



# Progetto Guitart

Corso di basi di dati e laboratorio

da un idea di,

**Leonardo e  
Nicola**

ANNO ACCADEMICO

**2022–2023**

**DATA DI CONSEGNA:**  
**LUGLIO 16 2023**

**STUDENTI:**

0124002551  
**NICOLA MENALE**

0124002501  
**LEONARDO PULZONE**

# INDICE

## ↓ PROGETTAZIONE

1.1 sintesi dei requisiti	<b>0</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>3B</b>
1.6 Operazioni tra gli utenti	<b>4</b>
1.7 Tavola dei Volumi	<b>5–8</b>
1.8 Vincoli di integrità	<b>9</b>
1.9 Vincoli di Normalità	<b>11</b>
	<b>12</b>

## ↓ IMPLEMENTAZIONE

2.1 Utenti del DB	<b>13–14</b>
2.1.a Creazione degli utenti	<b>15</b>
2.1.b Permessi degli utenti	<b>16</b>
2.2 Data definition language	<b>17–19</b>
2.3 Data Modify Language	<b>20–28</b>
2.4 TRIGGER	<b>29–48</b>
2.5 Procedure e funzioni	<b>49–52</b>
2.6 Viste	<b>53–56</b>
	<b>57</b>
2.7 SHEDULER	<b>58</b>



# PROGETTAZIONE





# 1.1 SINTESI DEI REQUISITI

Il database sviluppato nasce con l'intento di risolvere il problema di gestione dei dati dell'azienda (REALE) GUITART. Il database è stato progettato per soddisfare le esigenze specifiche dell'azienda. Offre una soluzione per gestire i prodotti, tracciare gli ordini e le spedizioni dei clienti registrati.

# 1.2 GLOSSARIO

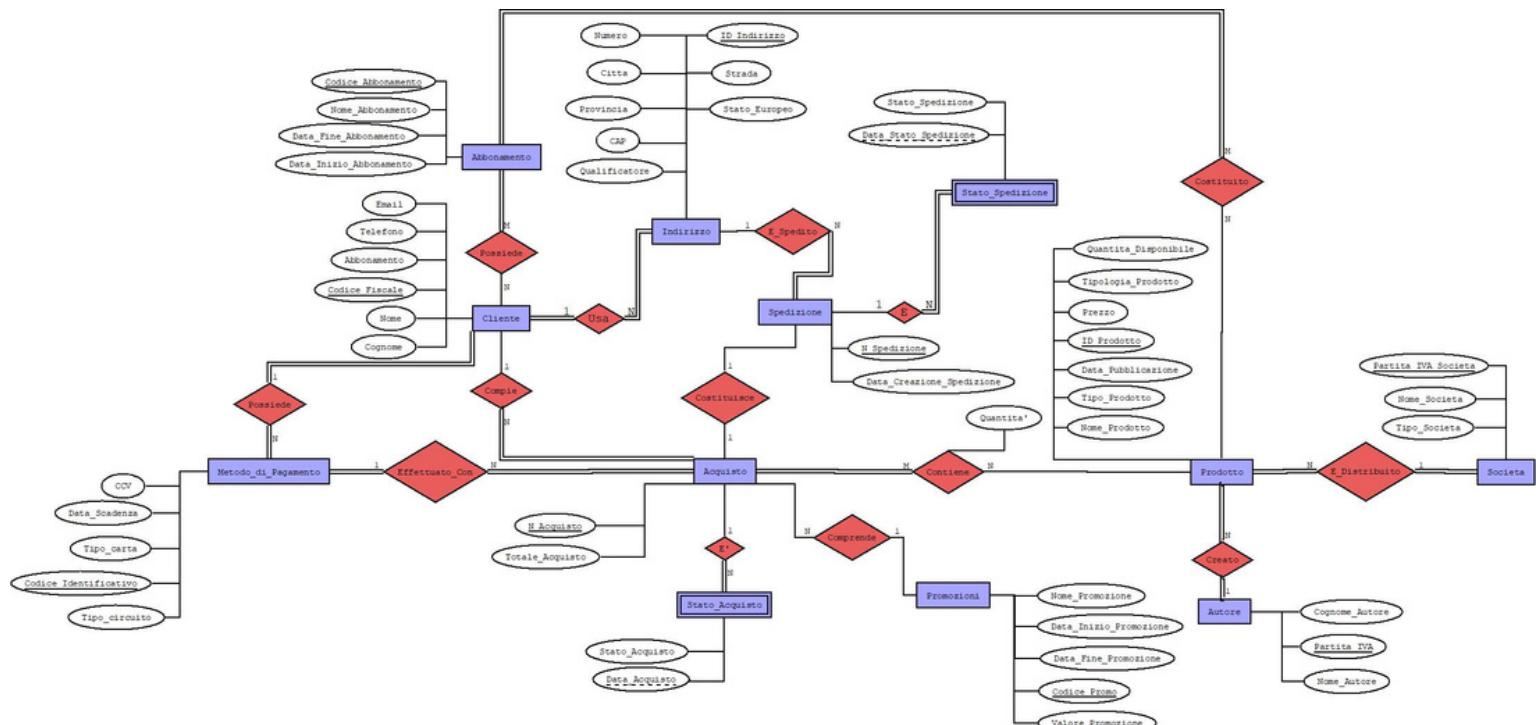
I glossario ha come fine ultimo di chiarire la Terminologia utilizzata in questo database e (anche se assente) il “gergo tecnico”

