



**RELAZIONE DEL PROGETTO
CORSO BASI DI DATI E LABORATORIO
ANNO 2022/2023**

COMPONENTI DEL PROGETTO:

ADAMO ADRIANO 0124002669

AMBROSIO FERDINANDO 0124002668

DATA CONSEGNA:

16/07/2023

CATEGORIA:

ATTIVITA' COMMERCIALE

TITOLO DEL PROGETTO:

DELIVERY VESUVIANO



INDICE

Sommario

PREMESSA.....	4
PROGETTAZIONE.....	5
DIAGRAMMA EE/R.....	6
UTENTI E LE LORO CATEGORIE.....	7
VINCOLI DI INTEGRITA'	8
VERIFICA DELLA NORMALITA'	9
IMPLEMENTAZIONE.....	10
TRIGGER.....	23
PROCEDURE	30
Viste	32

PREMESSA

L'idea di questo progetto è nata grazie alle nostre esperienze lavorative passate, entrambi i componenti hanno lavorato per diverse aziende di delivery locali, prendendo spunto da questi, abbiamo cercato di realizzare la nostra idea di "delivery Vesuviano". A differenza delle grandi aziende di delivery, che operano in grandi città, il nostro delivery si deve adattare alle esigenze della periferia, per questo abbiamo deciso di dividere i vari comuni in "zone", per ottimizzare le consegne e garantire tempi di consegna ragionevoli per i rider e i clienti. I comuni limitrofi sono raggruppati in un'unica zona per dare la possibilità ai rider di non percorrere una distanza troppo lunga dal punto di ritiro al punto di consegna. L'aspetto che abbiamo preso in considerazione nel nostro delivery è quella di verificare l'aspetto esecutivo, cioè la gestione di clienti, rider e locali che formano un ordine. Il rider ha anche la possibilità di decidere quando lavorare attraverso la prenotazione.

