



GREEN MIND

Simone Mione
Mattia Malgarini





Introduzione

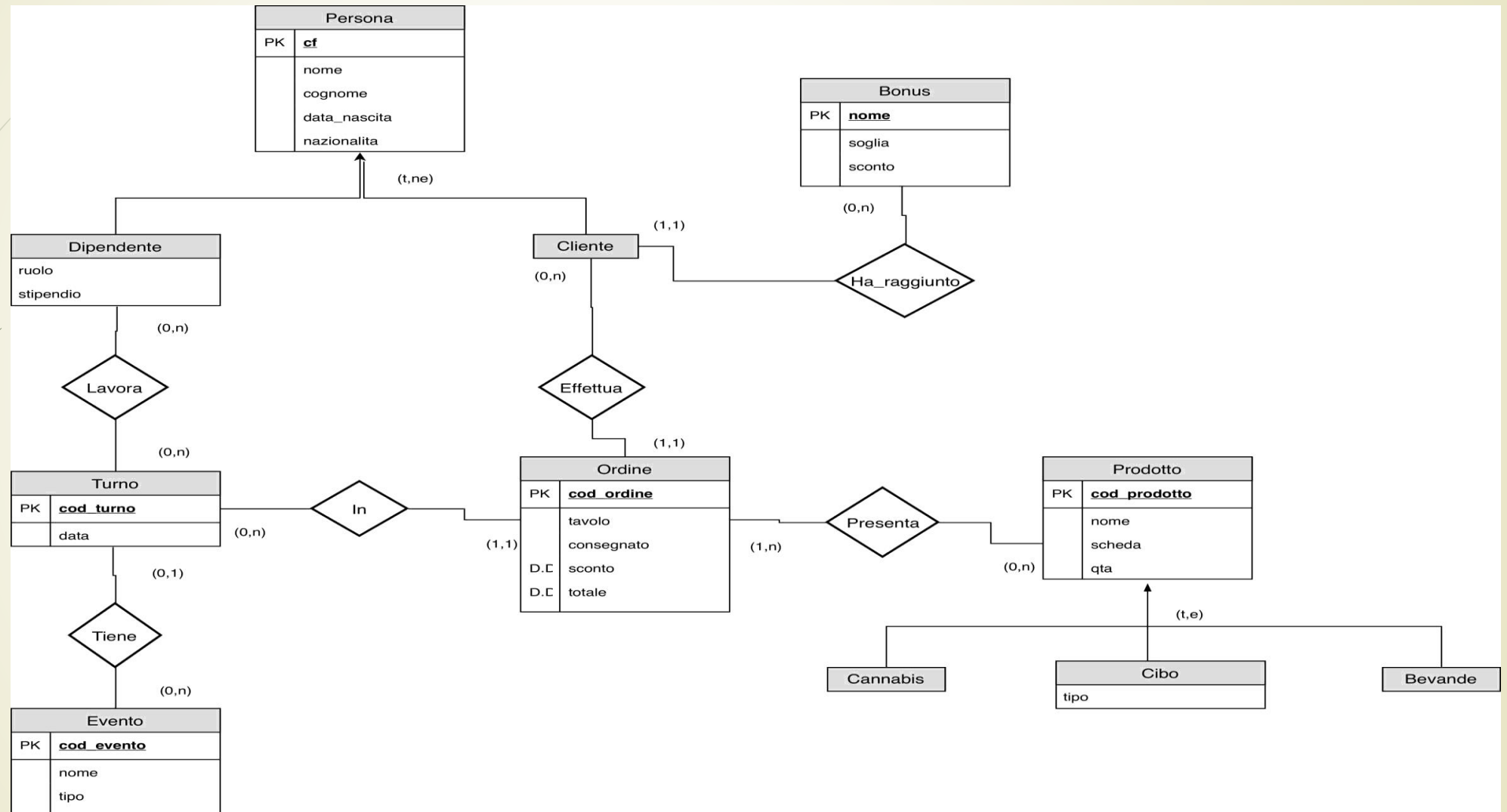
- ▶ Si vuole creare un database a supporto di un Coffee Shop che dovrà gestire i vari aspetti di un locale di questo tipo.
- ▶ In particolare l'inserimento di ordini da parte dei clienti (dopo essersi registrati), la gestione delle comande da parte dei camerieri ed una vista gestionale per il dirigente del locale, il quale potrà 'assumere' e 'licenziare' un dipendente del proprio organico, aggiungere prodotti nel menu, gestire eventi e turni ed eseguire query precompilate (oppure farne di nuove) per analisi statistiche sul proprio locale.
- ▶ Premessa: Il software che si utilizzerà per dialogare con il database è dimostrativo poiché, nel 'vero' locale ogni tavolo sarà dotato di un dispositivo nel quale l'utente potrà effettuare il login strisciando il proprio codice fiscale. Stesso cosa vale per i camerieri ed il dirigente.