TERMINI	DEFINIZIONE	SINONIMI	ONONIMI
ID_PRODUTO_CONTIENE	Codice identificativo del prodotto venduto dall'azienda guirart	-	
CODICE_ABBONAMENTO_COSTITUITO	Alternative 2	-	
N_ACQUISTO	Numero identificativo dell'acquisto di un cliente	-	
N_SPEDIZIONE	Numero identificativo della spedizione associato all'acquisto di un cliente, il numero della spedizione dipende dal agenzia scelta di corrieri	-	
STATO_AQUISTO	Un attributo che identifica se l'acquisto è stato confermato, respinto, in elaborazione, cancellato	-	
ID_INDIRIZZO	E UN IDENTIFICATIVO CHE ASSOCIA UNA PERSONA A UN GRUPPO DI INDIRIZZI	-	

# 1.3 DIAGRAMMA EE/R

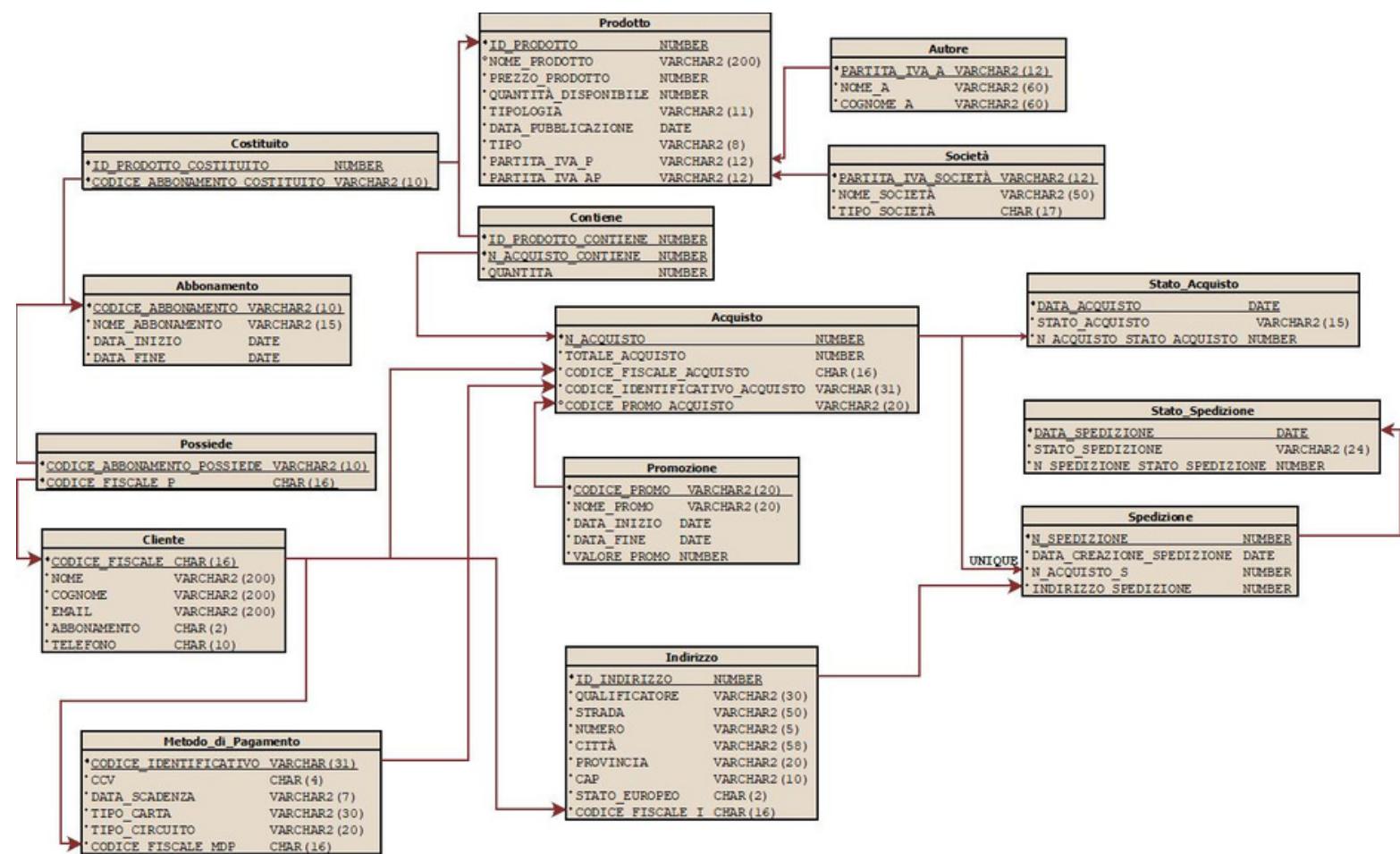
Di seguito abbiamo il diagramma EE/R.

L'entità cliente ha l'attributo Abbonamento che identifica se il cliente ha almeno un abbonamento attivo(1) o no(0).



# 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
Dipendente	Gestisce l'e-commerce
Cliente	Compie acquisti e può vedere le informazioni associate all'acquisto

# 1.6 OPERAZIONI TRA UTENTI

Le operazioni fra utenti sono operazioni implementate tramite procedure.

Vengono riportate le descrizioni delle varie procedure:

Operazioni	<b>VENDITE_PRODOTTI</b>
