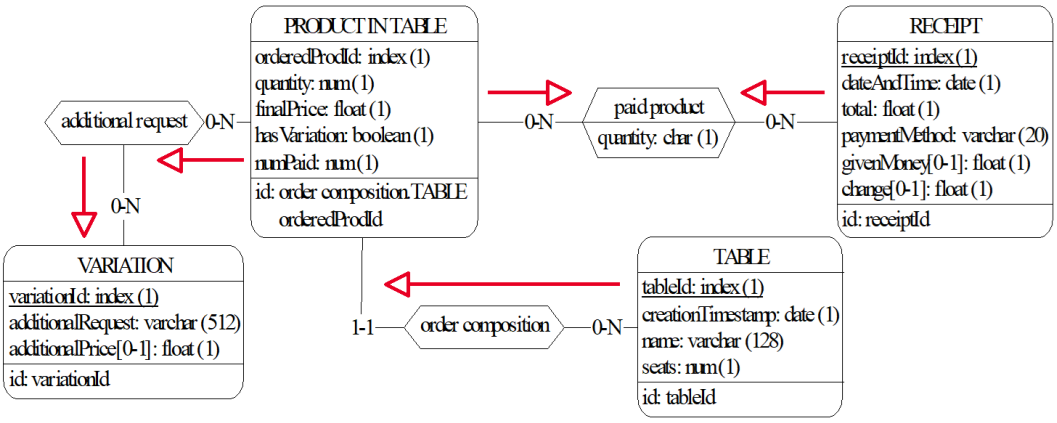


Figure 7: Schema di navigazione per l'operazione 9

3.3.10 Operazione 10 - Visualizzare i guadagni in un dato periodo

E' sufficiente accedere ai dati dei pagamenti degli *scontrini*. Considerando una base di visualizzazzione settimanale[9] di 400 scontrini da controllare:

Concetto	Costrutto	Accessi	Tipo
Receipt	E	400	L
	Totale:	400L → 4000 al mese	

3.3.11 Operazione 11 - Compilare gli ordini del magazzino

Per compilare un *ordine di magazzino* è necessario accedere ai *prodotti di magazzino*, in media 20 per ordine, scegliere il *fornitore*, in media sono 4, confrontando il *costo*. Occorre poi creare gli *elementi di fornitura* con le quantità scelte.

[9]Ci si attiene alle stime fatte precedentemente, che erano su base settimanale. Il costo può essere moltiplicato per trovare la stima mensile e/o annuale.

Concetto	Costrutto	Accessi	Tipo
Stock Order	E	1	S
order planning	R	1	S
Supply Item	E	20	S
inclusion	R	20	S
Stocked Up Product	E	20	L
type	R	20	S
provision	R	20	S
supply cost	R	80	L
Supplier	E	4	L
	Totale:	81S + 104L → 798 a settimana	

3.3.12 Operazione 12 - Visualizzare le spese per i rifornimenti in un dato periodo

Considerando come riferimento una settimana[10]:

Concetto	Costrutto	Accessi	Tipo
Stock Order	E	3	L
	Totale:	3L → 30 al mese	

3.3.13 Operazione 13 - Visualizzare le prenotazioni per un dato giorno

Si possono considerare in media 5 prenotazioni al giorno, per ogni prenotazione si vuole visualizzare il *tavolo* previsto se esiste.

Concetto	Costrutto	Accessi	Tipo
Prenotation	E	5	L
Table	E	5	L
request	R	5	L
	Totale:	15L → 1500 a settimana	

3.3.14 Operazione 14 - Visualizzare la ricetta di un prodotto

Per ogni *prodotto in menu* consideriamo una media di 3 *ingredienti*, come già fatto in precedenza.

Concetto	Costrutto	Accessi	Tipo
Menu Product	E	1	L
Stocked Up Product	E	3	L
ingredient	R	3	L
	Totale:	7L → 140 al giorno	

[10] Ci si attiene alle stime fatte precedentemente, che erano su base settimanale. Il costo può essere moltiplicato per trovare la stima mensile e/o annuale.

3.3.15 Operazione 15 - Aggiornare staff del locale

Concetto	Costrutto	Accessi	Tipo
Employee	E	1	S
Employee	E	1	L
	Totale:	1S + 1L → 15 all'anno	

3.4 Riepilogo dei costi stimati per operazione

Di seguito viene stilata una tabella riassuntiva coi costi sopraelencati:

Indice	Operazione	Frequenza	Costo stimato
1.	Aggiornare prodotti	50 all'anno	500 all'anno
2.	Aggiungere tavoli	200 a settimana	400 a settimana
3.	Compilare comande al tavolo	800 a settimana	31.200 a settimana
4.	Mostrare comande filtrate per stato e numerate per giornata	30 al giorno	1.800 al giorno
5.	Mostare numero tavoli e clienti serviti in un dato periodo	2 al giorno	180 al giorno
6.	Visualizzare prodotti e variazioni nella comanda di un tavolo	100 al giorno	1.900 al giorno
7.	Ricerca prodotti per pattern nome, categoria e sottocategoria	4000 a settimana	40.000 a settimana
8.	Visualizzare prodotti non pagati in un tavolo	2400 a settimana	26.400 a settimana
9.	Compilare scontrini di un tavolo	600 a settimana	15.000 a settimana
10.	Visualizzare i guadagni in un dato periodo	10 al mese	4.000 al mese
11.	Compilare ordini del magazzino	3 a settimana	798 a settimana
12.	Visualizzare spese per rifornimenti in un dato periodo	10 al mese	30 a settimana
13.	Visualizzare le prenotazioni per un dato giorno	100 a settimana	1.500 a settimana
14.	Visualizzare la ricetta di un prodotto	20 al giorno	140 al giorno
15.	Aggiornare staff del locale	5 all'anno	10 all'anno

3.5 Raffinamento dello schema

3.5.1 Eliminazione delle gerarchie

Per la gerarchia dei *dipendenti*, poiché la copertura è sovrapposta e le diverse specializzazioni non presentano attributi aggiuntivi, ma si configurano più come delle specie di permessi per effettuare

determinate operazioni, si è scelto il collasso verso l'alto, aggiungendo all'entità *dipendente* tanti attributi booleani quante sono le specializzazioni.