PROGETTAZIONE

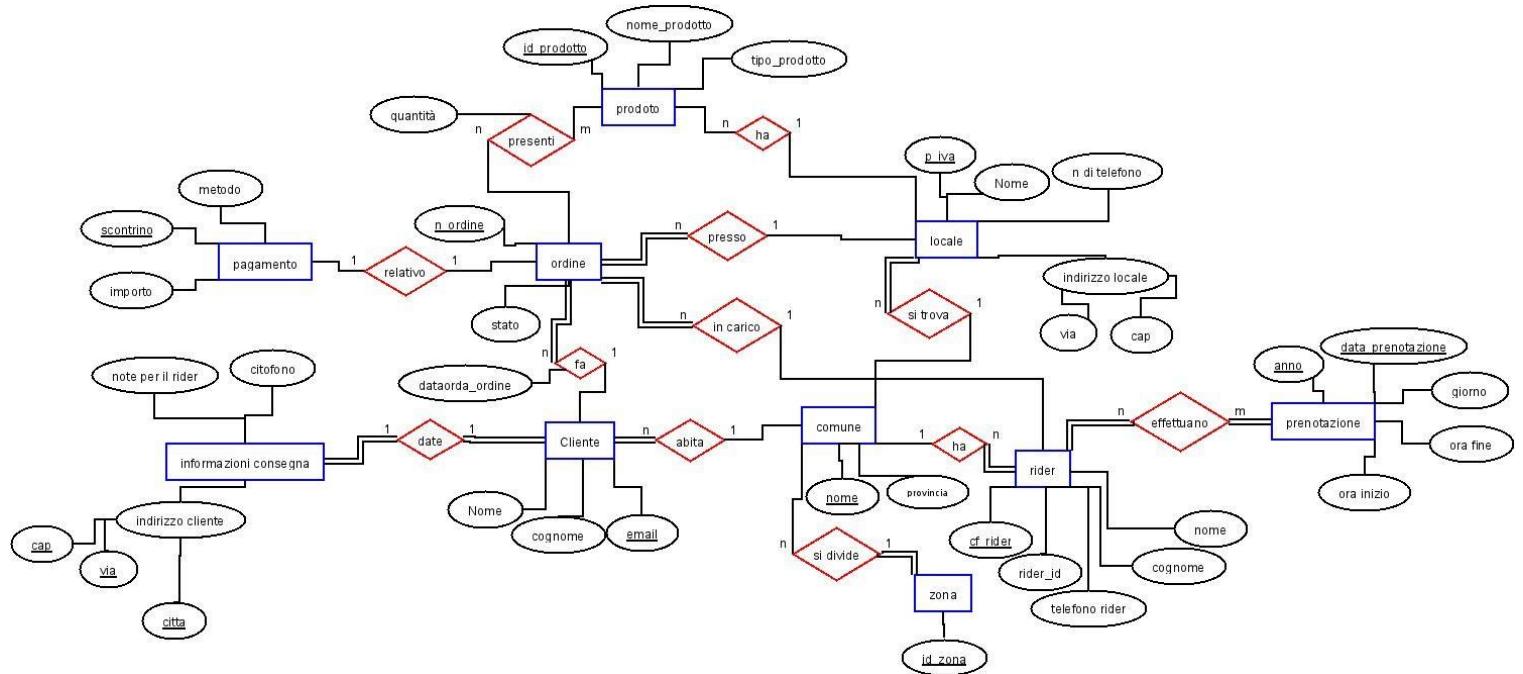
DIAGRAMMA EE/R

Il nostro database deve gestire maggiormente:

- Clienti;
- Locali;
- Rider;
- Ordini;
- Prenotazioni Rider;

I nostri clienti, dopo aver effettuato la registrazione tramite e-mail, ordinano comodamente dalla propria abitazione e hanno la possibilità di scegliere i locali situati nella propria zona. Una volta effettuato l'ordine, verrà assegnato al primo rider libero che lavora nella stessa zona. Una volta creato l'ordine si associa il numero ordine alla partita iva del locale e al codice fiscale del rider, in modo da avere un quadro chiaro su chi ordine, dove ordina, e chi consegna. I rider avranno la possibilità di prenotarsi contribuendo a garantire una copertura adeguata delle richieste di consegna.

DIAGRAMMA EE/R



UTENTI E LE LORO CATEGORIE

UTENTI

ADMIN:

```
CREATE USER RESPONSABILE_ADMIN IDENTIFIED BY ADMIN; GRANT CONNECT, RESOURCE, DBA,  
TO RESPONSABILE_ADMIN;
```

```
GRANT ALL PRIVILEGES TO RESPONSABILE_ADMIN;
```

Il responsabile ha tutti i permessi affinché possa accedere a qualsiasi aspetto del delivery.

RIDER:

```
CREATE USER RIDER_USER IDENTIFIED BY PASSWORD_RIDER;
```

```
GRANT CREATE SESSION TO RIDER_USER; GRANT SELECT ON INFORMAZIONI_CONSEGNA TO  
RIDER_USER; GRANT SELECT ON LOCALE TO RIDER_USER;
```

```
GRANT SELECT ON ORDINE TO RIDER_USER;
```

```
GRANT SELECT ON PAGAMENTO TO RIDER_USER;
```

```
GRANT INSERT, UPDATE ON EFFETTUANO TO RIDER_USER;
```

CLIENTE

```
CREATE USER CLIENTE_USER IDENTIFIED BY PASSWORD_CLIENTE;
```

```
GRANT CONNECT TO CLIENTE_USER;
```

```
GRANT CREATE SESSION TO CLIENTE_USER;
```

```
GRANT SELECT ON LOCALE TO CLIENTE_USER;
```

```
GRANT SELECT,INSERT,UPDATE ON INFORMAZIONI_CONSEGNA TO CLIENTE_USER;
```

```
GRANT SELECT ON PAGAMENTO TO CLIENTE_USER;
```

```
GRANT SELECT PRODOTTO TO CLIENTE_USER;
```

```
GRANT SELECT CLIENTE_VIEW TO CLIENTE_USER;
```

Il cliente può accedere al locale, può inserire e visionare le sue informazioni di consegna, può vedere il pagamento, e può verificare i prodotti che ha acquistato, inoltre abbiamo creato una view dove può visionare tutte le informazioni relative all'ordine.

VINCOLI DI INTEGRITA'

- 1) L'ora dell'inizio del turno deve precedere l'ora della fine del turno;**
- 2) Il turno è dalle ore 19:00 alle ore 00:00;**
- 3) Il metodo di pagamento deve essere necessariamente o con carta o in contanti;**
- 4) I giorni devono essere per forza dal lunedì alla domenica;**
- 5) Lo stato dell'ordine può essere: "in lavorazione", "ritirato", "in consegna", "consegnato";**

VERIFICA DELLA NORMALITA'

PRIMA FORMA NORMALE E' STATA RISPETTATA:

Poiché tutti i campi della tabella sono costituiti da valori atomici. Le uniche eccezioni sono i campi di tipo DATE e TIMESTAMP; tuttavia, questi ultimi sono considerati atomici nel DBMS Oracle.