Scopo	Mostra quante volte ogni prodotto è stato venduto
Argomento	-
Risultato	Mostra il nome del prodotto e il numero di volte che è stato venduto
Errori	-
Modifica	-
Usa	Contiene, Prodotto
Prima	-
Poi	Visualizzazione il numero di volte il cui prodotto è stato venduto

Operazioni	TOTALE_ACQUISTO
Scopo	Calcola il totale di un acquisto
Argomento	Codice Fiscale, N_Acquisto
Risultato	Mostra il totale dell'acquisto
Errori	Se gli argomenti inseriti non corrispondono ad alcun ordine o cliente
Modifica	-
Usa	Cliente, Acquisto, Contiene, Prodotto
Prima	-
Poi	Visualizza il totale dell'ordine

<b>Operazioni</b>	<b>INSERIMENTO_INDIRIZZO</b>
Scopo	Inserire un nuovo indirizzo nella tabella degli indirizzi
Argomento	Id_Indirizzo, Codice_Fiscale_Indirizzo, Qualificatore, Strada, Numero, Città, Provincia, CAP, Stato_Europeo
Risultato	Inserisce un nuovo indirizzo nella tabella degli indirizzi
Errori	Se gli argomenti inseriti non corrispondono ad alcun Codice Fiscale associato ad un cliente oppure si è raggiunto il limite massimo di indirizzi
Modifica	Indirizzo
Usa	Indirizzo
Prima	Si ha almeno 1 indirizzo
Poi	Si aggiunge 1 nuovo indirizzo fino ad un massimo di 5