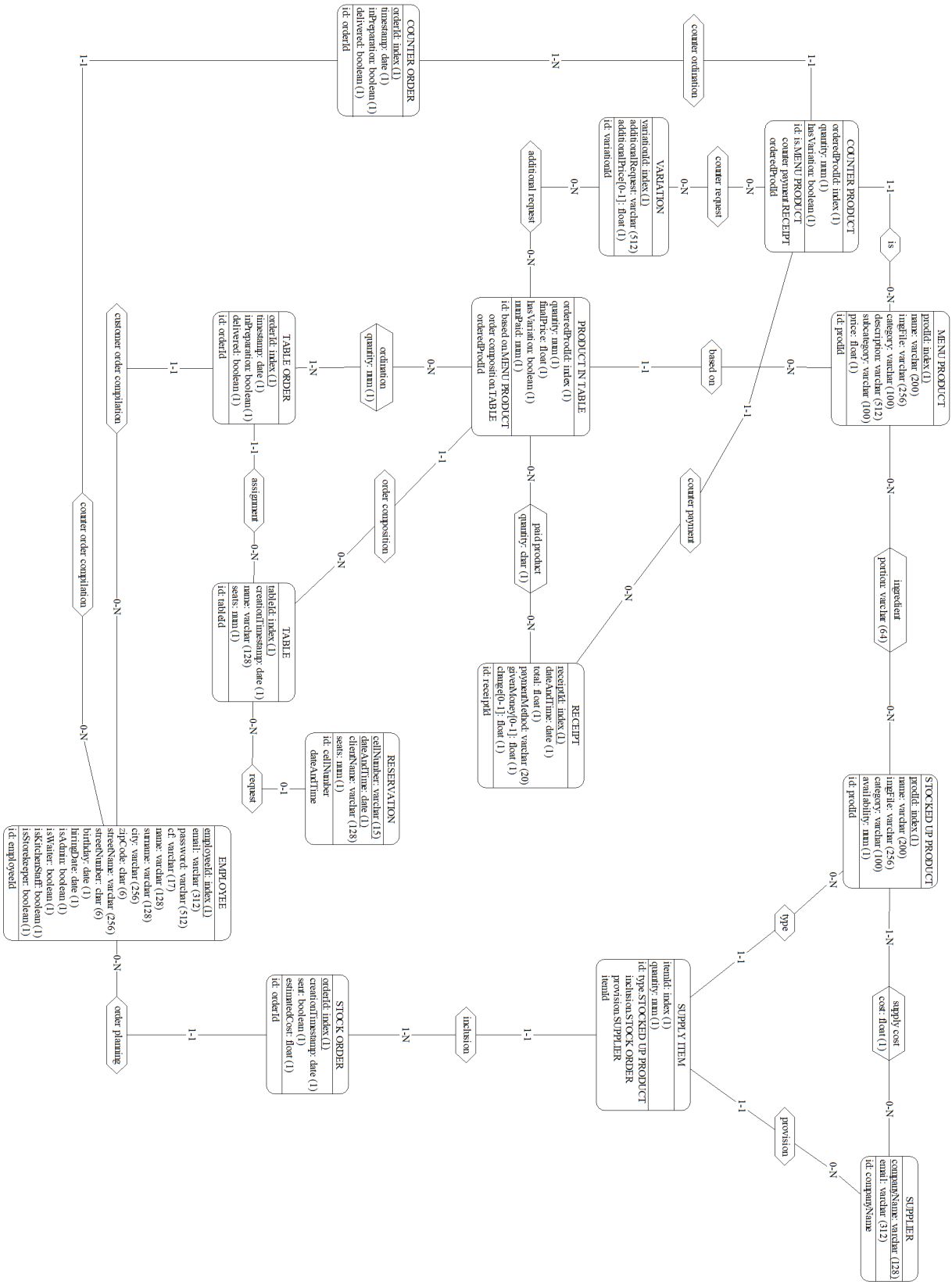
Per la gerarchia dei *prodotti* si è invece scelto un collasso verso il basso, replicando gli attributi di *prodotto* in *prodotto in menu* e *prodotto in magazzino*. Si adotta questa strategia in quanto la ridondanza, essendo non esclusiva, è presente, ma effettivamente trascurabile. La quantità di prodotti che sono sia prodotti da magazzino che prodotti in menù è infatti scarsa (ad esempio bevande e pochi altri prodotti). Inoltre le operazioni svolte sui prodotti in menù vengono effettuate molto più spesso di quanto venga fatto per i prodotti in magazzino.

Per la gerarchia delle *comande* si è scelto un collasso verso il basso, in quanto le tabelle delle *comande al banco* e quelle delle *comande al tavolo* vengono accedute in contesti separati. E' importante sottolineare che le operazioni che coinvolgono le comande di un tavolo sono generalmente più dispendiose, è preferibile dunque separare i contesti, anche considerando il fatto che la copertura è parziale e non sovrapposta, perciò non comporta ridondanza.

3.5.2 Eliminazione attributi composti

Nello schema è presente l'attributo composto *indirizzo* nell'entità *dipendente*. Tale attributo è stato diviso nelle sue sotto componenti, si renderà necessario poi, a livello applicativo, che tali valori siano impostati in modo coerente tra loro.