Specifiche

- Le **persone** sono identificate con il proprio **codice fiscale** e sono definite dal **nome**, **cognome**, **data di nascita**, **nazionalità**; possono essere **clienti** e/o **dipendenti**: questi ultimi avranno uno **stipendio** ed un **ruolo**. Il **ruolo** può assumere valori quali, ad esempio, cameriere o barista. Lo **stipendio** è controllato affinché non sia in negativo.
- Un dipendente può lavorare in un **turno**, identificato da un **codice** auto incrementale ed è definito da una **data**.
- In un **turno** può esserci un **evento**, identificato da un **codice** auto incrementale ed è definito da un **nome** (ad es. 'Cantiamo tutti insieme') ed un **tipo** (ad es. karaoke).
- Un **cliente**, in un dato **turno**, può effettuare un **ordine**, identificato, anch'esso da un **codice** auto incrementale ed è definito da un **tavolo** e da un valore booleano **consegnato** che indica se tale ordine è stato consegnato o meno al cliente ordinante.

- 
- 
- I **clienti** possono raggiungere un **bonus**, identificato da un **nome** e definito dalla **soglia** (che il cliente deve superare con la spesa totale effettuata) e dallo **sconto** (applicato sui successivi ordini al raggiungimento di tale soglia). Il controllo che verifica se il cliente ha raggiunto tale benefit è effettuato da un trigger attivato alla consegna di un ordine effettuato.
 - Gli **ordini** sono composti dai **prodotti** richiesti dal **cliente**, i quali sono identificati da un **codice** auto incrementale, un **nome**, **scheda** (es. pomodoro, mozzarella), **prezzo** e **qta** disponibile.
 - I **prodotti** possono essere **cannabis**, **bevande**, **cibo** e quest'ultimo ha un attributo **tipo**. Il **tipo** è controllato da un check e può assumere valori quali vegetariano, vegano. Il **prezzo** è controllato per non essere negativo.