SECONDA FORMA NORMALE NON E' STATA RISPETTATA:

Poiché esiste una sola chiave multi-attributo

TERZA FORMA NORMALE E' STATA RISPETTATA:

Poiché nessuna entità dipende dall'esistenza di un'altra

IMPLEMENTAZIONE

DDL

—Zona

```
create table zona(
    id_zona char(4),
    constraint pk_zona primary key(id_zona)
);
```

—Comune

```
create table comune(
    nome_comune char(20),
    provincia char(2),
    zona_associata char(4),
    constraint pk_comune primary key(nome_comune),
    constraint fk_comune foreign key (zona_associata) references zona(id_zona)
);
```

—Locale

```
create table locale(
    p_iva char(11),
    nome_locale varchar(20) unique not null,
    n_telefono_locale char(10) unique not null,
    via_locale varchar(20) not null,
    cap_locale varchar(20) not null,
    comune_sede char(20),
    constraint pk_locale primary key(p_iva),
```

```
constraint fk_locale foreign key (comune_sede) references  
comune(nome_comune)  
);
```

—Rider

```
create table rider(  
    cf_rider char(16),  
    nome_rider varchar(20) not null,  
    cognome_rider varchar(20) not null,  
    rider_id number(4,0) not null unique,  
    telefono_rider char(10) unique,  
    comune_lavorativo char(20),  
    constraint pk_rider primary key(cf_rider),  
    constraint fk_rider foreign key (comune_lavorativo) references  
comune(nome_comune));
```

—Prenotazione

```
CREATE TABLE prenotazione (  
    anno number(4,0) not null,  
    data_prenotazione date,  
    giorno varchar(20) not null,  
    ora_inizio timestamp not null,  
    ora_fine timestamp not null,  
    constraint pk_prenotazione primary key (data_prenotazione, anno, giorno),  
    constraint ck_orario_prenotazione check (  
        (EXTRACT(hour FROM ora_inizio) >= 19 AND EXTRACT(hour from ora_fine) <=  
24)  
        AND
```

```
(EXTRACT(hour FROM ora_inizio) < EXTRACT(hour FROM ora_fine))),  
constraint check_giorno check (giorno in ('lunedì', 'martedì', 'mercoledì', 'giovedì',  
'venerdì', 'sabato', 'domenica'))  
);
```

—Cliente

```
create table cliente(  
    nome_cliente varchar(20) not null,  
    cognome_cliente varchar(20),  
    email char(30) not null,  
    comune_res char(20),  
    constraint pk_cliente primary key(email),  
    constraint fk_cliente1 foreign key(comune_res) references  
comune(nome_comune),  
);
```

—Informazioni consegna

```
create table informazioni_Consegna(  
    via_cliente char(20) not null,  
    citta_cliente char(20) not null,  
    cap_cliente char(10) not null,  
    citofono varchar(20),  
    note_per_rider varchar(30),  
    email_cliente char(30),  
    constraint pk_info primary key(via_cliente,citta_cliente,cap_cliente),  
    constraint fk_info foreign key(email_cliente) references cliente(email)  
);
```

—Pagamento

```
create table pagamento(
    scontrino char(10),
    importo number(2,0),
    metodo_pay varchar(30),
    constraint pk_pagamento primary key(scontrino),
    constraint chk_metodo check(
        metodo_pay='contanti'
        or metodo_pay='Carta di credito/debito')
);
```

—Ordine

```
create table ordine(
    n_ordine number(3,0) not null,
    stato varchar(20) not null,
    p_ivalocale char(11),
    cf_consegnatore char(16),
    email_destinatario char(30),
    scontrino_ordine char(10),
    data_e_ora timestamp,
    constraint pk_ordine primary key(n_ordine),
    constraint fk_ordine_locale foreign key(p_ivalocale) references locale(p_iva),
    constraint fk_ordine_rider foreign key(cf_consegnatore) references rider(cf_rider),
    constraint fk_ordine_cliente foreign key (email_destinatario) references cliente(email),
```

```
constraint fk_ordine_scontrino foreign key(scontrino_ordine) references
pagamento(scontrino),
constraint chk_stato check(
stato in('in lavorazione','ritirato','in consegna','consegnato'))
);
```

—Prodotto

```
create table prodotto(
id_prodotto number not null,
nome_prodotto varchar(20) not null,
tipo_prodotto varchar(20),
p_iva_locale char(11),
constraint pk_prodotto primary key(id_prodotto),
constraint fk_prdotto foreign key (p_iva_locale) references locale(p_iva)
);

```

—Presenti

```
create table presenti(
idProdotto number,
nOrdine number,
quantità number(2,0),
constraint pk_presenti primary key(idProdotto,nOrdine),
constraint fk_presenti1 foreign key(idProdotto) references prodotto(id_prodotto),
constraint fk_presenti2 foreign key(nOrdine) references ordine(n_ordine)
);

