

# B&B Borbonico



**Da un idea di:**  
**Giuseppe Falco**  
**Alessandro Di Stefano**  
**Emanuela Baldassarre**

**ANNO ACCADEMICO**

**2023–2024**

**DATA DI CONSEGNA:  
GIOVEDÌ 20 GIUGNO 2024**

**STUDENTI:**

0124002276  
ALESSANDRO DI STEFANO  
0124002825  
GIUSEPPE FALCO  
0124003217  
EMANUELA BALDASSARRE

# INDICE

## ↓ PROGETTAZIONE

1.1 sintesi dei requisiti	<b>O 1</b>
1.2 Glossario	<b>1</b>
1.3 diagramma EE/R	<b>2</b>
1.4 Diagramma Relazionale	<b>3A</b>
1.5 Utenti e le loro categorie	<b>3</b>
1.6 Operazioni tra gli utenti	<b>4</b>
1.7 Tavola dei Volumi	<b>5</b>
1.8 Vincoli di integrità	<b>11</b>
1.9 Vincoli di Normalità	<b>12</b>
	<b>14</b>

## ↓ IMPLEMENTAZIONE

2.1 Creazione Utenti	<b>15</b>
2.2 Data definition language	<b>16</b>
2.3 Data Manipulation Language	<b>20</b>
2.4 TRIGGER	<b>24</b>
2.5 Procedure e funzioni	<b>31</b>
	<b>36</b>



# PROGETTAZIONE





# 1.1 SINTESI DEI REQUISITI

Il database sviluppato nasce con l'intento di risolvere il problema riguardante la gestione dei dati del database di un B&B. Offre una soluzione per gestire le prenotazioni, tracciare gli ordini e tutto ciò che riguarda la gestione del personale di servizio.

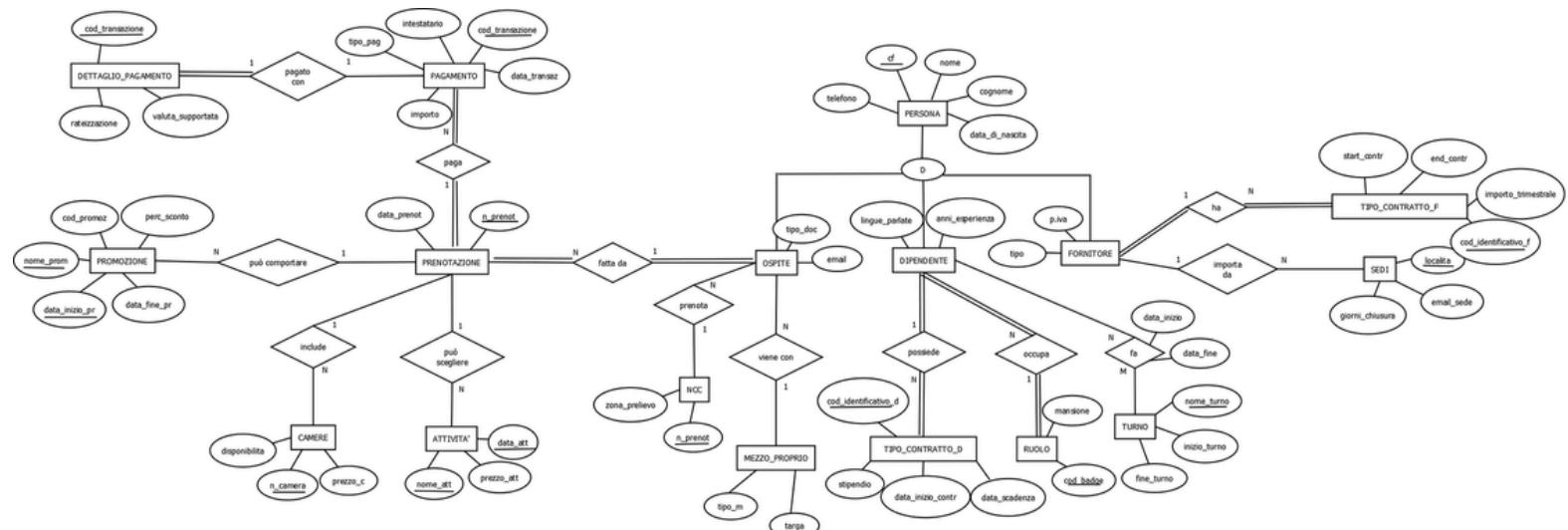
# 1.2 GLOSSARIO

I glossario ha come fine ultimo di chiarire la Terminologia utilizzata in questo database e (anche se assente) il “gergo tecnico”

TERMINE	DEFINIZIONE	SINONIMIO	ONONIMI
NUMERO_PRENOTAZIONE	Con codesto facciamo riferimento alla prenotazione del singolo ospite	-	
NOME_PROMOZIONE	Utilizzata per ricondurci alla tipologia di promozione in combinazione con la data di inizio della stessa		
NOME_ATTIVITA	Utilizzata per ricondurci alla tipologia di attività scelta in combinazione con la data in cui è stata svolta	-	
START_OF_CONTRACT	Inizio contratto forniture	-	
END_OF_CONTRACT	Scadenza contratto forniture	-	
DISPONIBILITA	Verifichiamo tramite un controllo se la camera in questione è disponibile o meno	-	

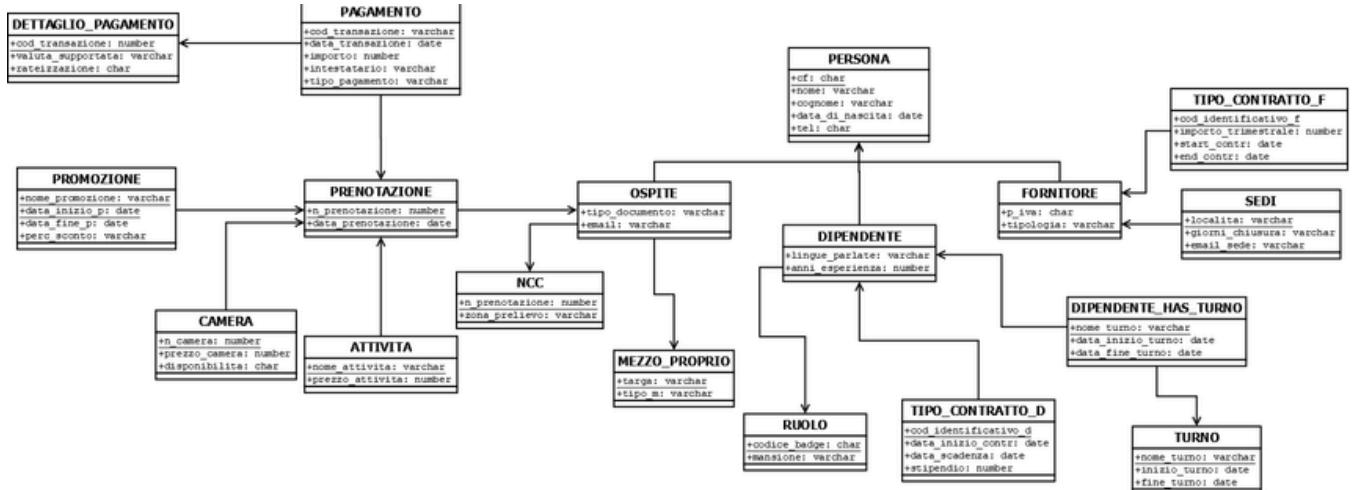
# 1.3 DIAGRAMMA EE/R

Di seguito abbiamo il diagramma EE/R.



# 1.4 DIAGRAMMA RELAZIONALE

Tenendo presente che questo diagramma è meno espressivo rispetto al diagramma EE/R, siamo limitati nella rappresentazione di alcuni aspetti, come la molteplicità tra entità e vincoli degli attributi delle entità. Tuttavia, con questo strumento possiamo entrare più in profondità su come sono state create le tabelle riportate nella sezione Implementazione.



# 1.5 UTENTI E LE LORO CATEGORIE

Presentiamo di seguito i vari tipi di utenti e le loro categorie che si possono identificare all'interno del database:

Utente	Categoria Utente
Admin	Responsabili del database
Gestore	Gestisce il B&B
Ospite	Prenota ed eventualmente può disdire le prenotazioni

# 1.6 OPERAZIONI TRA UTENTI

Le operazioni fra utenti sono operazioni implementate tramite procedure.