Scheda ER

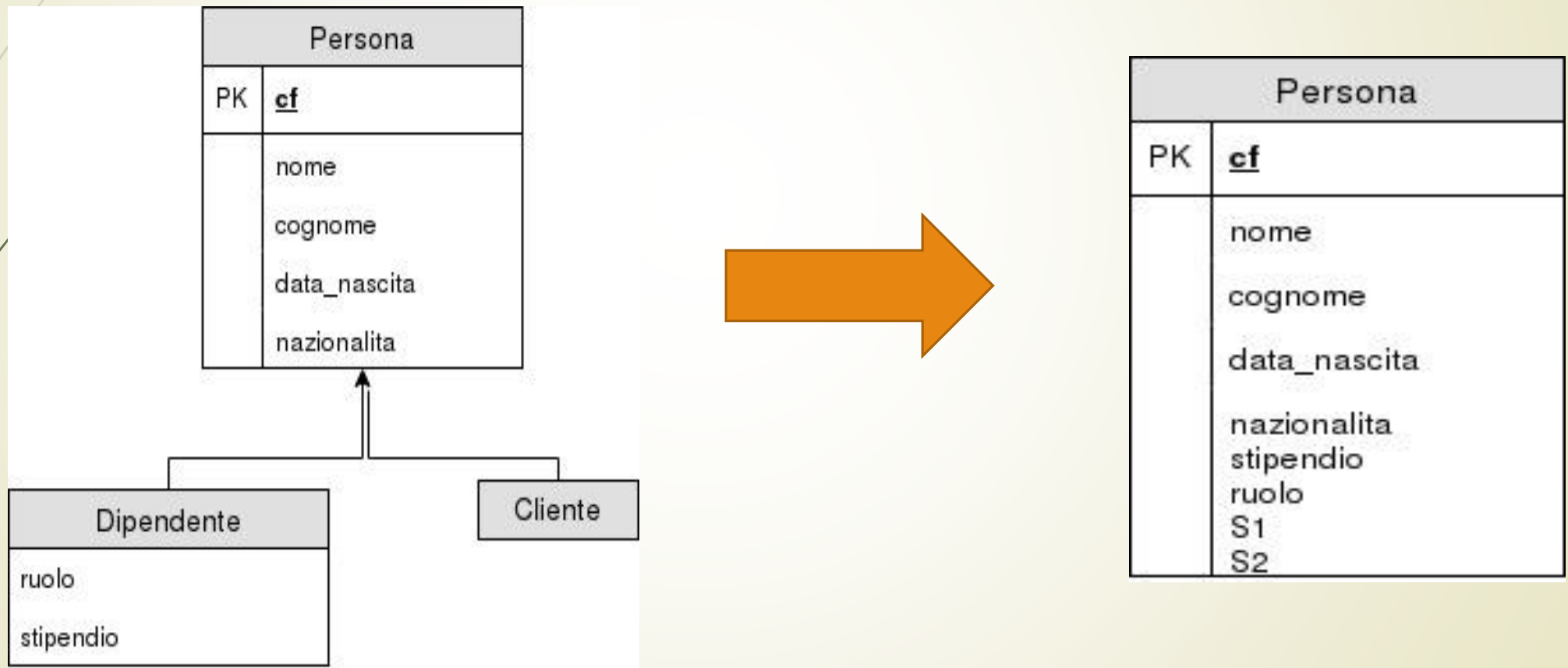


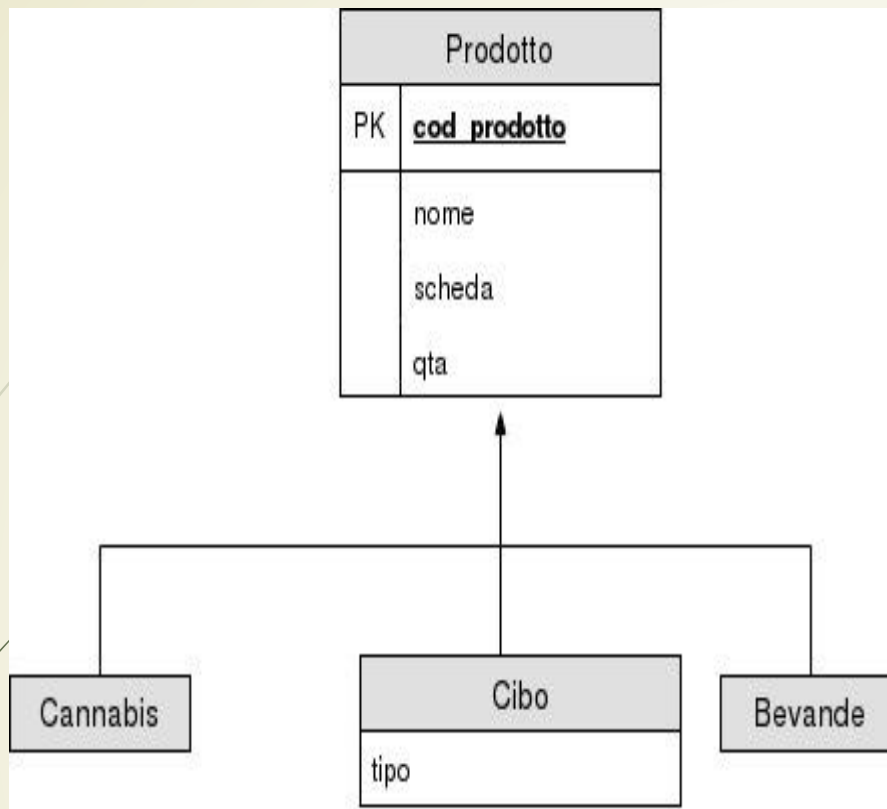
Progetto logico

Di seguito si descriveranno le varie scelte in fase di trasformazione nel progetto logico.

1. Eliminazione Gerarchie

Si è scelto, per entrambe le gerarchie presenti nel nostro db, un collasso verso l'alto dato che non sono presenti molti attributi di specializzazione.





Prodotto	
PK	<u>cod_prodotto</u>
	nome
	scheda
	qta
	tipo
	S1
	S2
	S3

2. Selezione chiavi primarie ed eliminazione delle relazioni esterne

Le chiavi primarie scelte per ogni relazione sono già state evidenziate nella specifica con grassetto e sottolineatura. Per la maggior parte delle relazioni abbiamo optato per un semplice intero auto incrementale. Le relazioni esterne eliminate sono tutte quelle per le quali avevamo cardinalità uno-a-molti, come ad esempio **turno-evento** per cui la chiave primaria di evento è diventata chiave esterna di **turno**.