Di seguito si riporta lo schema ER raffinato:



3.5.3 Eliminazione degli identificatori esterni

Nello schema E/R sono eliminate le seguenti relazioni:

- **ingredient**: reificata importando *menuProdId* da *Menu Product* e *ingredientId* da *Stocked Up Product*
- **based on**: eliminata importando *menuProdId* da *Menu Product* a *Product In Table*
- **is**: eliminata importando *menuProdId* da *Menu Product* a *Counter Product*
- **additional request**: reificata importando *menuProdId* da *Menu Product*, *tableId*, *orderedProdId* da *Product in Table* e *variationId* da *Variation*
- **counter request**: reificata importando *menuProdId* da *Menu Product*, *orderedProdId* da *Counter Product*, *receiptId* e *variationId*
- **ordination**: reificata importando *menuProdId* da *Menu Product*, *orderId* da *Customer Order*, *orderedProdId* da *Product In Table* e *tableId*
- **counter ordination**: eliminata importando *orderId* da *Counter Order*
- **counter payment**: eliminata importando *receiptId* da *Receipt*
- **assignment**: eliminata importando *tableId* in *Customer Order*
- **order composition**: eliminata importando *tableId* in *Product in Table*
- **paid product**: reificata importando *orderedProdId* da *Product In Table*, *menuProdId* da *Menu Product* e *tableId*
- **order compilation**: eliminata importando *waiterId* da *Employee*
- **request**: eliminata importando *tableId* da *Table* a *Reservation*
- **supply cost**: reificata importando *prodId* da *Stocked Up Product* e *supplierName* da *Supplier*
- **type**: eliminata importando *prodId* da *Stocked Up Product* a *Supply Item*
- **provision**: eliminata importando *supplierName* da *Supplier* a *Supply Item*
- **inclusion**: eliminata importando *orderId* da *Stock Order* a *Supply Item*
- **order planning**: eliminata importando *storekeeperId* da *Storekeeper* a *Stock Order*

3.6 Analisi delle ridondanze

3.6.1 Ridondanza attributi di Product in Table

Nell'entità *Product In Table* gli attributi *numPaid*, *quantity* e *finalPrice* sono ridondanze. Tuttavia, a discapito di un piccolo overhead in termini di memoria si ha vantaggio nelle operazioni più eseguite, infatti:

- Per reperire *numPaid* occorrerebbero, in media, 2 accessi in lettura a *Paid Product*
- Per ottenere *quantity* occorrerebbero, in media, 2 accessi in lettura a *ordination*
- Per conoscere la quantità dei **non pagati** occorrerebbero quindi almeno 4 accessi in lettura in più ogni volta
- Per ottenere *final price* occorrerebbe per *Product In Table* un accesso in lettura a *Menu Product*, *additional request* e *Variation*, quindi circa 3 accessi in lettura in più per prodotto considerando una sola variazione

Operazione 8 - visualizzare prodotti non pagati in un tavolo:

con ridondanza:

Concetto	Costrutto	Accessi	Tipo
Table	E	1	L
Product In Table	E	5	L
order composition	R	5	L
	Totale:	11L → 26.400 a settimana	

senza ridondanze di *quantity* e *numPaid*:

Concetto	Costrutto	Accessi	Tipo
Table	E	1	L
Product In Table	E	5	L
order composition	R	5	L
paid product	E	10	L
ordination	E	10	L
	Totale:	*31L → 74.400 a settimana*	

Le 10 letture in paid product e ordination sono dovute al fatto che si considera che ogni prodotto al tavolo viene ordinato in due comande differenti e pagato in 2 scontrini differenti. Per un totale di 24.000 accessi risparmiati per ognuna delle due ridondanze

Operazione 9 - compilare uno scontrino di un tavolo:

con ridondanza:

Concetto	Costrutto	Accessi	Tipo
Receipt	E	1	S
Table	E	1	L
Product In Table	E	5	L
order composition	R	5	L
paid product	R	3	S
Product In Table	E	3	S
	Totale:	7S + 11L → 15.000 a settimana	

senza ridondanza di *final price*:

Concetto	Costrutto	Accessi	Tipo
Receipt	E	1	S
Table	E	1	L
Product In Table	E	5	L
order composition	R	5	L
paid product	R	3	S
Product In Table	E	3	S
based on	R	3	L
Menu Product	E	3	L
additional request	R	3	L
Variation	E	3	L
	Totale:	7S + 23L → 22.200 a settimana	

E' inoltre necessario notare che questa operazione dovrebbe essere effettuata contestualmente alla visualizzazione dei prodotti non pagati. Per compilare uno scontrino infatti bisogna avere necessariamente conoscenza delle quantità dei prodotti pagati e non. Considerando ciò è doveroso sottolineare che senza le ridondanze *quantity* e *numPaid* avremmo nuovamente 20 accessi in lettura in più e 3 accessi in scrittura in meno. Il totale avrebbe perciò *14* accessi in più, portando il costo a *30.600* a settimana.

3.6.2 Ridondanza attributo di Stock Order

Nell'entità *Stock Order* l'attributo *estimatedCost* è ridondante. Si preferisce mantenere la ridondanza in quanto tale attributo viene utilizzato in query statistiche, le quali possono prendere in considerazione periodo di tempo di anni, nei quali il numero di ordini aumenta, insieme ai prodotti ordinati.
Per reperire *estimatedCost* sarebbe necessario:

- 20 accessi in lettura a **Supply Item** per reperire le quantità dei prodotti ordinati
- 20 accessi in lettura a **type**, **Stocked Up Product**, **supply cost**, **provision** e **Supplier** per ottenere il costo dei singoli prodotti

per un totale di 120 accessi in lettura in più per ogni ordine.