Vengono riportate le descrizioni delle varie procedure:

Operazioni	RINNOVA_CONTRATTI
Scopo	Estende la data di scadenza dei contratti di 12 mesi per quelli che scadono nel 2024
Argomento	-
Risultato	Aggiornamento della data di scadenza per i contratti selezionati
Errori	-
Modifica	-UPDATE su TIPO_CONTRATTO_D
Usa	ADD_MONTHS, TO_DATE
Prima	-
Poi	Visualizzazione delle date di scadenza aggiornate dei contratti

Operazioni	CERCA_PRENOTAZIONI OSPITE
Scopo	Cerca e mostra tutte le prenotazioni di un ospite specificato tramite il suo codice fiscale
Argomento	p_CF OSPITE CHAR
Risultato	Mostra il nome, cognome, codice fiscale dell'ospite, numero di prenotazione e data di prenotazione
Errori	-
Modifica	-
Usa	JOIN, SELECT, DBMS_OUTPUT.PUT_LINE
Prima	-
Poi	Visualizzazione delle prenotazioni dell'ospite specificato

<b>Operazioni</b>	<b>CERCA_DIPENDENTI_PER_MANSIONE</b>
Scopo	Cerca e mostra tutti i dipendenti che hanno una determinata mansione
Argomento	p_MANSIONE VARCHAR
Risultato	Mostra il codice fiscale, nome e cognome dei dipendenti che svolgono la mansione specificata
Errori	
Modifica	Indirizzo
Usa	JOIN, SELECT, DBMS_OUTPUT.PUT_LINE
Prima	
Poi	Visualizzazione dei dipendenti con la mansione specificata

Operazioni	CERCA_FORNITORI_PER_TIPOLOGIA
Scopo	Cerca e mostra tutti i fornitori che appartengono a una determinata tipologia
Argomento	p_TIPOLOGIA VARCHAR
Risultato	Mostra il codice fiscale, nome, cognome e località dei fornitori che appartengono alla tipologia specificata
Errori	
Modifica	
Usa	JOIN, SELECT, DBMS_OUTPUT.PUT_LINE
Prima	
Poi	Visualizzazione dei fornitori con la tipologia specificata

<b>Operazioni</b>	<b>CERCA_TURNI_PER_DIPENDENTE</b>
Scopo	Cerca e mostra tutti i turni di un dipendente specificato tramite il suo codice fiscale
Argomento	p_CF_DIPENDENTE CHAR
Risultato	Mostra il nome del turno, l'inizio e la fine del turno, il nome e il cognome del dipendente
Errori	
Modifica	
Usa	JOIN, SELECT, DBMS_OUTPUT.PUT_LINE
Prima	
Poi	Visualizzazione dei turni del dipendente specificato

<b>Operazioni</b>	<b>CERCA_PRENOTAZIONI_PER_ZONA_PRELIEVO</b>
Scopo	Cerca e mostra tutte le prenotazioni per una determinata zona di prelievo
Argomento	p_ZONA_PRELIEVO VARCHAR
Risultato	Mostra il numero di prenotazione, codice fiscale, nome e cognome dell'ospite che ha effettuato la prenotazione
Errori	
Modifica	
Usa	JOIN, SELECT, DBMS_OUTPUT.PUT_LINE
Prima	
Poi	Visualizzazione delle prenotazioni per la zona di prelievo specificata

# 1.7 I VOLUMI

I dati sono stati inseriti nella tabella in modo pseudo-realistico, basandosi su una previsione ottimistica delle buone performance aziendali e di una situazione economica favorevole. La tabella dei volumi è riportata di seguito, mostrando il numero di record presenti in ciascuna tabella e il loro incremento previsto in un certo intervallo di tempo. In questo contesto, si presume che i record più vecchi non vengano rimossi dal database e che tutte le relazioni crescano periodicamente.

TABELLA	TIPO	VOLUME BASE	INCREMENTO	PERIODO
PERSONA	E	VOLUME	21	SETTIMANALE
OSPITE	E	VOLUME	21	SETTIMANALE
MEZZO_PROPRIETARIO	E	VOLUME	1--3	GIORNALIERO
NCC	E	VOLUME	1--3	GIORNALIERO
DIPENDENTE	E	VOLUME	0	SETTIMANALE
TIPO_CONTRATTO_D	E	VOLUME	0	SETTIMANALE
TURNO	E	VOLUME	4	MENSILE
RUOLO	E	VOLUME	8	MENSILE
ORARIO_TURNI	A	VOLUME	8	MENSILE
FORNITORE	E	VOLUME	0	SETTIMANALE
TIPO_CONTRATTO_F	E	VOLUME	0	SETTIMANALE
SEDI	E	VOLUME	0	SETTIMANALE
PRENOTAZIONE	E	VOLUME	21-25	SETTIMANALE
CAMERA	E	VOLUME	21	SETTIMANALE
ATTIVITA	E	VOLUME	21	SETTIMANALE
PROMOZIONE	E	VOLUME	2	SETTIMANALE
PAGAMENTO	E	VOLUME	21-25	SETTIMANALE
DETtaglio_PAGAMENTO	E	VOLUME	21-25	SETTIMANALE

**legenda:** E = Entità A = Associazione

# 1.8 VINCOLI DI INTEGRITÀ

I vincoli di integrità in SQL sono importanti per assicurare la correttezza e la consistenza dei dati all'interno di un database. Essi definiscono le regole per l'eliminazione dei dati.

Abbiamo 2 tipi di vincoli:

## O1 STATICI

- Codice fiscale per ogni persona è unico
- Gli ospiti possono esibire come documento di riconoscimento solo : 'CARTA\_DI\_IDENTITA', 'PATENTE', 'PASSAPORTO'
- Il tipo di veicolo con cui possono venire gli ospiti deve essere uno dei seguenti: 'AUTOVEICOLO', 'MOTOVEICOLO'.
- I dipendenti possono avere solo uno dei seguenti ruoli : 'BARISTA', 'ADDETTO\_PULIZIE', 'RECEPTIONIST', 'ADDETTO\_CUCINA'
- il nome del turno può essere uno tra : 'MATTINA', 'POMERIGGIO', 'SERA', 'NOTTURNO'

## O 2 DINAMICI

- Un dipendente non può avere più di 2 ruoli
- Ogni dipendente deve fare un solo turno giornaliero
- Ogni ospite può prenotare fino a 2 ATTIVITA al giorno
- Una persona non può avere meno di 18 anni
- Un fornitore non può più di 2 sedi
- Per ogni prenotazione deve corrispondere una sola promozione

# 1.8 VINCOLI DI INTEGRITÀ

## PT2

I vincoli di integrità in SQL sono importanti per assicurare la correttezza e la consistenza dei dati all'interno di un database. Essi definiscono le regole per l'eliminazione dei dati.

Abbiamo 2 tipi di vincoli:

### O1 STATICI

- Le attività proposte dalla struttura possono essere solo: 'ESCURSIONE','PISCINA','SAUNA'
- La promozione può avere uno sconto pari solo a uno dei seguenti:'%15','%25'
- L'ospite può pagare con uno dei seguenti metodi di pagamento : 'CARTA','CONTANTI','BONIFICO'
- 
- 
- 

### O 2 DINAMICI

- Verifica sul minimo salariole dei dipendenti
- Verifica del massimo importo trimestrale ai fornitori
- Verifica validità dei contratti
- 
- 
-

# 1.9 VERIFICA DI NORMALITA

Le forme normali servono a garantire che lo schema abbia un livello di ridondanza minima. I 4 livelli di normalizzazione più importanti sono:

1. Ogni attributo è atomico
2. Non ci sono dipendenze parziali da chiavi
3. Gli attributi non-chiave dipendono unicamente dalla chiave
4. BCNF: Non ammette ridondanze

## PRIMA FORMA NORMALE

Per la prima forma normale, dobbiamo fare una considerazione: essendoci numerosi attributi date, la prima forma normale non è rispettata, però per molti linguaggi di programmazione, il tipo date è primitivo. Quindi, possiamo dire che tutti i campi dello schema sono atomici, di conseguenza la prima forma normale è rispettata.