```

—Effettuano

```
create table effettuano(
```

```
codf_rider char(16),
anno1 number(4,0),
dataPrenotazione date,
giorno_lavorativo varchar(20),
constraint pk_effettuano primary
key(codf_rider,anno1,dataPrenotazione,giorno_lavorativo),
constraint fk_effettuano1 foreign key (codf_rider) references rider(cf_rider),
constraint fk_effettuano2 foreign key (anno1,dataPrenotazione,giorno_lavorativo)
references prenotazione(anno,data_prenotazione,giorno)
);
```

DML

--Zona

```
insert into zona(id_zona)
```

```
values(1);
```

```
insert into zona(id_zona)
```

```
values(2);
```

```
insert into zona(id_zona)
```

```
values(3);
```

```
insert into zona(id_zona)
```

```
values(4);
```

--Comune

```
insert into comune(nome_comune,provincia,zona_associata)
```

```
values('striano','na',1);
```

```
insert into comune(nome_comune,provincia,zona_associata)
```

```
values('sarno','sa',2);
```

```
insert into comune(nome_comune,provincia,zona_associata)
```

```
values('palma campania','na',1);
```

```
insert into comune(nome_comune,provincia,zona_associata)
```

```
values('poggiomarino','na',1);
```

```
insert into comune(nome_comune,provincia,zona_associata)
```

```
values('ottaviano','na',3);
```

```
insert into comune(nome_comune,provincia,zona_associata)
```

```
values('san giuseppe vsv.','na',3);
```

```
insert into comune(nome_comune,provincia,zona_associata)
```

```
values('san valentino torio','sa',4);
```

--Prenotazione

```
INSERT INTO prenotazione (anno, data_prenotazione, giorno, ora_inizio, ora_fine)
VALUES (2023, TO_DATE('2023-07-10','YYYY-MM-DD'), 'lunedì',
TO_TIMESTAMP('2023-07-10 19:30:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-07-10 23:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO prenotazione (anno, data_prenotazione, giorno, ora_inizio, ora_fine)
VALUES (2023, TO_DATE('2023-07-11', 'YYYY-MM-DD'), 'martedì',
TO_TIMESTAMP('2023-07-11 19:30:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-07-11 23:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO prenotazione (anno, data_prenotazione, giorno, ora_inizio, ora_fine)
VALUES (2023, TO_DATE('2023-07-12','YYYY-MM-DD'), 'mercoledì',
TO_TIMESTAMP('2023-07-12 19:30:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-07-12 23:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO prenotazione (anno, data_prenotazione, giorno, ora_inizio, ora_fine)
VALUES (2023, TO_DATE('2023-07-13','YYYY-MM-DD'), 'giovedì',
TO_TIMESTAMP('2023-07-13 19:30:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-07-13 23:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO prenotazione (anno, data_prenotazione, giorno, ora_inizio, ora_fine)
VALUES (2023, TO_DATE('2023-07-14', 'YYYY-MM-DD'), 'venerdì',
TO_TIMESTAMP('2023-07-14 19:15:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-07-14 23:45:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO prenotazione (anno, data_prenotazione, giorno, ora_inizio, ora_fine)
VALUES (2023, TO_DATE('2023-07-15', 'YYYY-MM-DD'), 'sabato',
TO_TIMESTAMP('2023-07-15 19:15:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-07-15 23:45:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO prenotazione (anno, data_prenotazione, giorno, ora_inizio, ora_fine)
VALUES (2023, TO_DATE('2023-07-16', 'YYYY-MM-DD'), 'domenica',
TO_TIMESTAMP('2023-07-16 19:15:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-07-16 23:45:00', 'YYYY-MM-DD HH24:MI:SS'));
```

--Rider

```
INSERT INTO rider (cf_rider, nome_rider, cognome_rider, telefono_rider,
comune_lavorativo)
VALUES ('CF001', 'Massimo', 'Verdi', '1234567890', 'palma campania');

INSERT INTO rider (cf_rider, nome_rider, cognome_rider, telefono_rider,
comune_lavorativo)
VALUES ('CF002','Nello','Gallo','0987654321','terzigno');

INSERT INTO rider (cf_rider, nome_rider, cognome_rider, telefono_rider,
comune_lavorativo)
VALUES ('CF003','Pasquale','Piccolo','4563738901','poggiomarino');

INSERT INTO rider (cf_rider, nome_rider, cognome_rider, telefono_rider,
comune_lavorativo)
VALUES ('CF004','Maria','Nappi','4123998800','ottaviano');

INSERT INTO rider (cf_rider, nome_rider, cognome_rider, telefono_rider,
comune_lavorativo)
VALUES ('CF005','Carlo','De Filippo','1230987444','sarno');

INSERT INTO rider (cf_rider, nome_rider, cognome_rider, telefono_rider,
comune_lavorativo)
VALUES ('CF006','Alberto','Ammirati','0932177250','san giuseppe vsv.');

INSERT INTO rider (cf_rider, nome_rider, cognome_rider, telefono_rider,
comune_lavorativo)
VALUES ('CF007','Michele','Annunziata','0817720112','san valentino torio');
```