Operazione 12 - visualizzare le spese dei rifornimenti in un dato periodo

con ridondanza:

Concetto	Costrutto	Accessi	Tipo
Stock Order	E	3	L
	Totale:	3L → 30 al mese	

senza ridondanza di *estimatedCost*:

Concetto	Costrutto	Accessi	Tipo
Stock Order	E	3	L
Supply Item	E	20 * 3	L
type	R	20 * 3	L
Stocked Up Product	E	20 * 3	L
supply cost	R	20 * 3	L
provision	R	20 * 3	L
Supplier	E	20 * 3	L
	Totale:	363L → 3630 al mese	

Tutto ciò è calcolato considerando visualizzazzioni di una settimana, ma una singola visualizzazzione su base annuale avrebbe un costo di 17.424L (121 accessi in lettura moltiplicati per 144 ordini medi annuali)!

3.6.3 Ridondanza attributo in Receipt

Nell'entità *Receipt* l'attributo *total* è ridondante. Si sceglie di mantenere tale ridondanza per permettere al software funzionalità di compilazione scontrini senza alcun prodotto collegato. Tale funzionalità è spesso desiderata nel settore per poter permettere di pagare scontrini generici, magari con prodotti di cui non si vuole tener traccia nel database. Inoltre permette di confrontare l'andamento dei guadagni anche rispetto ad anni in cui questo database non esisteva ancora semplicemente redigendo scontrini fittizi, ai quali non è collegato nessun prodotto.

3.6.4 Ridondanza associazione *assignment*

L'associazione **assignment** tra **Table Order** e **Table** è ridondante. Si potrebbe infatti ricavare da **Product in Table**, attraverso l'associazione **ordination**, tuttavia ciò comporterebbe due accessi addizionali su due tabelle molto numerose. Il progetto non ha attualmente operazioni che sfruttano tale vantaggio, ma è facile intuire che in implementazioni future associare tutte le comande ad un tavolo sarà un operazione sicuramente da aggiungere e che verrà utilizzata molto di frequente.

3.7 Traduzione di entità e associazioni in relazioni

- EMPLOYEES(employeeId, email, password, cf, name, surname, birthday, hiringDate, city, zipCode, streetName, streetNumber, isAdmin, isWaiter, isKitchenStaff, isStoreKeeper)
- MENU_PRODUCTS(prodId, name, imgFile, category, description, subcategory, price)
- STOCKED_UP_PRODUCTS(prodId, name, imgFile, category, subcategory, availability)
- INGREDIENTS(menuProdId, ingredientId, portion)

  FK: menuProdId REFERENCES MENU_PRODUCTS
  FK: menuProdId REFERENCES MENU_PRODUCTS
- VARIATIONS(variationId, additionalRequest, additionalPrice*)
- TABLES(tableId, creationTimestamp, name, seats)
- RESERVATIONS(cellNumber, dateAndTime, clientName, seats, tableId*)

  FK: tableId REFERENCES TABLES
- RECEIPTS(receiptId, dateAndTime, total, paymentMethod, givenMoney*, change*)
- COUNTER_ORDERS(orderId, timestamp, inPreparation, delivered, waiterId)

  FK: waiterId REFERENCES EMPLOYEES
- COUNTER_PRODUCTS(orderedProdId, menuProdId, receiptId, quantity, orderId)

  FK: menuProdId REFERENCES MENU_PRODUCTS FK: receiptId REFERENCES RECEIPTS FK: orderId REFERENCES COUNTER_ORDERS
- PRODUCTS_IN_TABLE(orderedProdId, menuProdId, tableId, quantity, finalPrice, hasVariations, numPaid)

  FK: tableId REFERENCES TABLES
  FK: menuProdId REFERENCES MENU_PRODUCTS
- PAID_PRODUCTS(orderedProdId, menuProdId, tableId, receiptId, quantity)

  FK: receiptId REFERENCES RECIPTS
  FK: (orderedProdId, menuProdId, tableId) REFERENCES PRODUCTS_IN_TABLE
- ADDITIONAL_REQUESTS(variationId, tableId, orderedProdId, menuProdId)

  FK: (variationId) REFERENCES VARIATIONS
  FK: (orderedProdId, menuProdId, tableId) REFERENCES PRODUCTS_IN_TABLE
- COUNTER_REQUESTS(variationId, receiptId, orderedProdId, menuProdId)

  FK: (variationId) REFERENCES VARIATIONS
  FK: (orderedProdId, menuProdId, receiptId) REFERENCES COUNTER_PRODUCTS
- TABLE_ORDERS(orderId, tableId, timestamp, inPreparation, delivered, waiterId)

FK: tabledId REFERENCES TABLES
FK: waiterId REFERENCES EMPLOYEES

- ORDINATIONS(orderId, menuProdId, tableId, orderedProdId, quantity)

  FK: (orderId) REFERENCES TABLE_ORDERS
  FK: (orderedProdId, menuProdId, tableId) REFERENCES PRODUCTS_IN_TABLE

- SUPPLIERS(companyName, email)
- SUPPLY_COSTS(prodId, companyName, cost)