## SECONDA FORMA NORMALE

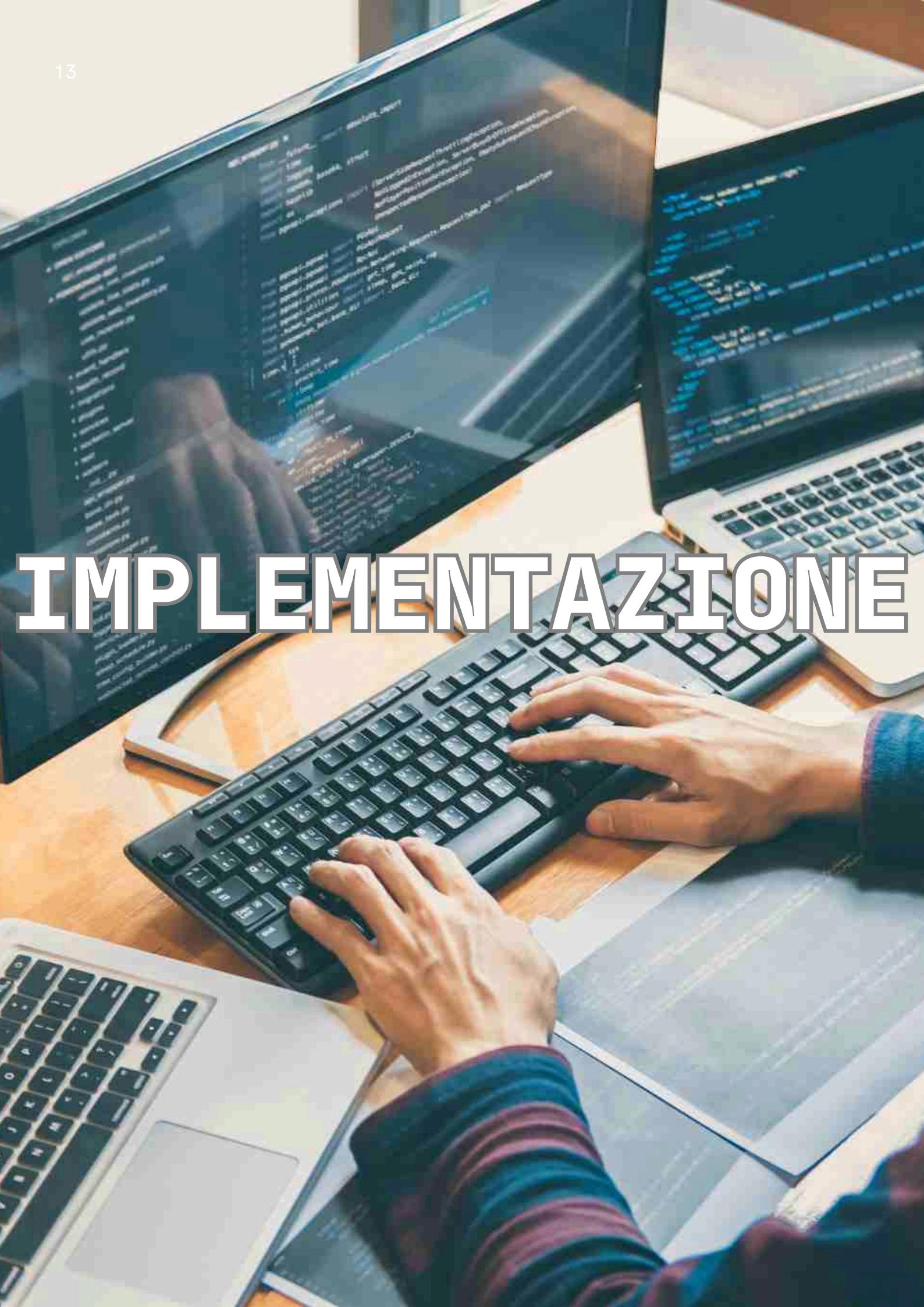
Abbiamo due chiavi multi-attributo in Attivita e Promozione. Per entrambe le relazioni, le chiavi non presentano alcuna dipendenza parziale con altri attributi.

## TERZA FORMA NORMALE

Lo schema non presenta dipendenze anomale.

## BCNF

Essendo che lo schema non presenta dipendenze anomale, esso è in BCNF.



# IMPLEMENTAZIONE

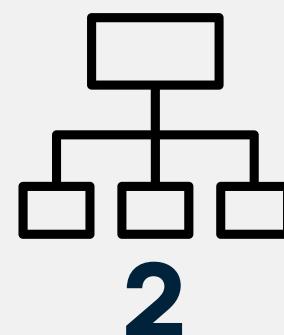
# Implementazione del Progetto con PL/SQL

Creazione, Gestione e Ottimizzazione del Database Oracle"

L'implementazione ha coinvolto la traduzione delle specifiche progettuali in codice PL/SQL eseguibile all'interno del DBMS Oracle. Questo ha incluso la creazione di tabelle per memorizzare i dati, la definizione di utenti con vari livelli di accesso, e la scrittura di procedure e trigger per gestire le operazioni del database.



ideare



Progettare



Programmare pl/sql



Base dati guitart

ci troviamo in questa fase!

## 2.1.a CREAZIONE DEGLI UTENTI:

### ADMIN

QUESTO BLOCCO DI CODICE CREA UN NUOVO UTENTE NEL DATABASE CHIAMATO ADMIN CON LA PASSWORD ADM.

SUCCESSIVAMENTE, VENGONO CONCESSI TUTTI I PRIVILEGI A QUESTO UTENTE, IL CHE SIGNIFICA CHE ADMIN PUÒ ESEGUIRE QUALSIASI OPERAZIONE SUL DATABASE, INCLUSI SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT, REVOKE E COSÌ VIA.

```
//ADMIN CREATE USER ADMIN IDENTIFIED BY  
ADM; --PERMESSI DEFINITI 2.1.b
```

### GESTORE

QUESTO BLOCCO DI CODICE CREA UN NUOVO UTENTE NEL DATABASE

CHIAMATO GESTORE CON LA PASSWORD GESTISCE.

SUCCESSIVAMENTE, GLI VENGONO CONCESSI VARI PRIVILEGI.

```
//GESTORE CREATE USER GESTORE IDENTIFIED  
BY GESTISCE; --PERMESSI DEFINITI 2.1.b
```

### OSPITE

QUESTO BLOCCO DI CODICE CREA UN NUOVO UTENTE NEL DATABASE

CHIAMATO OSPITE CON LA PASSWORD NOSPITE.

SUCCESSIVAMENTE, GLI VENGONO CONCESSI VARI PRIVILEGI SUCCESSIVAMENTE RIPORTATI.

```
//OSPITE CREATE USER OSPITE  
IDENTIFIED BY NOSPITE; --PRIVILEGI  
RIPORTATI IN 2.1.b
```

## 2.1.b PERMESSI DATI A VARI UTENTI (DATA CONTROL LANGUAGE).

La gestione dei permessi e l'assegnazione di ruoli sono aspetti essenziali per garantire la sicurezza e l'integrità dei dati. Questo processo viene gestito tramite il Data Control Language (DCL), che consente di configurare i permessi per vari utenti.

Nel nostro sistema, abbiamo definito tre ruoli principali: RESPONSABILE\_ADMIN e DIPENDENTE\_USER e CLIENTE\_USER. Il primo è dotato dei più alti privilegi di amministrazione, mentre il secondo e il terzo ha permessi limitati, adeguati alle sue funzioni specifiche. Questo ci consente di mantenere il controllo sui dati e garantire che solo gli utenti autorizzati abbiano accesso alle informazioni pertinenti.

Seguono i dettagli di ciascun ruolo e i comandi DCL utilizzati per assegnare i permessi appropriati.

### PERMESSI ADMIN

ADMIN: Questo utente ha i privilegi di amministratore più elevati nel sistema.

Il comando GRANT ALL PRIVILEGES gli concede il diritto di eseguire qualsiasi operazione sul database.

```
GRANT ALL PRIVILEGES to ADMIN;  
GRANT CONNECT, CREATE SESSION TO ADMIN;
```

# PERMESSI GESTORE

GESTORE: Questo utente ha privilegi limitati che gli permettono di eseguire solo determinate operazioni.

GRANT CREATE SESSION gli permette di avviare una sessione nel database.

In base ai permessi ottenuti così puo' modificare o eventualmente eliminare i dati nelle tabelle.

```
GRANT CONNECT, CREATE SESSION TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON TURNO TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON RUOLO TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON COSTITUITO TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON MEZZO_PROPRIETARIO TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON SEDI TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON NCC TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON CAMERA TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON ATTIVITA TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON PROMOZIONE TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON PRENOTAZIONE TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON PAGAMENTO TO GESTORE;
```

```
GRANT SELECT, DELETE, UPDATE ON DETTAGLIO_PAGAMENTO TO GESTORE;
```

```
GRANT EXECUTE ON CercaDipendentiPerMansione TO GESTORE;
```

```
GRANT EXECUTE ON CercaTurniPerDipendente TO GESTORE;
```

```
GRANT EXECUTE ON CercaFornitoriPerTipologia TO GESTORE;
```

```
GRANT EXECUTE ON CercaPrenotazioniPerZonaPrelievo TO GESTORE;
```

# PERMESSI OSPITE

OSPISTE: Questo utente ha privilegi molto limitati, il che è appropriato per un utente esterno che interagisce con il sistema.

GRANT CONNECT gli permette di stabilire una connessione con il database.

Ha anche il permesso di eseguire la procedura memorizzata CercaPrenotazioniOspite, che presumibilmente gli permette di vedere le proprie prenotazioni.

```
GRANT CONNECT TO OSPITE;  
GRANT SELECT, DELETE, UPDATE ON PRENOTAZIONE TO  
OSPISTE;  
GRANT SELECT, DELETE, UPDATE ON ATTIVITA TO  
OSPISTE;  
GRANT EXECUTE ON CercaprenotazioniOspite TO  
OSPISTE;
```

## 2.2 DATA DEFINITION LANGUAGE:

Il DDL è utilizzato per definire, modificare e gestire la struttura dei dati e gli oggetti del database. Qui vengono riportate la creazione delle tabelle che permettono al sistema di gestione del B&B, gestendo le informazioni relative agli ospiti, ai dipendenti, ai fornitori e quindi le forniture.