--Locale

```
INSERT INTO locale (p_iva, nome_locale, n_telefono_locale,
via_locale,cap_locale,comune_sede)
VALUES ('12345678901', 'Ristorante Bella','0123456789', 'Via Roma
1','84047','sarno');
```

```
INSERT INTO locale (p_iva, nome_locale, n_telefono_locale, via_locale,
cap_locale,comune_sede)
VALUES ('23456789012', 'Bar La Piazzetta', '1234567890', 'Via Milano 2', '80047','san
giuseppe vsv.');

INSERT INTO locale (p_iva, nome_locale, n_telefono_locale, via_locale,
cap_locale,comune_sede)
VALUES ('34567890123', 'Pizzeria Da Giovanni', '2345678901', 'Via Napoli 3', '80040',
'poggiomarino');

INSERT INTO locale (p_iva, nome_locale, n_telefono_locale, via_locale,
cap_locale,comune_sede)
VALUES ('45678901234', 'Osteria Buonaiuto', '3456789012', 'Via Torino
4','80036','palma campania');

INSERT INTO locale (p_iva, nome_locale, n_telefono_locale, via_locale,
cap_locale,comune_sede)
VALUES ('67890123456', 'Sushi Bar Sakura', '5678901234', 'Via Bologna 6',
'80044','ottaviano');

INSERT INTO locale (p_iva, nome_locale, n_telefono_locale, via_locale,
cap_locale,comune_sede)
VALUES ('89012345678', 'Seafood Restaurant', '7890123456', 'Via Genova 8', '84010'
,'san valentino torio');

--Cliente

INSERT INTO cliente (nome_cliente, cognome_cliente, email, comune_res)
VALUES ('Giulia', 'Verdi', 'giulia@email.com', 'striano');

    INSERT INTO cliente (nome_cliente, cognome_cliente, email, comune_res)
VALUES ('Paola', 'Gialli', 'paola@email.com', 'sarno');

        INSERT INTO cliente (nome_cliente, cognome_cliente, email, comune_res)
VALUES('Alessandro', 'Marroni', 'alessandro@email.com', 'palma campania');

            INSERT INTO cliente (nome_cliente, cognome_cliente, email, comune_res)
```

```
VALUES ('Federica', 'Rosa', 'federica@email.com', 'sarno');

INSERT INTO cliente (nome_cliente, cognome_cliente, email, comune_res)
VALUES ('Giovanni', 'Arancio', 'giovanni@email.com', 'poggiomarino');

INSERT INTO cliente (nome_cliente, cognome_cliente, email, comune_res)
VALUES ('Stefania', 'Viola', 'stefania@email.com', 'san giuseppe vsv.');

INSERT INTO cliente (nome_cliente, cognome_cliente, email, comune_res)
VALUES ('Martina', 'Azzurro', 'martina@email.com', 'palma campania');

INSERT INTO cliente (nome_cliente, cognome_cliente, email, comune_res)
VALUES ('Mario', 'Rossi', 'mario@email.com', 'striano');
```

--**Pagamento**

```
INSERT INTO pagamento (scontrino, importo, metodo_pay)
VALUES ('6', 55, 'Carta di credito/debito');

INSERT INTO pagamento (scontrino, importo, metodo_pay)
VALUES ('7', 12, 'contanti');

INSERT INTO pagamento (scontrino, importo, metodo_pay)
VALUES ('8', 18, 'Carta di credito/debito');

INSERT INTO pagamento (scontrino, importo, metodo_pay)
VALUES ('2', 28, 'Carta di credito/debito');
```

--**Informazioni consegna**

```
INSERT INTO informazioni_Consegna (via_cliente, citta_cliente, cap_cliente,
email_cliente)
VALUES ('via roma,8', 'san valentino torio', '84010', 'luca@email.com');

INSERT INTO informazioni_Consegna (via_cliente, citta_cliente, cap_cliente,
email_cliente)
VALUES ('Via Torino 1 ', 'sarno', '84087', 'paola@email.com');
```

```
INSERT INTO informazioni_Consegna (via_cliente, citta_cliente, cap_cliente, email_cliente)
```

```
VALUES ('via passanti 8', 'poggiomarino', '80040', 'giovanni@email.com');
```