3. Trasformazione degli attributi composti e multipli

Non abbiamo introdotto nessun attributo di questo Genere già nello schema ER.

4. Traduzione di entità ed associazioni in schemi di relazioni

La traduzione in schemi di relazione segue le regole standard.

Nel nostro schema non abbiamo nessun tipo di Relazione 1-a-1.

Per le relazioni 1-a-molti, quindi, si è portata la chiave primaria della relazione 'n' nella relazione '1' come chiave esterna.

Per le relazioni molti-a-molti si è creata, in aggiunta, l'entità della relazione contenente le chiavi primarie delle entità alle quali si riferisce. Nel caso di **presenta** si è aggiunto anche un attributo **qta** che rappresenta quante volte quel prodotto è presente nell'ordine.



Tabelle

Le tabelle del database saranno quindi:

- **Bonus** (nome, soglia, sconto)
- **Persona** (cf, nome, cognome, data_nascita, nazionalita, stipendio, ruolo, s1, s2, cod_bonus) FK: cod_bonus REFERENCES bonus.nome
- **Evento** (cod evento, nome, tipo)
- **Turno** (cod turno, data, cod_evento) FK: cod_evento REFERENCES evento.cod_evento
- **Lavora** (cod turno, cf)
- **Ordine** (cod ordine, tavolo, consegnato, totale, cf, cod_turno, sconto) FK: cf REFERENCES persona.cf cod_turno REFERENCES turno.cod_turno
- **Prodotto** (cod prodotto, nome, scheda, tipo, qta, prezzo, s1, s2, s3)
- **Presenta** (cod prodotto, cod ordine, qta)

Implementazione

Di seguito si mostrerà il codice per la creazione delle tabelle, delle funzioni e trigger, dell'inserimento di valori di esempio e di esecuzione di qualche query tipica.


1. Creazione Tabelle

CREATE TABLE IF NOT EXISTS bonus

```
( nome varchar primary key,  
  soglia integer check (soglia >= 0) not null unique,  
  sconto integer check (sconto >= 0) not null  
)
```

CREATE TABLE IF NOT EXISTS persona



```
( Cf varchar primary key, nome varchar not null,  
  cognome varchar not null,  
  data_nascita date not null,  
  nazionalita varchar not null,  
  stipendio real check (stipendio >= 0),  
  ruolo varchar check (ruolo in ('cameriere', 'dirigente', 'barista', null)),  
  s1 boolean,  
  s2 boolean,  
  cod_bonus varchar,  
  foreign key (cod_bonus) references bonus(nome)  
  ON DELETE SET NULL)
```



CREATE TABLE IF NOT EXISTS evento
(cod_evento SERIAL primary key,
nome varchar not null,
tipo varchar not null)



CREATE TABLE IF NOT EXISTS turno
(cod_turno SERIAL primary key,
data date not null unique,
cod_evento integer,
foreign key(cod_evento) references evento
ON DELETE SET NULL)

CREATE TABLE IF NOT EXISTS lavora
(cf varchar not null,
cod_turno integer not null,
primary key(cod_turno,cf),
foreign key(cf) references persona
ON DELETE CASCADE,
foreign key(cod_turno) references turno
ON DELETE CASCADE)



```
CREATE TABLE IF NOT EXISTS ordine
(cod_ordine SERIAL primary key,
 sconto integer not null,
 tavolo integer not null, cf varchar(16) not null,
 cf varchar(16) not null,
 cod_turno integer,
 consegnato boolean nonnull,
 totale real not null,
 foreign key(cf) references persona,
 foreign key(cod_turno) references turno
 ON DELETE SET NULL)
```

```
CREATE TABLE IF NOT EXISTS prodotto
( cod_prodotto SERIAL primary key,
 s1 boolean,
 s2 boolean,
 s3 boolean,
 qta integer check (qta >= 0),
 nome varchar not null unique,
 scheda varchar not null,"
 tipo varchar check (tipo in (null,'vegetariano','vegano')),
 prezzo real not null check (prezzo > 0))
```



CREATE TABLE IF NOT EXISTS presenta
(cod_prodotto integer not null,
cod_ordine integer not null,
qta integer not null,
Primary key(cod_prodotto,cod_ordine),
foreign key(cod_prodotto) references prodotto
ON DELETE CASCADE,
foreign key(cod_ordine) references ordine
ON DELETE CASCADE)

2. Funzioni e Trigger

CREATE OR REPLACE FUNCTION

aggiungiDipendenteAlTurno (character varying, integer)

RETURNS BOOLEAN AS

\$\$

DECLARE

dip character varying := (select s1 from persona where cf = \$1);

BEGIN

IF (dip) THEN

INSERT INTO lavora values (\$1,\$2)

return true;

ELSE

return false;

END IF;

END;

\$\$

LANGUAGE plpgsql;

Questa funzione è chiamata all'inserimento di un dipendente in un turno di lavoro. Viene controllato se la persona che si cerca di inserire sia effettivamente un dipendente e non solo un cliente.