**PERSONA:** La tabella "PERSONA" memorizza le informazioni relative a tutte le persone, ovvero sia fornitori, ospiti che dipendenti; tutti i campi sono obbligatori e il tel è unico

```
CREATE TABLE PERSONA (
    CF CHAR(16) PRIMARY KEY,
    NOME VARCHAR(25) NOT NULL,
    COGNOME VARCHAR(25) NOT NULL,
    DATA_NASCITA DATE NOT NULL,
    TEL CHAR(10) UNIQUE
);
```

**OSPITE:** La tabella "OSPITE" memorizza le informazioni relative a tutti gli ospiti.

```
CREATE TABLE OSPITE (
    TIPO_DOCUMENTO VARCHAR(20),
    EMAIL VARCHAR(40),
    CF OSPITE CHAR(16) PRIMARY KEY,
    CONSTRAINT FK OSPITE FOREIGN KEY (CF OSPITE) REFERENCES PERSONA(CF),
    CONSTRAINT TIPO_DOCUMENTO_CHECK CHECK (TIPO_DOCUMENTO IN
        ('CARTA DI IDENTITA','PATENTE','PASSAPORTO'))
);
```

**MEZZO\_PROPRIOPRIO:** La tabella "MEZZO\_PROPRIOPRIO" memorizza le informazioni relative a tutti gli ospiti che vengono col mezzo proprio identificandoli con la targa del veicolo

```
CREATE TABLE MEZZO_PROPRIOPRIO(
    TARGA VARCHAR(7) PRIMARY KEY,
    TIPO_M VARCHAR(11),
    CF_PROPRIETARIO CHAR(16),
    CONSTRAINT TIPO_M_CHECK CHECK (TIPO_M IN ('MOTOVEICOLO','AUTOVEICOLO')),
    CONSTRAINT FK_mez FOREIGN KEY (CF_PROPRIETARIO) REFERENCES OSPITE(CF_OSPITE)
);
```

**NCC:** La tabella "NCC" memorizza le informazioni relative a tutti gli ospiti che vengono col un NCC

```
CREATE TABLE NCC(
    N_PRENOTAZIONE_NCC NUMBER(3,0) PRIMARY KEY,
    ZONA_PRELIEVO_OSP VARCHAR(35) NOT NULL,
    CF_RICHIEDENTE CHAR(16),
    CONSTRAINT FK_NCC FOREIGN KEY (CF_RICHIEDENTE) REFERENCES OSPITE(CF_OSPITE)
);
```

**RUOLO:** La tabella "RUOLO" ha lo scopo di identificare i vari ruoli che possono essere riscoperti dai dipendenti

```
CREATE TABLE RUOLO(
    CODICE_BADGE CHAR(10) PRIMARY KEY,
    MANSIONE VARCHAR(20),
    CONSTRAINT MANSIONE_CHECK CHECK (MANSIONE IN
    ('BARISTA','ADDETTO_PULIZIE','RECEPTIONIST','ADDETTO_CUCINA'))
);
```

**DIPENDENTE:** La tabella "DIPENDENTE" memorizza le informazioni relative a tutti gli dipendenti.

```
CREATE TABLE DIPENDENTE(
    CF_DIPENDENTE CHAR(16) PRIMARY KEY,
    LINGUE_PARLATE VARCHAR(8),
    ANNI_ESPERIENZA NUMBER(2,0) NOT NULL,
    CODICE_BADGE_D CHAR(10),
    CONSTRAINT FK_DIP FOREIGN KEY (CF_DIPENDENTE) REFERENCES PERSONA(CF),
    CONSTRAINT FK_RL FOREIGN KEY (CODICE_BADGE_D) REFERENCES RUOLO(CODICE_BADGE),
    CONSTRAINT LINGUE_PARLATE_CHECK CHECK(LINGUE_PARLATE IN
        ('SPAGNOLO','INGLESE','FRANCESE'))
);
```

**TURNO:** La tabella "TURNO" memorizza le informazioni relative a tutte le tipologie di turno esistenti.

```
CREATE TABLE TURNO(
    NOME_TURNO VARCHAR(10) PRIMARY KEY,
    INIZIO_TURNO DATE NOT NULL,
    FINE_TURNO DATE NOT NULL,
    CONSTRAINT NOME_TURNO_CHECK CHECK (NOME_TURNO IN
        ('MATTINA','POMERIGGIO','SERÀ','NOTTURNO'))
);
```

**ORARIO\_TURNI:** La tabella "ORARIO\_TURNI" serve da tramite per combinare i dipendenti con i turni a loro assegnati

```
CREATE TABLE ORARIO_TURNI(
    CF_DIPENDENTE CHAR(16),
    NOME_TURNO VARCHAR(10),
    DATA_INIZIO_TURNI DATE,
    DATA_FINE_TURNI DATE,
    FOREIGN KEY (CF_DIPENDENTE) REFERENCES DIPENDENTE(CF_DIPENDENTE),
    FOREIGN KEY (NOME_TURNO) REFERENCES TURNO(NOME_TURNO)
);
```

**TIPO\_CONTRATTO\_D:** La tabella "TIPO\_CONTRATTO\_D" serve a descrivere i tipi di contratto dei dipendenti

```
CREATE TABLE TIPO_CONTRATTO_D(
    CODICE_IDENTIFICATIVO_D CHAR(11) PRIMARY KEY,
    CF_DIPENDENTE CHAR(16),
    DATA_INIZIO_CONTRATTO DATE NOT NULL,
    DATA_SCADENZA DATE,
    STIPENDIO NUMBER(5,0) NOT NULL,
    CONSTRAINT FK_CT FOREIGN KEY (CF_DIPENDENTE) REFERENCES
        DIPENDENTE(CF_DIPENDENTE)
);
```

**FORNITORE:** La tabella "FORNITORE" memorizza le informazioni relative a tutti i fornitori.

```
CREATE TABLE FORNITORE(
    CF_FORNITORE CHAR(16) PRIMARY KEY,
    P_IVA CHAR(10) NOT NULL,
    TIPOLOGIA VARCHAR(15)
);
```

**TIPO\_CONTRATTO\_F:** La tabella "TIPO\_CONTRATTO\_F" serve a descrivere i tipi di contratto dei fornitori

```
CREATE TABLE TIPO_CONTRATTO_F(
    CODICE_IDENTIFICATIVO_F CHAR(11) PRIMARY KEY,
    CF_FORNITORE CHAR(16),
    IMPORTO_TRIMESTRALE NUMBER(5,0) NOT NULL,
    START_OF_CONTRACT DATE,
    END_OF_CONTRACT DATE,
    CONSTRAINT FK_CF FOREIGN KEY (CF_FORNITORE) REFERENCES
        FORNITORE(CF_FORNITORE)
);
```

**SEDI:** La tabella "SEDI" serve a descrivere le sedi dei fornitori

```
CREATE TABLE SEDI(
    LOCALITA VARCHAR(25) PRIMARY KEY,
    FORNITORE_CF CHAR(16),
    GIORNI_CHIUSURA VARCHAR(15),
    EMAIL_SEDE VARCHAR(45),
    CONSTRAINT FK_SD FOREIGN KEY (FORNITORE_CF) REFERENCES
        FORNITORE(CF_FORNITORE)
);
```

**PRENOTAZIONE:** La tabella "PRENOTAZIONE" serve a memorizzare le informazioni sulle prenotazioni

```
CREATE TABLE PRENOTAZIONE(
    NUMERO_PRENOTAZIONE NUMBER(10,0) PRIMARY KEY,
    DATA_PRENOTAZIONE DATE NOT NULL,
    CF OSPITE_PRENOTANTE CHAR(16),
    CONSTRAINT FK_P FOREIGN KEY (CF OSPITE_PRENOTANTE) REFERENCES
        OSPITE(CF OSPITE)
);
```

**CAMERA:** La tabella "CAMERA" serve a memorizzare le informazioni sulle camere

```
CREATE TABLE CAMERA(
    NUMERO_CAMERA NUMBER(1,0) PRIMARY KEY,
    PREZZO_CAMERA NUMBER (3,0) NOT NULL,
    DISPONIBILITA CHAR(1) NOT NULL,
    NUMERO_PRENOTAZIONE NUMBER(10,0),
    CONSTRAINT DISPONIBILITA_CHECK CHECK (DISPONIBILITA IN ('T','F')),
    CONSTRAINT FK_CM FOREIGN KEY (NUMERO_PRENOTAZIONE) REFERENCES
        PRENOTAZIONE(NUMERO_PRENOTAZIONE)
);
```