--Ordine

```
INSERT INTO ordine (n_ordine, stato, p_ivalocale, cf_consegnatore, email_destinatario, scontrino_ordine, data_e_ora)
```

```
VALUES (1, 'consegnato', '45678901234', 'CF001', 'alessandro@email.com ', '8 ', CURRENT_TIMESTAMP);
```

```
INSERT INTO ordine (n_ordine, stato, p_ivalocale, cf_consegnatore, email_destinatario, scontrino_ordine, data_e_ora)
```

```
VALUES (2, 'in consegna', '2345678901', 'CF003', 'giovanni@email.com ', '2 ', CURRENT_TIMESTAMP);
```

```
INSERT INTO ordine (n_ordine, stato, p_ivalocale, cf_consegnatore, email_destinatario, scontrino_ordine, data_e_ora)
```

```
VALUES (3, 'in lavorazione', '3456789012', 'CF001', 'alessandro@email.com ', '6', CURRENT_TIMESTAMP);
```

```
INSERT INTO ordine (n_ordine, stato, p_ivalocale, cf_consegnatore, email_destinatario, scontrino_ordine, data_e_ora)
```

```
VALUES (4, 'in consegna', '0123456789', 'CF005', 'paola@email.com ', '7 ', CURRENT_TIMESTAMP);
```

--Prodotto

```
INSERT INTO prodotto (nome_prodotto, tipo_prodotto, p_iva_locale)
```

```
VALUES('pizza','diavola','34567890123');
```

```
INSERT INTO prodotto (nome_prodotto, tipo_prodotto, p_iva_locale)
```

```
VALUES('pizza','diavola','34567890123');
```

```
INSERT INTO prodotto (nome_prodotto, tipo_prodotto, p_iva_locale)
```

```
VALUES('pizza','marinara','34567890123');
```

```
INSERT INTO prodotto (nome_prodotto, tipo_prodotto, p_iva_locale)
VALUES('pasta','pennette al pomodoro','12345678901');
```

```
INSERT INTO prodotto (nome_prodotto, tipo_prodotto, p_iva_locale)
VALUES('pasta','carbonara','12345678901');
```

--Presenti

```
Insert into presenti(idProdotto,nOrdine,quantità)
values(2,1,3)
```

```
insert into presenti(idProdotto,nOrdine,quantità)
values(4,2,3)
```

--Effettuano

```
INSERT INTO effettuano (codf_rider, anno1, dataPrenotazione,giorno_lavorativo)
VALUES ('CF001', 2023, TO_DATE('2023-07-15', 'YYYY-MM-DD'),'sabato');
```

```
INSERT INTO effettuano (codf_rider, anno1, dataPrenotazione,giorno_lavorativo)
VALUES ('CF003', 2023, TO_DATE('2023-07-10', 'YYYY-MM-DD'),'lunedì');
```

```
INSERT INTO effettuano (codf_rider, anno1, dataPrenotazione,giorno_lavorativo)
VALUES ('CF005', 2023, TO_DATE('2023-07-10', 'YYYY-MM-DD'),'lunedì');
```

TRIGGER

- 1) In questo trigger specifichiamo che, dal lunedì al giovedì, dato che c'è meno richiesta, abbiamo bisogno di conseguenza di pochi rider rispetto ai giorni del weekend.

```
CREATE OR REPLACE TRIGGER limita_prenotazioni
```

```
BEFORE INSERT ON effettuano
```

```
FOR EACH ROW
```

```
DECLARE
```

```
lunedì_count number;
```

```
martedì_count number;
```

```
mercoledì_count number;
```

```
giovedì_count number;
```

```
venerdì_count number;
```

```
sabato_count number;
```

```
domenica_count number;
```

```
BEGIN
```

```
IF :NEW.giorno_lavorativo = 'lunedì' THEN
```

```
    SELECT COUNT(*)
```

```
    INTO lunedì_count
```

```
    FROM effettuano
```

```
    WHERE anno1 = EXTRACT(YEAR FROM :NEW.dataPrenotazione)
```

```
    AND giorno_lavorativo = 'lunedì';
```

```
    IF lunedì_count >= 5 THEN
```

```
        RAISE_APPLICATION_ERROR(-20001, 'È stato raggiunto il limite massimo di prenotazioni per il lunedì.');
```

```
    END IF;
```

```
ELSIF :NEW.giorno_lavorativo = 'martedì' THEN
```

```

SELECT COUNT(*)
INTO martedì_count
FROM effettuano
WHERE anno1 = EXTRACT(YEAR FROM :NEW.dataPrenotazione)
AND giorno_lavorativo = 'martedì';

IF martedì_count >= 5 THEN
    RAISE_APPLICATION_ERROR(-20002, 'È stato raggiunto il limite massimo di prenotazioni per
il martedì.');
END IF;

ELSIF :NEW.giorno_lavorativo = 'mercoledì' THEN
    SELECT COUNT(*)
    INTO mercoledì_count
    FROM effettuano
    WHERE anno1 = EXTRACT(YEAR FROM :NEW.dataPrenotazione)
    AND giorno_lavorativo = 'mercoledì';

    IF mercoledì_count >= 5 THEN
        RAISE_APPLICATION_ERROR(-20003, 'È stato raggiunto il limite massimo di prenotazioni per
il mercoledì.');
    END IF;

ELSIF :NEW.giorno_lavorativo = 'giovedì' THEN
    SELECT COUNT(*)
    INTO giovedì_count
    FROM effettuano
    WHERE anno1 = EXTRACT(YEAR FROM :NEW.dataPrenotazione)
    AND giorno_lavorativo = 'giovedì';

    IF giovedì_count >= 5 THEN