  FK: prodIs REFERENCES STOCKED_UP_PRODUCTS
  FK: companyName REFERENCES SUPPLIERS

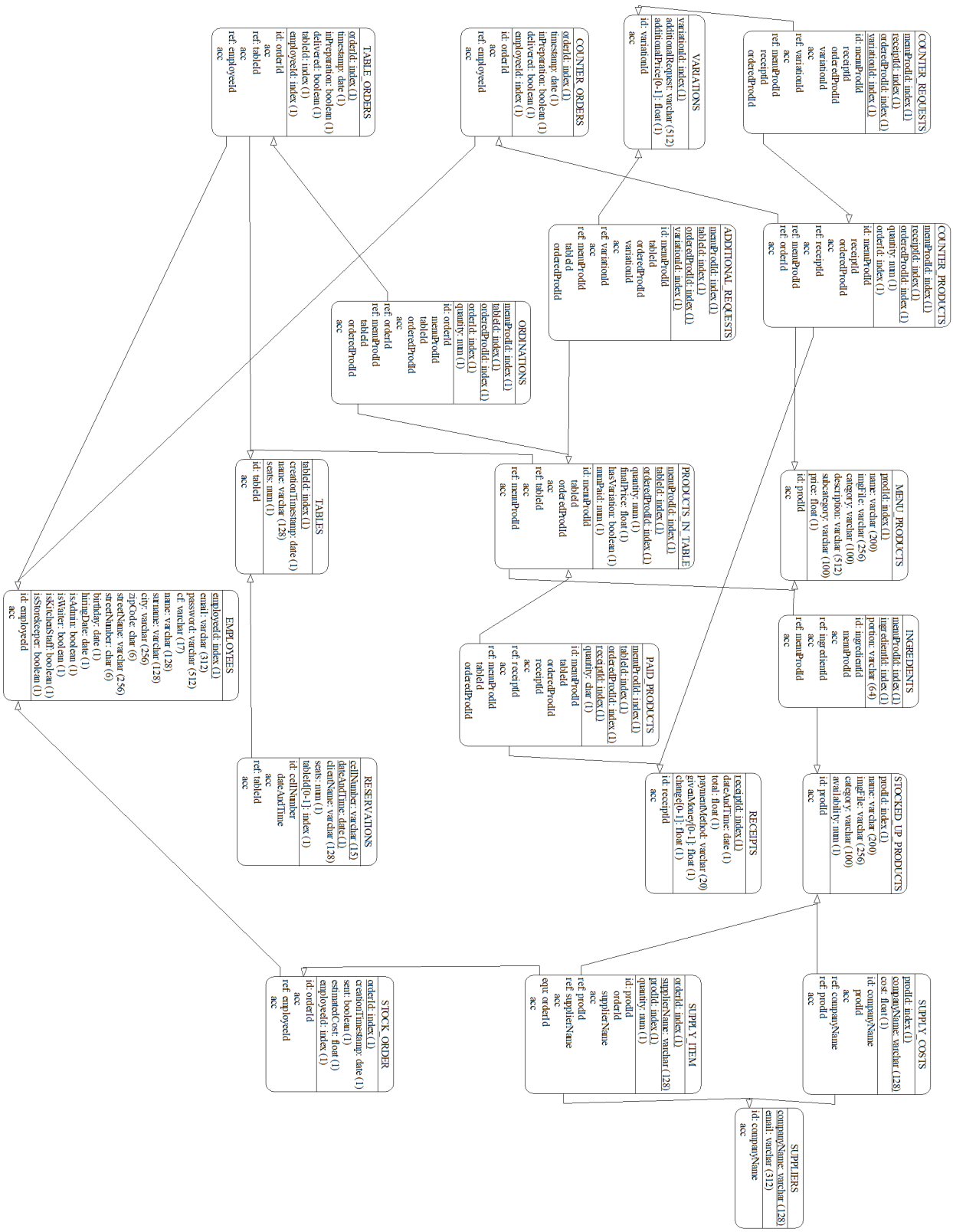
- STOCK_ORDERS(orderId, creationTimestamp, sent, estimatedCost, storekeeperId)

  FK: storekeeperId REFERENCES EMPLOYEES

- SUPPLY_ITEMS(prodId, orderDate, supplierName, quantity)

  FK: supplierName REFERENCES SUPPLIERS
  FK: orderDate REFERENCES STOCK_ORDERS
  FK: prodId REFERENCES STOCKED_UP_PRODUCTS

Di seguito si allega lo schema logico risultante:



4 Traduzione delle operazioni in query SQL

4.1 Operazione 1

Di seguito vengono elencate le query per i **MENU_PRODUCTS**, ma le query sono esattamente analoghe per gli **STOCKED_UP_PRODUCTS**.

- Aggiungere un prodotto:

```
INSERT INTO
MENU_PRODUCTS
    (name, `category`, subcategory, description, price, imgFile)
VALUES
    (?, ?, ?, ?, ?, ?);
```

- Rimuovere un prodotto:

```
DELETE FROM MENU_PRODUCTS
WHERE prodId = ?;
```

- Modificare un prodotto:

```
UPDATE MENU_PRODUCTS
SET name = ?, ..., price = ?
WHERE prodId = ?
```

  La query viene modificata dinamicamente utilizzando PHP per aggiungere o togliere gli attributi da modificare.