**ATTIVITA:** La tabella "ATTIVITA" serve a memorizzare le informazioni sulle attivita svolte

```
CREATE TABLE ATTIVITA(
    NOME_ATTIVITA VARCHAR(10) NOT NULL,
    PREZZO_ATTIVITA NUMBER (3,0) NOT NULL,
    NUM_PREN NUMBER(10,0),
    DATA_ATTIVITA DATE,
    CONSTRAINT PK_A PRIMARY KEY(NOME_ATTIVITA,DATA_ATTIVITA),
    CONSTRAINT NOME_ATTIVITA_CHECK CHECK (NOME_ATTIVITA IN
        ('ESCURSIONE','PISCINA','SAUNA')),
    CONSTRAINT FK_AT FOREIGN KEY (NUM_PREN) REFERENCES
        PRENOTAZIONE(NUMERO_PRENOTAZIONE)
);
```

**PROMOZIONE:** La tabella "PROMOZIONE" serve a memorizzare le informazioni sulle promozioni attive

```
CREATE TABLE PROMOZIONE(
    NOME_PROMOZIONE VARCHAR(10) NOT NULL,
    DATA_INIZIO DATE NOT NULL,
    DATA_FINE DATE NOT NULL,
    PERC_SCONTI VARCHAR(3) NOT NULL,
    COD_PROMOZIONE NUMBER(10,0) NOT NULL,
    CONSTRAINT PK_P PRIMARY KEY (NOME_PROMOZIONE,DATA_INIZIO),
    CONSTRAINT PERC_SCONTI_CHECK CHECK (PERC_SCONTI IN ('%15','%25')),
    CONSTRAINT FK_PR FOREIGN KEY (COD_PROMOZIONE) REFERENCES
        PRENOTAZIONE(NUMERO_PRENOTAZIONE)
);
```

**PAGAMENTO:** La tabella "PAGAMENTO" riprende le informazioni generiche sul pagamento

```
CREATE TABLE PAGAMENTO(
    COD_TRANSAZIONE VARCHAR(25) PRIMARY KEY,
    TIPO_PAGAMENTO VARCHAR(20) NOT NULL,
    DATA_TRANSAZIONE DATE NOT NULL,
    IMPORTO NUMBER(3,0) NOT NULL,
    INTESTATARIO VARCHAR(25) NOT NULL,
    NUM_PRENOT NUMBER(10,0),
    CONSTRAINT TIPO_PAGAMENTO_CHECK CHECK (TIPO_PAGAMENTO IN
        ('CARTA','CONTANTI','BONIFICO')),
    CONSTRAINT FK_PA FOREIGN KEY (NUM_PRENOT) REFERENCES
        PRENOTAZIONE(NUMERO_PRENOTAZIONE)
);
```

**DETTOGLIO\_PAGAMENTO:** La tabella "DETTOGLIO\_PAGAMENTO" riprende dettagli particolari sul pagamento

```
CREATE TABLE DETTOGLIO_PAGAMENTO(  
    COD_TRANSAZIONE VARCHAR(25) PRIMARY KEY,  
    VALUTA_EUROPEA CHAR(1) NOT NULL,  
    RATEIZZAZIONE CHAR(1),  
    CONSTRAINT RATERIZZAZIONE_CHECK CHECK (RATERIZZAZIONE IN ('T','F')),  
    CONSTRAINT VALUTA_EUROPEA_CHECK CHECK (VALUTA_EUROPEA IN ('T','F')),  
    CONSTRAINT FK_PP FOREIGN KEY (COD_TRANSAZIONE) REFERENCES  
        PAGAMENTO(COD_TRANSAZIONE)  
);
```

# 2.2 DATA MANIPULATION LANGUAGE:

Il Data Manipulation Language (DML) è uno strumento fondamentale per gestire i dati in un database SQL, consentendo di inserire, modificare, eliminare e recuperare dati. Inoltre, il DML supporta l'applicazione di vincoli di integrità, che garantiscono l'accuratezza e l'affidabilità dei dati. Nel popolare il nostro database, abbiamo seguito un ordine preciso: prima abbiamo popolato le tabelle che non dipendono da altre. Questo approccio ha semplificato il processo e ha evitato potenziali conflitti di dipendenza.

## PERSONA

```
INSERT INTO PERSONA VALUES('AAAABBBBCCCC12DD','Mario','Rossi','11-DEC-90','3303303314');
INSERT INTO PERSONA VALUES('AAAABBBBCCCC12EE','Marco','Verdi','12-DEC-95','3303302214');
INSERT INTO PERSONA VALUES('HHHHBBBBCCCC12EE','Marco','Bianchi','12-DEC-95','3303301214');
INSERT INTO PERSONA VALUES('CCDCBBBBCCCC12EE','Luigi','Bianchi','12-NOV-96','3303301213');

INSERT INTO PERSONA VALUES('FFFFDDDD1616GGGG','Luigi','Rossi','11-JUL-2000','3303305544');
INSERT INTO PERSONA VALUES('FFFFDDDD1616DDCC','Tommaso','Verdi','14-DEC-97','3303302115');
INSERT INTO PERSONA VALUES('FFFFDDDD1616DCCE','John','Rossi','14-DEC-97','3303301315');
INSERT INTO PERSONA VALUES('FFFFDDDD161610FF','Ginevra','Esposito','12-DEC-94','3303301777');
INSERT INTO PERSONA VALUES('FFFFDDDD1616IIII','Giovanni','Rossi','22-JUL-2001','3303305336');
INSERT INTO PERSONA VALUES('FFFFDDDD1616DDII','Taddeo','Bianchi','12-DEC-95','3303302225');
INSERT INTO PERSONA VALUES('FFFFDDDD1616DICE','Matteo','Esposito','12-NOV-97','3304301315');
INSERT INTO PERSONA VALUES('FFFFDDDY161610FF','Emanuele','Esposito','18-OCT-93','3313301777');
INSERT INTO PERSONA VALUES('B10HBBBBCCCC13ZZ','Ciro','De santis','18-SEP-78','7763301214');
INSERT INTO PERSONA VALUES('A101BBBBCCCC12DD','Salvatore','Coppola','14-DEC-80','3301244314');
INSERT INTO PERSONA VALUES('B10HBBBBCCCC12EE','Gennaro','Esposito','11-JUL-75','3301200214');
INSERT INTO PERSONA VALUES('EEEEBBBBCCCC12DD','Giovanni','Iodice','11-JUL-70','3303305543');
```

**OSPITE**

```

INSERT INTO OSPITE
VALUES('PATENTE','mariorossi@libero.it','AAAABBBBCCCC12DD');
INSERT INTO OSPITE
VALUES('PASSAPORTO','luigibianchi@libero.it','CCDCBBBBCCCC12EE');
INSERT INTO OSPITE
VALUES('CARTA_DI_IDENTITA','marcobianchi@libero.it','HHHHBBBBCCCC12EE');
INSERT INTO OSPITE
VALUES('PATENTE','marcoverdi@libero.it','AAAABBBBCCCC12EE');

```

**MEZZO\_PROPRI**

```

INSERT INTO MEZZO_PROPRI
VALUES('GA700WA','MOTOVEICOLO','AAAABBBBCCCC12DD');
INSERT INTO MEZZO_PROPRI
VALUES('BA544QA','AUTOVEICOLO','HHHHBBBBCCCC12EE');
INSERT INTO MEZZO_PROPRI
VALUES('FC500DA','AUTOVEICOLO','AAAABBBBCCCC12EE');

```

**NCC**

```
INSERT INTO NCC VALUES(100,'PIAZZA GARIBALDI','CCDCBBBBCCCC12EE');
```

**RUOLO**

```

INSERT INTO RUOLO VALUES('ACC1020EE7','BARISTA');
INSERT INTO RUOLO VALUES('ABB507E105','RECEPTIONIST');
INSERT INTO RUOLO VALUES('ABB507EBBC','ADDETTO_PULIZIE');
INSERT INTO RUOLO VALUES('DDE507EBBC','ADDETTO_CUCINA');

INSERT INTO RUOLO VALUES('ACC1020EE8','BARISTA');
INSERT INTO RUOLO VALUES('ABB507E106','RECEPTIONIST');
INSERT INTO RUOLO VALUES('ABB507EBBD','ADDETTO_PULIZIE');
INSERT INTO RUOLO VALUES('DDE507EBBD','ADDETTO_CUCINA');