```

```
RAISE_APPLICATION_ERROR(-20004, 'È stato raggiunto il limite massimo di prenotazioni per  
il giovedì.');
```

```
END IF;
```

```
ELSIF :NEW.giorno_lavorativo = 'venerdì' THEN
```

```
    SELECT COUNT(*)
```

```
    INTO venerdi_count
```

```
    FROM effettuano
```

```
    WHERE anno1 = EXTRACT(YEAR FROM :NEW.dataPrenotazione)
```

```
        AND giorno_lavorativo = 'venerdì';
```



```
IF venerdi_count >= 10 THEN
```

```
    RAISE_APPLICATION_ERROR(-20005, 'È stato raggiunto il limite massimo di prenotazioni per  
il venerdì.');
```

```
END IF;
```

```
ELSIF :NEW.giorno_lavorativo = 'sabato' THEN
```

```
    SELECT COUNT(*)
```

```
    INTO sabato_count
```

```
    FROM effettuano
```

```
    WHERE anno1 = EXTRACT(YEAR FROM :NEW.dataPrenotazione)
```

```
        AND giorno_lavorativo = 'sabato';
```



```
IF sabato_count >= 10 THEN
```

```
    RAISE_APPLICATION_ERROR(-20006, 'È stato raggiunto il limite massimo di prenotazioni per  
il sabato.');
```

```
END IF;
```

```
ELSIF :NEW.giorno_lavorativo = 'domenica' THEN
```

```
    SELECT COUNT(*)
```

```
    INTO domenica_count
```

```
    FROM effettuano
```

```
    WHERE anno1 = EXTRACT(YEAR FROM :NEW.dataPrenotazione)
```

```
AND giorno_lavorativo = 'domenica';

IF domenica_count >= 10 THEN
    RAISE_APPLICATION_ERROR(-20007, 'È stato raggiunto il limite massimo di prenotazioni per
la domenica.');
END IF;

END IF;

END;
```

2) Il secondo trigger è quello di auto-incrementare il rider_id:

```
CREATE SEQUENCE seq_rider_id START WITH 1 INCREMENT BY 1;
CREATE OR REPLACE TRIGGER Trg_IncrementaRiderId
BEFORE INSERT ON rider
FOR EACH ROW
BEGIN
    SELECT seq_rider_id.NEXTVAL INTO :new.rider_id FROM dual;
END;
```

3) Per far sì che un rider non abbia troppi contanti con se, impostiamo un limite di importo pagabile in contanti:

```
CREATE OR REPLACE TRIGGER obbliga_pagamento_carta
BEFORE INSERT ON pagamento
FOR EACH ROW
BEGIN
IF :NEW.metodo_pay != 'Carta di credito/debito' AND :NEW.importo > 35 THEN
    RAISE_APPLICATION_ERROR(-20008, 'L"importo supera i 35 euro. È obbligatorio pagare con carta di credito/debito.');
END IF;
END;
```

4) Questo trigger si occupa del ritiro e della consegna nella stessa zona(comune)

```
CREATE OR REPLACE TRIGGER Trg_RitiraConsegna
BEFORE INSERT ON ordine
FOR EACH ROW
DECLARE
v_comune_rider comune.nome_comune%TYPE;
v_comune_locale locale.comune_sede%TYPE;
v_comune_destinatario cliente.comune_res%TYPE;
BEGIN
SELECT comune_lavorativo INTO v_comune_rider FROM rider WHERE cf_rider =
:new.cf_consegnatore;
SELECT comune_sede INTO v_comune_locale FROM locale WHERE p_iva = :new.p_ivalocale;
SELECT comune_res INTO v_comune_destinatario FROM cliente WHERE email =
:new.email_destinatario;
IF v_comune_rider <> v_comune_locale OR v_comune_rider <> v_comune_destinatario THEN
    raise_application_error(-20009, 'Il rider può ritirare al locale e consegnare al cliente solo nello STESSO comune.');
END IF;
```

END IF;

END;

5) QUESTO TRIGGER CONSENTE DI FARE UN ORDINE MINIMO DI ALMENO 10 euro

CREATE OR REPLACE TRIGGER Trg_ControlloImportoMinimo

BEFORE INSERT ON pagamento

FOR EACH ROW

BEGIN

IF :new.importo <= 10 THEN

 raise_application_error(-20010, 'L"importo del pagamento deve essere maggiore di 10.');

END IF;

EXCEPTION

WHEN OTHERS THEN

 DBMS_OUTPUT.PUT_LINE('Errore: ' || SQLERRM)

 RAISE;

END;

6) Questo trigger verifica se il rider è impegnato non può ricevere un nuovo ordine, appena sarà libero lo potrà ricevere.