4.2 Operazione 2

```
INSERT INTO TABLES (creationTimestamp, name, seats)
VALUES (?, ?, ?);
```

4.3 Operazione 3

```
START TRANSACTION;

-- Inserimento di un nuovo ordine
INSERT INTO TABLE_ORDERS (tableId, waiterId)
VALUES (?, ?);

-- Le seguenti operzioni vanno ripetute per ciascun prodotto

-- Setta @finalPrice col prezzo di menu del prodotto
SELECT price
INTO @finalPrice
FROM MENU_PRODUCTS
```

```
WHERE prodId = ?;

-- Aggiunge al prezzo iniziale i prezzi delle variazioni
-- (oppure 0 se non ci sono variazioni)
SELECT
  @finalPrice := @finalPrice + COALESCE(SUM(additionalPrice), 0)
FROM VARIATIONS
WHERE variationId IN (?, ..., ?);

-- Controlla se esiste un prodotto al tavolo con:
-- - Stesso ID
-- - Stesse variazioni
SET @existingProdId := (
    SELECT pit.orderedProdId
    FROM PRODUCTS_IN_TABLE pit
    LEFT JOIN ADDITIONAL_REQUESTS ar
        ON pit.orderedProdId = ar.orderedProdId
        AND pit.menuProdId = ar.menuProdId
        AND pit.tableId = ar.tableId
    WHERE pit.menuProdId = ?
        AND pit.tableId = ?
        AND pit.hasVariation = ?
    GROUP BY pit.orderedProdId
    HAVING COUNT(DISTINCT ar.variationId) = ?
    AND SUM(CASE WHEN ar.variationId IN (?, ..., ?)
            THEN 1 ELSE 0 END
           ) = ?
);

-- Se non ho trovato il prodotto, lo creo
IF @existingProdId IS NULL THEN
    INSERT INTO PRODUCTS_IN_TABLE
        (menuProdId, tableId, quantity,
          finalPrice, hasVariation, numPaid)
    VALUES (?, ?, ?, ?, ?, 0);

    -- Inserisci variazioni se presenti
    INSERT INTO ADDITIONAL_REQUESTS
    (variationId, tableId, orderedProdId, menuProdId)
    VALUES (?, ?, INSERT_ID(), ?);
ELSE
    -- Altrimenti, aggiorna la quantità
    UPDATE PRODUCTS_IN_TABLE
    SET quantity = quantity + ?
    WHERE orderedProdId = @existingProdId
        AND menuProdId = ?
```

```
            AND tableId = ?;
    END IF;

    -- Inserisci nuova riga nelle ordinazioni
    INSERT INTO ORDINATIONS
        (orderId, menuProdId, tableId, orderedProdId, quantity)
    VALUES (?, ?, ?, ?, ?);

    COMMIT;

    -- Se ci sono errori:
    ROLLBACK;
```

La query nel progetto è implementata attraverso funzioni PHP che settano le variabili ed eseguono controlli. Le operazioni multiquery sono gestite tramite le apposite funzioni della libreria MySQLi. Quella sopra sarebbe la traduzione in SQL.

4.4 Operazione 4

```
SELECT
    `TABLES`.name AS tableName,
    `TABLES`.tableId,
    EMPLOYEES.name AS waiterName,
    EMPLOYEES.surname AS waiterSurname,
    ord.timestamp,
    ord.inPreparation,
    ord.delivered,
    ord.orderId,
    ROW_NUMBER() OVER
                (PARTITION BY DATE(ord.timestamp)
                 ORDER BY ord.timestamp) as numOfDay
FROM
    CUSTOMER_ORDERS AS ord
    LEFT JOIN `TABLES` ON ord.tableId = `TABLES`.tableId
    LEFT JOIN EMPLOYEES ON ord.waiterId = EMPLOYEES.employeeId
WHERE
    ord.timestamp LIKE ?
    AND ord.inPreparation = ?
    AND ord.delivered = ?
ORDER BY
    ord.timestamp;
```

4.5 Operazione 5

```
SELECT
    DATE(creationTimestamp) as serviceDate,
```

```
    SUM(seats) as peopleServed,
    COUNT(*) as tablesServed
FROM
    `TABLES`
WHERE
    DATE(creationTimestamp) >= ?
    AND DATE(creationTimestamp) <= ?
GROUP BY
    serviceDate
ORDER BY
    serviceDate ASC;
```

4.6 Operazione 6

```
SELECT o.*, mp.name, pit.finalPrice
FROM ORDINATIONS o
    RIGHT JOIN MENU_PRODUCTS mp ON o.menuProdId = mp.prodId
    LEFT JOIN PRODUCTS_IN_TABLE pit ON
        o.orderedProdId = pit.orderedProdId
        AND o.menuProdId = pit.menuProdId
        AND o.tableId = pit.tableId
WHERE o.tableId = ?
    AND o.orderId = ?;
```

E poi con PHP per ognuno di questi prodotti viene eseguita la query per ottenere le variazioni:

```
SELECT
    v.additionalRequest as name,
    v.additionalPrice
FROM ADDITIONAL_REQUESTS ar
    LEFT JOIN VARIATIONS v
        ON ar.variationId = v.variationId
WHERE ar.tableId = ?
    AND ar.orderedProdId = ?
    AND ar.menuProdId = ?
```

4.7 Operazione 7

```
(SELECT * FROM MENU_PRODUCTS WHERE name LIKE ?)
UNION
(SELECT * FROM MENU_PRODUCTS WHERE name LIKE ?)
UNION
(SELECT * FROM MENU_PRODUCTS WHERE category LIKE ?)
UNION
(SELECT * FROM MENU_PRODUCTS WHERE subcategory LIKE ?)
```

4.8 Operazione 8

```
SELECT
    pit.*, mp.name as prodName, t.name as tableName
FROM PRODUCTS_IN_TABLE pit
    LEFT JOIN MENU_PRODUCTS mp
        ON pit.menuProdId = mp.prodId
    LEFT JOIN `TABLES` t
        ON pit.tableId = t.tableId
WHERE pit.tableId = ?
    AND (pit.quantity - pit.numPaid) > 0;
```

A questa query può poi essere aggiunta la query per ottenere le variazioni descritta nell'Operazione 6

4.9 Operazione 9

```
START TRANSACTION;

INSERT INTO
    RECEIPTS(total, paymentMethod, givenMoney, changeAmount)
VALUES (?, ?, ?, ?);

SET @receiptId := INSERT_ID();

-- Le seguenti operazioni vanno ripetute per ogni prodotto
INSERT INTO PAID_PRODUCTS
    (orderedProdId, menuProdId, tableId, receiptId, quantity)
VALUES (?, ?, ?, @receiptId, ?);

UPDATE PRODUCTS_IN_TABLE
SET numPaid = numPaid + ?
WHERE orderedProdId = ?
    AND menuProdId = ?
    AND tableId = ?;

COMMIT;
-- Se ci sono errori:
ROLLBACK;
```