```

**DIPENDENTE**

```

INSERT INTO DIPENDENTE
VALUES('FFFFDDDD1616GGGG', 'SPAGNOLO', 10, 'ACC1020EE7');
INSERT INTO DIPENDENTE
VALUES('FFFFDDDD1616DDCC', 'SPAGNOLO', 10, 'ABB507E105');
INSERT INTO DIPENDENTE
VALUES('FFFFDDDD1616DCCE', 'INGLESE', 8, 'ABB507EBBC');
INSERT INTO DIPENDENTE
VALUES('FFFFDDDD161610FF', 'FRANCESE', 7, 'DDE507EBBC');

INSERT INTO DIPENDENTE
VALUES('FFFFDDDD1616IIII', 'INGLESE', 12, 'ACC1020EE8');
INSERT INTO DIPENDENTE
VALUES('FFFFDDDD1616DDII', 'FRANCESE', 11, 'ABB507E106');
INSERT INTO DIPENDENTE
VALUES('FFFFDDDD1616DICE', 'INGLESE', 6, 'ABB507EBBD');
INSERT INTO DIPENDENTE
VALUES('FFFFDDDY161610FF', 'INGLESE', 8, 'DDE507EBBD');

```

**TIPO\_CONTRATTO\_D**

```

INSERT INTO TIPO_CONTRATTO_D VALUES('320AABCDD90', 'FFFFDDDD1616GGGG', '12-
OCT-2012', '11-DEC-2024', 20000);
INSERT INTO TIPO_CONTRATTO_D VALUES('450AABEEE70', 'FFFFDDDD1616DDCC', '12-
FEB-2012', '11-DEC-2025', 18000);
INSERT INTO TIPO_CONTRATTO_D VALUES('555CCRREE75', 'FFFFDDDD1616DCCE', '14-
OCT-2014', NULL, 25000);
INSERT INTO TIPO_CONTRATTO_D VALUES('777WWWET80', 'FFFFDDDD161610FF', '15-
JUL-2015', '08-NOV-2025', 24000);
INSERT INTO TIPO_CONTRATTO_D VALUES('663EEQQAT79', 'FFFFDDDD1616IIII', '12-
NOV-2015', '11-DEC-2024', 20000);
INSERT INTO TIPO_CONTRATTO_D VALUES('660RRYYRR81', 'FFFFDDDD1616DDII', '12-
OCT-2012', '08-NOV-2025', 18000);
INSERT INTO TIPO_CONTRATTO_D VALUES('300JJZZRK87', 'FFFFDDDD1616DICE', '11-
JUL-2013', NULL, 25000);
INSERT INTO TIPO_CONTRATTO_D VALUES('960JKJNLM90', 'FFFFDDDY161610FF', '25-
OCT-2014', '08-NOV-2025', 24000);

```

**TURNO**

```

INSERT INTO TURNO
VALUES('MATTINA', TO_DATE('06:30:00', 'HH24:MI:SS'), TO_DATE('12:30:00', 'HH24
:MI:SS'));
INSERT INTO TURNO
VALUES('POMERIGGIO', TO_DATE('12:30:00', 'HH24:MI:SS'), TO_DATE('18:30:00', 'H
H24:MI:SS'));
INSERT INTO TURNO
VALUES('SERA', TO_DATE('18:30:00', 'HH24:MI:SS'), TO_DATE('00:30:00', 'HH24:MI
:SS'));
INSERT INTO TURNO
VALUES('NOTTURNO', TO_DATE('00:30:00', 'HH24:MI:SS'), TO_DATE('06:30:00', 'HH2
4:MI:SS'));

```

**ORARIO\_TURNI**

```

INSERT INTO ORARIO_TURNI VALUES('FFFFDDDD1616GGGG', 'MATTINA', '15-MAY-2024', '25-JUN-2024');
INSERT INTO ORARIO_TURNI VALUES('FFFFDDDD1616DDCC', 'POMERIGGIO', '15-MAY-2024', '25-JUN-2024');
INSERT INTO ORARIO_TURNI VALUES('FFFFDDDD1616DCCE', 'SERA', '15-MAY-2024', '25-JUN-2024');
INSERT INTO ORARIO_TURNI VALUES('FFFFDDDD161610FF', 'NOTTURNO', '15-MAY-2024', '25-JUN-2024');
INSERT INTO ORARIO_TURNI VALUES('FFFFDDDD1616IIII', 'MATTINA', '15-MAY-2024', '25-JUN-2024');
INSERT INTO ORARIO_TURNI VALUES('FFFFDDDD1616DDII', 'POMERIGGIO', '15-MAY-2024', '25-JUN-2024');
INSERT INTO ORARIO_TURNI VALUES('FFFFDDDD1616DICE', 'SERA', '15-MAY-2024', '25-JUN-2024');
INSERT INTO ORARIO_TURNI VALUES('FFFFDDDY161610FF', 'NOTTURNO', '15-MAY-2024', '25-JUN-2024');

```

**FORNITORE**

```

INSERT INTO FORNITORE VALUES('B10HBBBBCCCC13ZZ', 'B2024AA91', 'Bar Services');
INSERT INTO FORNITORE VALUES('A101BBBBCCCC12DD', 'B66BB47A91', 'Laundry Service');
INSERT INTO FORNITORE VALUES('B10HBBBBCCCC12EE', 'B66BC98A91', 'Bibite');
INSERT INTO FORNITORE
VALUES('EEEEBBBBCCCC12DD', 'B66BB57B91', 'Alimentare');

```

**TIPO\_CONTRATTO\_F**

```

INSERT INTO TIPO_CONTRATTO_F VALUES
('MNN541RRD65', 'B10HBBBBCCCC13ZZ', 6000, '01-JAN-24', '31-DEC-25');
INSERT INTO TIPO_CONTRATTO_F VALUES
('KRR698DFR44', 'A101BBBBCCCC12DD', 6500, '01-JAN-23', '31-DEC-24');
INSERT INTO TIPO_CONTRATTO_F VALUES
('GGD155JJN98', 'B10HBBBBCCCC12EE', 7000, '01-JAN-23', '31-DEC-26');
INSERT INTO TIPO_CONTRATTO_F VALUES
('ZZR514FFE81', 'EEEEBBBBCCCC12DD', 6000, '01-FEB-24', '31-DEC-26');

```

**TURNO**

```

INSERT INTO TURNO
VALUES('MATTINA', TO_DATE('06:30:00', 'HH24:MI:SS'), TO_DATE('12:30:00', 'HH24:MI:SS'));
INSERT INTO TURNO
VALUES('POMERIGGIO', TO_DATE('12:30:00', 'HH24:MI:SS'), TO_DATE('18:30:00', 'HH24:MI:SS'));
INSERT INTO TURNO
VALUES('SERA', TO_DATE('18:30:00', 'HH24:MI:SS'), TO_DATE('00:30:00', 'HH24:MI:SS'));
INSERT INTO TURNO
VALUES('NOTTURNO', TO_DATE('00:30:00', 'HH24:MI:SS'), TO_DATE('06:30:00', 'HH24:MI:SS'));

```

**INSERIMENTO SEDI**

```

INSERT INTO SEDI VALUES
('Ercolano','B10HBBBBCCCC12EE','LUNEDI','ercolano_services@gmail.com');
INSERT INTO SEDI VALUES ('San Giuseppe
Vesuviano','B10HBBBBCCCC12EE','VENERDI','sede_SGV@gmail.com');
INSERT INTO SEDI VALUES
('Giugliano','A101BBBBCCCC12DD','MARTEDI','giugliano_services@gmail.com');
INSERT INTO SEDI VALUES
('Pompei','A101BBBBCCCC12DD','DOMENICA','pompei_services@gmail.com');
INSERT INTO SEDI VALUES
('Villaricca','B10HBBBBCCCC13ZZ','LUNEDI','villaricca_bibite@libero.it');
INSERT INTO SEDI VALUES
('Casoria','B10HBBBBCCCC13ZZ','MERCOLEDI','Casoria_bibite@libero.it');
INSERT INTO SEDI VALUES
('Napoli','EEEEBBBBCCCC12DD','MARTEDI','giugliano_alimentari@libero.it');
INSERT INTO SEDI VALUES
('Sorrento','EEEEBBBBCCCC12DD','SABATO','Sorrento_alimenti@libero.it');

```

**INSERIMENTO PRENOTAZIONI**

```

INSERT INTO PRENOTAZIONE VALUES(5000,'20-MAY-2024','AAAABBBBCCCC12DD');
INSERT INTO PRENOTAZIONE VALUES(4000,'20-MAY-2024','CCDCBBBBCCCC12EE');
INSERT INTO PRENOTAZIONE VALUES(6666,'20-FEB-2024','HHHHBBBBCCCC12EE');
INSERT INTO PRENOTAZIONE VALUES(10000,'20-APR-2024','AAAABBBBCCCC12EE');

```

**INSERIMENTO CAMERA**

```

INSERT INTO CAMERA VALUES(1,150,'T',4000);
INSERT INTO CAMERA VALUES(2,175,'T',5000);
INSERT INTO CAMERA VALUES(3,100,'T',10000);