CREATE OR REPLACE TRIGGER trg_ordine_before_insert

BEFORE INSERT ON ordine

FOR EACH ROW

DECLARE

 v_rider_stato VARCHAR(20);

BEGIN

 SELECT stato

 INTO v_rider_stato

```
FROM ordine  
WHERE cf_consegnatore = :NEW.cf_consegnatore  
ORDER BY data_e_ora DESC  
FETCH FIRST 1 ROW ONLY;  
  
IF v_rider_stato IN ('in lavorazione', 'ritirato', 'in consegna') THEN  
    RAISE_APPLICATION_ERROR(-20011, 'Il rider è già occupato con la consegna di un altro ordine.');
```

END IF;

EXCEPTION

```
WHEN NO_DATA_FOUND THEN  
    NULL;  
END;
```

PROCEDURE

1) Questa procedura indica quante consegne ha effettuato il singolo rider.

```
CREATE OR REPLACE PROCEDURE NUMERO_CONSEGNE_RIDER (CF RIDER IN CHAR)
IS
CONT INTEGER;
BEGIN
SELECT COUNT(*) INTO CONT
FROM ordine
WHERE cf_consegnatore = CF RIDER;
DBMS_OUTPUT.PUT_LINE('Il rider con CF ' || CF RIDER || ' ha effettuato ' || CONT || '
consegne.');
END;
```

**2) Questa procedura conta quanti ordini per comune sono stati effettuati,
questa procedura è molto utile per capire in che comune la richiesta è
superiore rispetto ad altri.**

```
CREATE OR REPLACE PROCEDURE NUMERO_ORDINI_PER_COMUNE
IS
BEGIN
FOR rec IN (SELECT comune.nome_comune, COUNT(*) AS num_ordini
            FROM comune
            INNER JOIN cliente ON comune.nome_comune = cliente.comune_res
            INNER JOIN ordine ON cliente.email= ordine.email_destinatario
            GROUP BY comune.nome_comune)
LOOP
DBMS_OUTPUT.PUT_LINE('Comune: ' || rec.nome_comune || ', Numero ordini: ' ||
rec.num_ordini);
END LOOP;
END;
```

3) Questa procedura calcola tutti gli importi generati dagli ordini

```
CREATE OR REPLACE PROCEDURE CalcolaGuadagni
IS
BEGIN
    FOR rec IN (SELECT SUM (importo) AS totale_pagamenti
                FROM pagamento
                INNER JOIN ordine ON ordine.scontrino_ordine = pagamento.scontrino
                GROUP BY ordine.scontrino_ordine)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Totale pagamenti: ' || rec.totale_pagamenti);
    END LOOP;
END;
```

4) Questa procedura conta il numero di prenotazioni per ogni rider, ciò ci consente di capire quanto un rider effettivamente lavora.

```
CREATE OR REPLACE PROCEDURE RIDER_PRENOTATI
IS
BEGIN
    FOR rec IN (
        SELECT effettuano.codf_rider, COUNT(*) AS num_prenotazioni
        FROM effettuano
        INNER JOIN rider ON effettuano.codf_rider=rider.cf_rider
        GROUP BY effettuano.codf_rider
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Rider:' || rec.codf_rider || ', Numero di prenotazioni: '
        || rec.num_prenotazioni);
    END LOOP;
END;
```

Viste

1) Questa vista ci permette di vedere il rider prenotato in che giorno.

```
CREATE VIEW prenotazionerider AS  
SELECT rider.cf_rider,prenotazione.data_prenotazione  
FROM rider  
INNER JOIN effettuano on rider.cf_rider=effettuano.codf_rider  
INNER JOIN prenotazione on  
effettuano.dataPrenotazione=prenotazione.data_prenotazione;
```

2) Questa vista permette al cliente di vedere tutte le informazioni relative all'ordine

```
CREATE VIEW cliente_view AS  
SELECT locale.nome_locale,ordine.n_ordine,ordine.stato,pagamento.importo  
FROM locale  
inner join ordine on locale.p_iva=ordine.p_ivalocale  
inner join pagamento on ordine.scontrino_ordine=pagamento.scontrino
```