CREATE OR REPLACE FUNCTION add_bonus() RETURNS trigger AS
\$BODY\$

DECLARE\n"

nome_bonus varchar;

spesa real;

cf_per varchar;

BEGIN

SELECT cf into cf_per FROM

ordine,presenta where ordine.cod_ordine = presenta.cod_ordine;

SELECT sum((presenta.qta*prodotto.prezzo)) into spesa from

persona, prodotto,presenta,ordine WHERE presenta.cod_prodotto =

prodotto.cod_prodotto AND presenta.cod_ordine = ordine.cod_ordine AND

ordine.cf=persona.cf AND persona.cf=cf_per;

limit 1;
SELECT nome INTO nome_bonus FROM bonus WHERE soglia<spesa order by soglia desc

UPDATE persona SET cod_bonus = nome_bonus WHERE persona.cf=cf_per;

RETURN null;

END;

\$BODY\$

LANGUAGE plpgsql VOLATILE;



```
CREATE TRIGGER setBonus AFTER INSERT ON presenta  
FOR EACH ROW EXECUTE PROCEDURE add_bonus();
```

Questo trigger, attivato alla consegna di un ordine, verifica se il cliente ha raggiunto una certa soglia di spesa totale e, in tal caso, viene settato il suo bonus.

CREATE OR REPLACE FUNCTION

aggiungiPresenta(character varying, integer, integer, integer) RETURNS boolean AS
\$\$

DECLARE

eta integer := (select date_part('year',age(data_nascita)) from persona where cf=\$1)

iscannabis := (select s1 from prodotto where cod_prodotto=\$2)

BEGIN

IF (eta < 18 AND iscannabis) THEN

delete from ordine where cod_ordine=\$3;

return false;

ELSE

INSERT INTO presenta VALUES (\$2,\$3,\$4);

UPDATE prodotto SET qta=(SELECT qta

FROM prodotto WHERE cod_prodotto=\$2)-\$4 WHERE cod_prodotto=\$2;

END IF;

END;

\$\$

LANGUAGE plpgsql;

Questa funzione è chiamata per inserire un ordine e controlla se un minorenne sta tentando di acquistare cannabis. In tal caso l'ordine viene annullato eliminandolo e ripristinando le qta nei prodotti, quest'ultima funzionalità è implementata dal prossimo trigger.

CREATE OR REPLACE FUNCTION ripristinoQta()

RETURNS trigger AS

\$\$

BEGIN

**UPDATE prodotto SET qta = qta+(SELECT qta from presenta where
presenta.cod_ordine =
OLD.cod_ordine AND presenta.cod_prodotto =
prodotto.cod_prodotto) WHERE prodotto.cod_prodotto in (SELECT
presenta.cod_prodotto from presenta where presenta.cod_ordine =
OLD.cod_ordine);**

RETURN OLD;

END

\$\$

LANGUAGE plpgsql VOLATILE;

CREATE TRIGGER ripristinaQta BEFORE DELETE ON ordine

FOR EACH ROW EXECUTE PROCEDURE

ripristinoQta();

Questo trigger ripristina le qta nel caso in cui un ordine viene eliminato.