4.10 Operazione 10

```
SELECT
    DATE(dateAndTime) as receiptDate,
    SUM(total) as totalSum,
    COUNT(*) as totalPayments
```

```sql
FROM
    RECEIPTS
WHERE
    DATE(dateAndTime) >= ?      -- day from
    AND DATE(dateAndTime) <= ?  -- day to
GROUP BY
    DATE(dateAndTime)
ORDER BY
    receiptDate ASC;
```

4.11 Operazione 11

```sql
START TRANSACTION;

INSERT INTO STOCK_ORDERS (storekeeperId, estimatedCost)
VALUES (?, ?);

SET @orderId := INSERT_ID();

-- Le seguente operazione va ripetuta per ogni prodotto
INSERT INTO SUPPLY_ITEMS(prodId, orderId, supplierName, quantity)
VALUES (?, @orderId, ?, ?);

COMMIT;
-- Se ci sono errori:
ROLLBACK;
```

4.12 Operazione 12

```sql
SELECT
    DATE(creationTimestamp) as orderDate,
    SUM(estimatedCost) as totalSum,
    COUNT(*) as totalPayments
FROM
    STOCK_ORDERS
WHERE
    DATE(creationTimestamp) >= ?      -- day from
    AND DATE(creationTimestamp) <= ?  -- day to
GROUP BY
    DATE(creationTimestamp)
ORDER BY
    orderDate ASC;
```

4.13 Operazione 13

```
SELECT r.*,
    t.name as tableName
FROM RESERVATIONS r
LEFT JOIN `TABLES` t ON r.tableId = t.tableId
WHERE r.dateAndTime LIKE ?
ORDER BY r.dateAndTime;
```

4.14 Operazione 14

```
SELECT name as ingredientName, portionSize
FROM INGREDIENTS
LEFT JOIN STOCKED_UP_PRODUCTS AS stocked
    ON ingredientId = stocked.prodId
WHERE INGREDIENTS.menuProdId = ?;
```

4.15 Operazione 15

- Aggiungere un dipendente:

```
INSERT INTO EMPLOYEES(
    email, password, cf, name, surname, birthday, hiringDate,
    isWaiter, isStorekeeper, isKitchenStaff, isAdmin,
    city, zipCode, streetName, streetNumber)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
```

- Eliminare un dipendente:

```
DELETE FROM EMPLOYEES WHERE employeeId = ?;
```

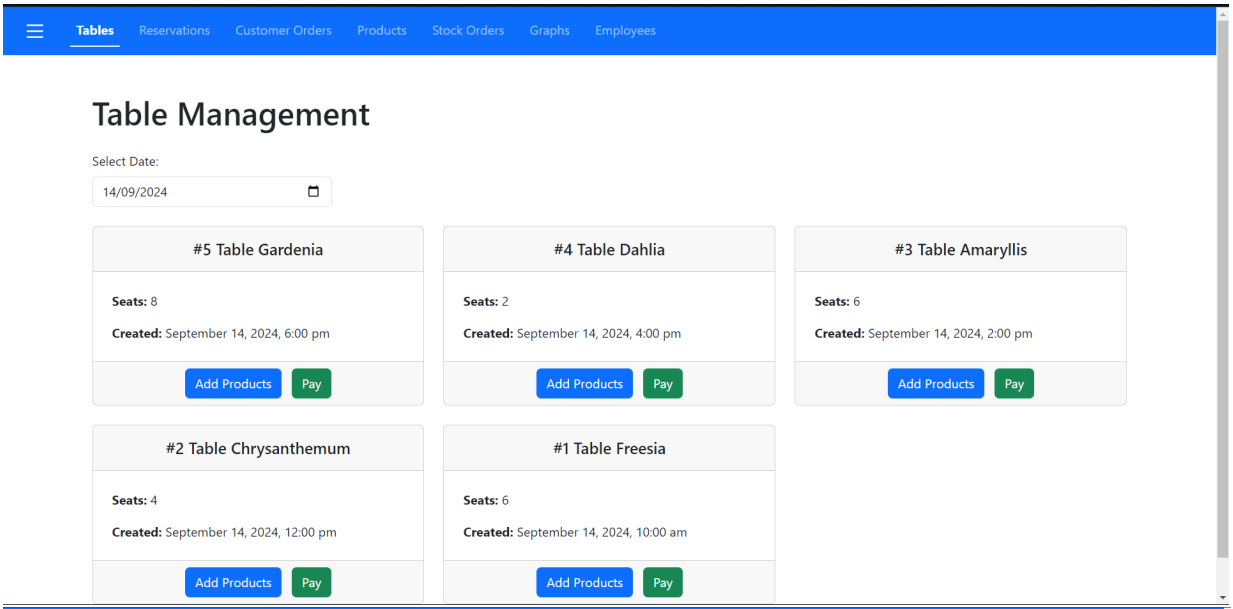
- Modificare un dipendente:

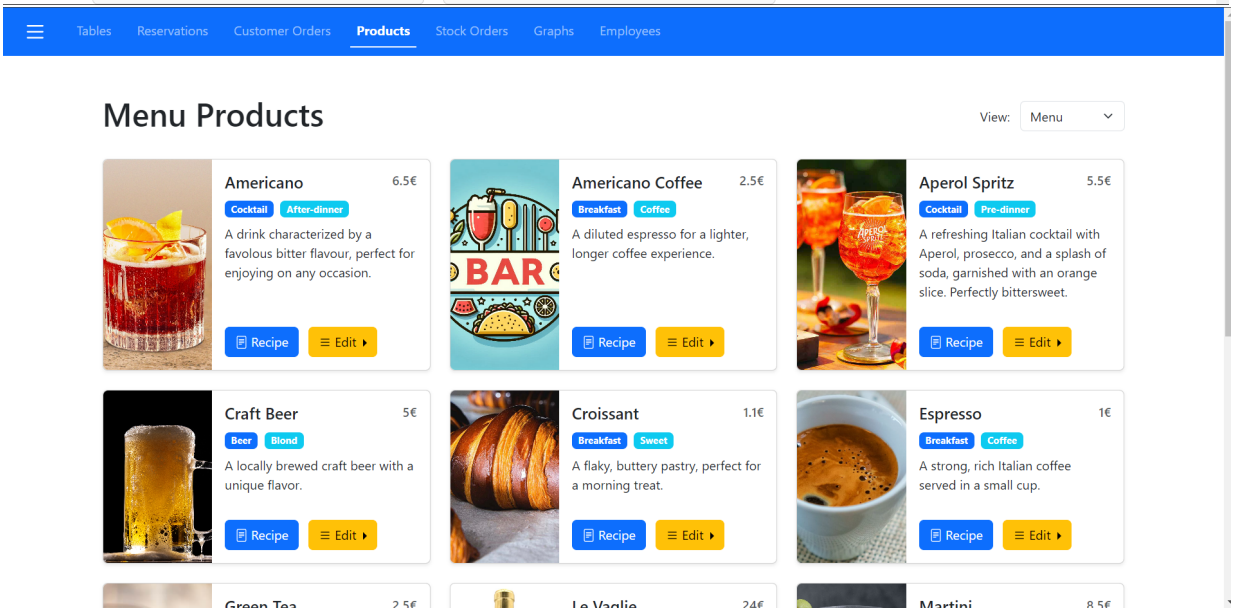
```
UPDATE EMPLOYEES
SET name = ?, surname = ?, ..., birthDay = ? WHERE employeeId = ?;
```

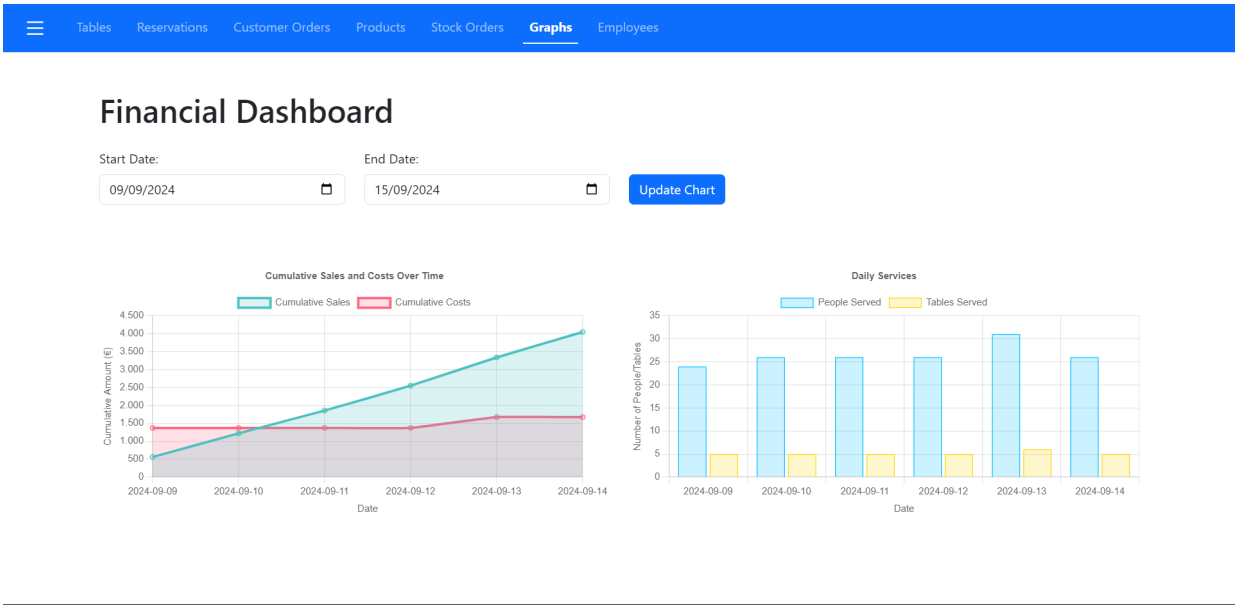
  La query viene modificata dinamicamente con PHP per scegliere gli attributi da modificare

5 Progettazione dell'applicativo web

Per lo sviluppo dell'applicativo web è stato utilizzato PHP per quanto riguarda il backend. L'interfacciamento col database è stato realizzato utilizzando la libreria MySQLi. L'applicazione presenta una semplice interfaccia grafica realizzata tramite HTML e Bootstrap. Per l'aggiornamento delle informazioni nell'interfaccia sono state utilizzate delle richieste HTTP tramite Javascript a delle API appositamente scritte in PHP. Le API fanno uso dell'oggetto DatabaseHelper per le esecuzioni delle query. Le transazioni sono gestite con PHP facendo uso delle apposite funzioni di libreria. L'applicativo fornisce in generale tutte le funzioni richieste, più qualche semplice funzione per i login e le visualizzazioni/ricerche aggiuntive.






Tables
Reservations
Customer Orders
Products
Stock Orders
Graphs
Employees
Financial Dashboard
Start Date:
09/09/2024
End Date:
15/09/2024
Update Chart
Cumulative Sales and Costs Over Time
Cumulative Sales
Cumulative Costs
Cumulative Amount (€)
4.500
4.000
3.500
3.000
2.500
2.000
1.500
1.000
500
0
2024-09-09
2024-09-10
2024-09-11
2024-09-12
2024-09-13
2024-09-14
Date
Daily Services
People Served
Tables Served
Number of People/Tables
35
30
25
20
15
10
5
0
2024-09-09
2024-09-10
2024-09-11
2024-09-12
2024-09-13
2024-09-14
Date