```

**INSERIMENTO ATTIVITA'**

```

INSERT INTO ATTIVITA VALUES('ESCURSIONE',100,6666,'6-JUN-2024');
INSERT INTO ATTIVITA VALUES('PISCINA',40,4000,'14-JUL-2024');
INSERT INTO ATTIVITA VALUES('SAUNA',75,5000,'13-JUN-2024');
INSERT INTO ATTIVITA VALUES('SAUNA',75,4000,'14-JUL-2024');
INSERT INTO ATTIVITA VALUES('PISCINA',40,5000,'13-JUN-2024');

```

**INSERIMENTO PROMOZIONE**

```

INSERT INTO PROMOZIONE VALUES('MANAGER','1-JUN-2024','31-JUL-
2024','%15',10000);
INSERT INTO PROMOZIONE VALUES('ERASMUS','1-JUN-2024','31-JUL-
2024','%25',4000);

```

**INserimento PAGAMENTO**

```
INSERT INTO PAGAMENTO VALUES('8792 4536 2187 7654', 'CARTA', '15-JUN-2024', 255, 'B&B BORBONICO', 10000);
INSERT INTO PAGAMENTO VALUES('5634 9812 7456 3201', 'CONTANTI', '15-JUL-2024', 726, 'B&B BORBONICO', 4000);
INSERT INTO PAGAMENTO VALUES('1789 6543 2901 4367', 'BONIFICO', '15-JUN-2024', 600, 'B&B BORBONICO', 5000);
INSERT INTO PAGAMENTO VALUES('9012 3456 7689 1234', 'CONTANTI', '7-JUN-2024', 100, 'B&B BORBONICO', 6666);
```

**INserimento DETTAGLIO\_PAGAMENTO**

```
INSERT INTO DETTAGLIO_PAGAMENTO VALUES('8792 4536 2187 7654', 'F', 'T');
INSERT INTO DETTAGLIO_PAGAMENTO VALUES('5634 9812 7456 3201', 'F', 'F');
INSERT INTO DETTAGLIO_PAGAMENTO VALUES('1789 6543 2901 4367', 'F', 'T');
INSERT INTO DETTAGLIO_PAGAMENTO VALUES('9012 3456 7689 1234', 'F', 'T');
```

**INserimento PROMOZIONE**

```
INSERT INTO PROMOZIONE VALUES('MANAGER', '1-JUN-2024', '31-JUL-2024', '%15', 10000);
INSERT INTO PROMOZIONE VALUES('ERASMUS', '1-JUN-2024', '31-JUL-2024', '%25', 4000);
```

## 2.4 TRIGGER

Un trigger viene eseguito automaticamente in risposta a determinati eventi o azioni che si verificano sulla tabella a cui è associato. Lo scopo del trigger è quello di controllare i vincoli dinamici all'inserimento, cancellazione o modifica dei dati.

### **NUMERO MASSIMO DI RUOLI**

Questo trigger controlla che ogni volta che si aggiunge un dipendente, questo non superi il limite massimo di ruoli

```
CREATE OR REPLACE TRIGGER TROPPI_RUOLI
BEFORE INSERT ON DIPENDENTE
FOR EACH ROW
DECLARE
    CONTATORE NUMBER(1,0):=0;
    TROPPI_RUOLI EXCEPTION;
BEGIN
    SELECT COUNT(*)
    INTO CONTATORE
    FROM DIPENDENTE
    WHERE CF_DIPENDENTE = :NEW.CF_DIPENDENTE;
    IF CONTATORE>1 THEN
        RAISE TROPPI_RUOLI;
    END IF;
EXCEPTION
    WHEN TROPPI_RUOLI THEN
        RAISE_APPLICATION_ERROR(-20001,'Un dipendente non può avere più di 2 ruoli');
    END;
```

## NUMERO MASSIMO DI TURNI

Questo trigger controlla che ogni volta che si aggiunge in orario\_turni, questo non superi il limite massimo di turni

```
CREATE OR REPLACE TRIGGER UN_TURNO
BEFORE INSERT ON ORARIO_TURNI
FOR EACH ROW
DECLARE
    CONTA_TURNI NUMBER(1,0) := 0;
    UN_TURNO EXCEPTION;
BEGIN
    SELECT COUNT(*)
    INTO CONTA_TURNI
    FROM ORARIO_TURNI
    WHERE CF_DIPENDENTE = :NEW.CF_DIPENDENTE
    AND DATA_INIZIO_TURNI = :NEW.DATA_INIZIO_TURNI;

    IF CONTA_TURNI >= 1 THEN
        RAISE UN_TURNO;
    END IF;
EXCEPTION
    WHEN UN_TURNO THEN
        RAISE_APPLICATION_ERROR(-20002, 'Il dipendente può avere al massimo un turno al giorno');
    END;
```

## NUMERO MASSIMO DI ATTIVITA GIORNALIERE

Questo trigger controlla che ogni volta che si aggiunge in attivita, un ospite non superi il limite di attivita consentite in una giornata

```
CREATE OR REPLACE TRIGGER TROPPE_ATTIVITA
BEFORE INSERT ON ATTIVITA
FOR EACH ROW
DECLARE
    CONTATORE NUMBER(1,0):=0;
    TROPPE_ATTIVITA EXCEPTION;
BEGIN
    SELECT COUNT(*)
    INTO CONTATORE
    FROM ATTIVITA
    WHERE NUM_PREN = :NEW.NUM_PREN
    AND DATA_ATTIVITA = :NEW.DATA_ATTIVITA;

    IF CONTATORE >= 2 THEN
        RAISE TROPPE_ATTIVITA;
    END IF;
EXCEPTION
    WHEN TROPPE_ATTIVITA THEN
        RAISE_APPLICATION_ERROR(-20003, 'Ogni ospite può scegliere al massimo due attività al giorno');
    END;
```

## ETA MINIMA

Questo trigger controlla che ogni volta che si aggiunge in persona, l'età non sia inferiore alla maggiore età

```
CREATE OR REPLACE TRIGGER ETA_BASSA
BEFORE INSERT ON PERSONA
FOR EACH ROW
DECLARE
    eta NUMBER(2,0):=0;
    ETA_BASSA EXCEPTION;
BEGIN
    SELECT EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM
:NEW.DATA_NASCITA)
    INTO eta
    FROM DUAL;

    IF eta < 18 THEN
        RAISE ETA_BASSA;
    END IF;
    EXCEPTION
    WHEN ETA_BASSA THEN
        RAISE_APPLICATION_ERROR(-20004,'Una persona non puo essere minorenne');
    END;
```

## NUMERO MASSIMO SEDI FORNITORI

Questo trigger controlla che un fornitore non abbia più di due sedi

```
CREATE OR REPLACE TRIGGER TROPPE_SEDI
BEFORE INSERT ON SEDI
FOR EACH ROW
DECLARE
    CONTATORE NUMBER(1,0):=0;
    TROPPE_SEDI EXCEPTION;
BEGIN
    SELECT COUNT(*)
    INTO CONTATORE
    FROM SEDI
    WHERE LOCALITA= :NEW.LOCALITA;
    IF CONTATORE >= 2 THEN
        RAISE TROPPE_SEDI;
    END IF;
    EXCEPTION
    WHEN TROPPE_SEDI THEN
        RAISE_APPLICATION_ERROR(-20005, 'Ogni fornitore puo avere al massimo due
sedi');
    END;
```

## VERIFICA 1 PRENOTAZIONE 1 PROMOZIONE

Con questo trigger verifico che per ogni prenotazione venga corrisposta una sola promozione

```
CREATE OR REPLACE TRIGGER TROPPE_PROMOZIONI
BEFORE INSERT ON PROMOZIONE
FOR EACH ROW
DECLARE
CONTATORE NUMBER(1,0):=0;
TROPPE_PROMOZIONI EXCEPTION;
BEGIN
SELECT COUNT(*)
INTO CONTATORE
FROM PROMOZIONE
WHERE NOME_PROMOZIONE= :NEW.NOME_PROMOZIONE
AND DATA_INIZIO= :NEW.DATA_INIZIO;
IF CONTATORE >1 THEN
RAISE TROPPE_PROMOZIONI;
END IF;
EXCEPTION
WHEN TROPPE_PROMOZIONI THEN
RAISE_APPLICATION_ERROR(-20006, 'Per ogni prenotazione puo corrispondere
al massimo una promozione');
END;
```

## MINIMO SALARIALE DIPENDENTI

Con questo trigger verifico prima di inserire in tipo\_contratto\_d se il salario corrisponde almeno al minimo salariale

```
CREATE OR REPLACE TRIGGER MIN_SALARIO
BEFORE INSERT ON TIPO_CONTRATTO_D
FOR EACH ROW
DECLARE
MIN_SALARIO EXCEPTION;
BEGIN
IF :NEW.STIPENDIO < 16000 THEN
RAISE MIN_SALARIO;
END IF;
EXCEPTION
WHEN MIN_SALARIO THEN
RAISE_APPLICATION_ERROR(-20008, 'Lo stipendio inserito deve essere almeno
di 16000');
END;
```