3. Utilizzo database con valori di esempio

- *Quando un cliente accede per la prima volta al locale viene registrato*
INSERT INTO persona VALUES ('ABCDEF93G07H345I', 'Simone', 'Malgarini', date('07/07/1993'), 'Italia', null, null, 'base');
- *Il cliente tenta di effettuare il login e nel caso viene lasciato entrare*
SELECT * FROM cliente WHERE cf='BCDEF93G07H345I' AND s2=true;
- *La vista del cliente, quindi il menù disponibile, permette di effettuare un ordine scegliendo la quantità desiderata per ogni prodotto*
INSERT INTO ordine (tavolo, cf, cod_turno, consegnato, sconto, totale)
VALUES (1, 'BCDEF93G07H345I', 1, false, 0, 10);
SELECT aggiungiPresenta('BCDEF93G07H345I', 1, 1, 1);
SELECT aggiungiPresenta('BCDEF93G07H345I', 9, 1, 1);
n.b. essendo maggiorenne l'acquisto è andato a buon fine.
- *Ora nella vista del cameriere possiamo vedere quali tavoli hanno un ordine ancora da consegnare*
SELECT tavolo FROM ordine WHERE consegnato=false
- *Entrando nella vista del tavolo il cameriere può vedere quali ordini ci sono (ed i relativi prodotti facenti parte) e può eliminarli o settarli come consegnati*
SELECT ordine.cod_ordine, tavolo, cf, presenta.qta, nome, prezzo, sconto, totale
FROM ordine, presenta, prodotto WHERE ordine.cod_ordine = presenta.cod_ordine
AND prodotto.cod_prodotto = presenta.cod_prodotto
AND ordine.consegnato = false AND tavolo = 1
GROUP BY ordine.cod_ordine, tavolo, cf, presenta.qta, nome, prezzo, sconto, totale;

Elimina: DELETE FROM ordine WHERE cod_ordine = 1;

n.b. In questo caso si attiva il trigger che ripristina le qta che il cliente aveva ordinato.

Set Consegnato: UPDATE ordine SET consegnato = true WHERE cod_ordine = 1;

n.b. In questo caso si attiva il trigger per impostare al cliente il bonus nel caso lo abbia raggiunto con quest'ultima spesa.

- Il dirigente può, invece, dalla sua vista, gestire le persone, i prodotto, gli eventi, i turni ed in più ha delle query pre-compilate per statistiche quali, ad esempio, la provenienza dei clienti, l'età media, i prodotti più richiesti e quelli con i maggior guadagni e così via

INSERT INTO persona VALUES ('FRRLRD93N48A341E', 'Loredana', 'Ferrante',
date('08/01/1993'), 'Italia', 3000, 'dirigente', true, true);


Inserimento evento nel turno: UPDATE turno SET cod_evento=2 WHERE cod_turno=1

Elimina dipendente da un turno: DELETE FROM lavora WHERE cf='ABDSDF92R42A342B'
AND cod_turno=1

Nazionalità più frequente: SELECT nazionalita, count(*) as num FROM persona GROUP
BY nazionalita ORDER BY num DESC;

Età media clienti: SELECT AVG(date_part('year', age(data_nascita))) as eta_media
FROM persona WHERE s2=true;

Prodotti più richiesti: SELECT prodotto.nome, presenta.qta FROM prodotto, ordine,
presenta WHERE presenta.cod_prodotto = prodotto.cod_prodotto AND
presenta.cod_ordine = ordine.cod_ordine ORDER BY presenta.qta DESC;



Prodotti più lucrosi: SELECT prodotto.nome,
(presenta.qta*prezzo) as tot
FROM prodotto, ordine, presenta
WHERE presenta.cod_prodotto = prodotto.cod_prodotto
AND presenta.cod_ordine = ordine.cod_ordine
ORDER BY tot DESC;

Spesa totale per cliente: SELECT persona.cf, sum(totale) as tot
FROM persona, ordine
WHERE ordine.cf = persona.cf
GROUP BY persona.cf ORDER BY tot DESC;

Cliente con più ordini: SELECT persona.cf, count(*) as num_ordini
FROM persona, ordine
WHERE ordine.cf = persona.cf
GROUP BY persona.cf
ORDER BY num_ordini DESC LIMIT 1

Altre note

- ▶ Come si sarà notato nello schema ER e nell'implementazione, abbiamo aggiunto dei dati derivati e costruito degli indici per rendere più efficiente il database.
- ▶ I dati derivati che abbiamo aggiunto sono i campi **totale** e **sconto** entrambi riferiti alla tabella **ordine** che tengono conto della spesa totale di quell'ordine e dello sconto applicato su di esso, rispettabilmente, visto che dai conti da noi effettuati risultano convenienti sulle query per l'inserimento di un ordine ed il recupero del totale o dello sconto.
- ▶ Gli indici che abbiamo creato sono sul campo **cf** di **persona**, **cod_ordine** di **ordine** e **cod_prodotto** di **prodotto**. Ci sono risultati convenienti sulle query per il recupero delle credenziali di una persona (login), per il recupero di un ordine dato un codice fiscale ed il recupero dei prodotti di un dato ordine.