## MASSIMO IMPORTO TRIMESTRALE

Con questo trigger verifichiamo prima di inserire in tipo\_contratto\_f che l'importo trimestrale non sia superiore al limite consentito

```
CREATE OR REPLACE TRIGGER MAX_IMPORTO
BEFORE INSERT ON TIPO_CONTRATTO_F
FOR EACH ROW
DECLARE
MAX_IMPORTO EXCEPTION;
BEGIN
IF :NEW.IMPORTO_TRIMESTRALE > 10000 THEN
RAISE MAX_IMPORTO;
END IF;
EXCEPTION
WHEN MAX_IMPORTO THEN
RAISE_APPLICATION_ERROR(-20009, 'Per ogni fornitore il massimo importo
trimestrale deve essere di 10000');
END;
```

## VALIDITA' CONTRATTI DEI DIPENDENTI

Con questo trigger verifichiamo la validità dei contratti dei dipendenti, poichè ogni dipendente può avere al massimo 1 contratto attivo

```
CREATE OR REPLACE TRIGGER UN_CONTR_ATT
BEFORE INSERT ON TIPO_CONTRATTO_D
FOR EACH ROW
DECLARE
CONTA_CONTRATTI NUMBER;
UN_CONTR_ATT EXCEPTION;
BEGIN
SELECT COUNT(*)
INTO CONTA_CONTRATTI
FROM TIPO_CONTRATTO_D
WHERE CODICE_IDENTIFICATIVO_D = :NEW.CODICE_IDENTIFICATIVO_D;

IF CONTA_CONTRATTI >= 1 THEN
RAISE UN_CONTR_ATT;
END IF;
EXCEPTION
WHEN UN_CONTR_ATT THEN
RAISE_APPLICATION_ERROR(-20010, 'Il dipendente può avere al massimo un
contratto attivo');
END;
```

# 2.5 PROCEDURE E FUNZIONI

Le procedure sono legate alla logica di business e all'automazione; Queste sono sequenze di istruzioni che permettono la facilitazione di molteplici operazioni e consentono la riutilizzabilità del codice.

## CERCA PRENOTAZIONI OSPITI

Con questa procedura cerchiamo le informazioni riguardanti l'ospite e la prenotazione come il nome e la data di prenotazione

```
CREATE OR REPLACE PROCEDURE CercaPrenotazioniOspite(
    p_CF_OSPITE CHAR
) IS
BEGIN
    FOR r IN (SELECT
    P.NOME,P.COGNOME,P.CF,PP.NUMERO_PRENOTAZIONE,PP.DATA_PRENOTAZIONE
    FROM PERSONA P JOIN OSPITE O ON P.CF=O.CF_OSPITE
    JOIN PRENOTAZIONE PP ON PP.CF_OSPITE_PRENOTANTE=O.CF_OSPITE
    WHERE CF_OSPITE_PRENOTANTE = p_CF_OSPITE
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE(' Nome: ' || r.NOME || ', Cognome: ' || r.COGNOME || ',
        CF ospite: ' || r.CF || ', Numero Prenotazione: ' || r.NUMERO_PRENOTAZIONE || ', Data
        Prenotazione: ' || r.DATA_PRENOTAZIONE);
    END LOOP;
END;
ESEMPIO DI ESECUZIONE
BEGIN CercaPrenotazioniOspite('CCDCBBBBCCCC12EE');END;
```

## CERCA DIPENDENTE PER MANSIONE

Con questa procedura cerchiamo le informazioni riguardanti dipendente e la mansione che ricopre

```
CREATE OR REPLACE PROCEDURE CercaDipendentiPerMansione(
    p_MANSIONE VARCHAR
) IS
BEGIN
    FOR r IN (SELECT D.CF_DIPENDENTE, P.NOME, P.COGNOME
              FROM DIPENDENTE D
              JOIN PERSONA P ON D.CF_DIPENDENTE = P.CF
              JOIN RUOLO R ON D.CODICE_BADGE_D = R.CODICE_BADGE
              WHERE R.MANSIONE = p_MANSIONE)
    LOOP
        DBMS_OUTPUT.PUT_LINE('CF Dipendente: ' || r.CF_DIPENDENTE || ', Nome: ' ||
r.NOME || ', Cognome: ' || r.COGNOME);
    END LOOP;
END;
```

PER RICERCARE ESEMPIO:

```
BEGIN
    CercaDipendentiPerMansione('BARISTA');
END;
esempio di esecuzione
begin CercaDipendentiPerMansione('BARISTA');end;
```

## CERCA FORNITORI PER TIPO

Con questa procedura riprendo le informazioni riguardo i fornitori le sedi e le tipologie di forniture a riguardo

```
CREATE OR REPLACE PROCEDURE CercaFornitoriPerTipologia(
    p_TIPOLOGIA VARCHAR
) IS
BEGIN
    FOR r IN (SELECT F.CF_FORNITORE, P.NOME, P.COGNOME, S.LOCALITA
              FROM FORNITORE F
              JOIN SEDI S ON F.CF_FORNITORE = S.FORNITORE_CF
              JOIN PERSONA P ON F.CF_FORNITORE = P.CF
              WHERE F.TIPOLOGIA = p_TIPOLOGIA)
    LOOP
        DBMS_OUTPUT.PUT_LINE('CF Fornitore: ' || r.CF_FORNITORE || ', Nome: ' ||
r.NOME || ', Cognome: ' || r.COGNOME || ', Localita: ' || r.LOCALITA);
    END LOOP;
END;
esempio di esecuzione
begin CercaFornitoriPerTipologia('Bibite');end;
```

## CERCA TURNI PER DIPENDENTE

Con questa procedura cerco le info del dipendente e del turno che deve svolgere

```
CREATE OR REPLACE PROCEDURE CercaTurniPerDipendente(
    p_CF_DIPENDENTE CHAR
) IS
BEGIN
    FOR r IN (SELECT T.NOME_TURNO, TO_CHAR(T.INIZIO_TURNO, 'HH24:MI:SS') AS
    INIZIO_TURNO,
                  TO_CHAR(T.FINE_TURNO, 'HH24:MI:SS') AS FINE_TURNO, P.NOME,
    P.COGNOME
        FROM ORARIO_TURNI OT
        JOIN TURNO T ON OT.NOME_TURNO = T.NOME_TURNO
        JOIN PERSONA P ON OT.CF_DIPENDENTE = P.CF
        WHERE OT.CF_DIPENDENTE = p_CF_DIPENDENTE)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Nome Turno: ' || r.NOME_TURNO || ', Inizio Turno: ' ||
r.INIZIO_TURNO || ', Fine Turno: ' || r.FINE_TURNO || ', Nome: ' || r.NOME || ',
Cognome: ' || r.COGNOME);
    END LOOP;
END;
esempio di esecuzione
begin CercaTurniPerDipendente('FFFFDDDD1616DDCC');end;
```

## RINNOVO CONTRATTI

Con questa procedura rinnovo automaticamente i contratti a chi li ha in scadenza nell'anno corrente

```
CREATE OR REPLACE PROCEDURE RINNOVA_CONTRATTI
IS
BEGIN
    UPDATE TIPO_CONTRATTO_D
    SET DATA_SCADENZA = ADD_MONTHS(DATA_SCADENZA, 12)
    WHERE DATA_SCADENZA BETWEEN TO_DATE('01-JAN-2024', 'DD-MON-YYYY')
    AND TO_DATE('31-DEC-2024', 'DD-MON-YYYY');

    COMMIT;
END;
```

## CERCA PREN PER ZONA DI PRELIEVO

Con questa procedura identifico una prenotazione e le informazioni a riguardo tramite la zona di prelievo memorizzata

```
CREATE OR REPLACE PROCEDURE CercaPrenotazioniPerZonaPrelievo(
    p_ZONA_PRELIEVO VARCHAR
) IS
BEGIN
    FOR r IN (SELECT N.N_PRENOTAZIONE_NCC, O.CF_OSPITE, P.NOME,
    P.COGNOME
        FROM NCC N
        JOIN OSPITE O ON N.CF_RICHIEDENTE = O.CF_OSPITE
        JOIN PERSONA P ON O.CF_OSPITE = P.CF
        WHERE N.ZONA_PRELIEVO_OSP = p_ZONA_PRELIEVO)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Numero Prenotazione: ' || r.N_PRENOTAZIONE_NCC
    || ', CF Ospite: ' || r.CF_OSPITE || ', Nome: ' || r.NOME || ', Cognome: ' || r.COGNOME);
    END LOOP;
END;
esempio di esecuzione
begin CercaPrenotazioniPerZonaPrelievo('PIAZZA GARIBALDI');end;
```