<b>Operazioni</b>		<b>CANCELLAZIONE_ACQUISTO</b>
Scopo	Cancellare un ordine effettuato da un cliente	
Argomento	Codice_Fiscale, N_Acquisto, N_Spedizione	
Risultato	Cancella un acquisto	
Errori	Se gli argomenti inseriti non corrispondono ad alcun ordine o cliente oppure non è possibile cancellarlo	
Modifica	Stato_Acquisto, Stato_Spedizione	
Usa	Acquisto, Spedizione, Stato_Acquisto, Stato_Spedizione	
Prima	Si ha un ordine attivo	
Poi	Si ha un ordine cancellato	

# 1.7 I VOLUMI

I valori sono stati inseriti nella tabella in un modo pseudo-realistico, basato su una stima ottimista che l'azienda stia performando bene e che ci troviamo in una situazione economica positiva. La tavola dei volumi è illustrata nella tabella sottostante. Viene mostrato un numero di tuple presenti in ogni tabella, così come il loro incremento previsto in un determinato periodo di tempo. In questo scenario, supponiamo che le tuple più antiche non vengano eliminate dal database, e che vi sia una crescita periodica in tutte le relazioni.

TABELLA	TIPO	VOLUME	INCREMENTO	PERIODO
CLIENTE	E	2K	10-12	SETTIMANALE
METODO DI PAGAMENTO	E	3K	<=8	MENSILE
ABBONAMENTO	E	2	1	ANNUALE
PRODOTTO	E	150	10	ANNUALE
ACQUISTO	E	2.5K	10-12	SETTIMANALE

STATO AQUISTO	ED	2.5K	10-12	SETTIMANALE
INDIRIZZO	E	4K	20-24	SETTIMANALE
PROMOZIONE	E	25	1-2	MESE
SPEDIZIONE	E	2K	7-9	SETTIMANALE
STATO SPEDIZIONE	ED	2K	7-9	SETTIMANALE
AUTORE	E	20	1	TRIMESTRALE
SOCIETA	E	5	1	DUE ANNI
POSSIEDE	A	4K	55	MENSILE
CONTIENE	A	5K	15	SETTIMANALE
COSTITUITO	A	200	1	ANNUALE

legenda:

E = Entità

ED = Entità debole

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'inserimento, l'aggiornamento e l'eliminazione dei dati.

Abbiamo 2 tipi di vincoli:

## 01 STATICI

- L'email è univoca
- Il numero di telefono e il CAP non contengono lettere
- La email ha la seguente struttura:  
nome\_utente@dominio.estensione
- La data di partenza della spedizione precede la data di arrivo
- La data di inizio dell'abbonamento precede la data di fine

## 02 DINAMICI

- Ogni cliente può avere un massimo di 5 indirizzi
- Ogni cliente può avere un massimo di 5 metodi di pagamento
- In ogni acquisto, ogni prodotto può essere acquistato un massimo di 10 volte
- Un vincolo per controllare se ci sono abbastanza scorte
- La data dello stato della spedizione non può precedere la data di acquisto
- Un vincolo che aggiorna la quantità di prodotti dopo un acquisto

# 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 Stato\_Acquisto e Stato\_Spedizione. Per entrambe le relazioni, le chiavi non presentano alcuna dipendenza parziale con altri attributi.

## TERZA FORMA NORMALE

Lo schema non presenta dipendenze anomalie.

## BCNF

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

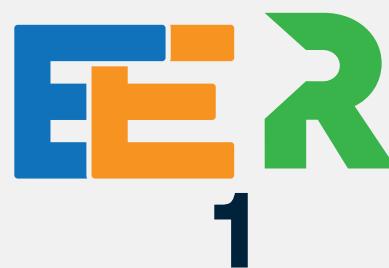
# IMPLEMENTAZIONE



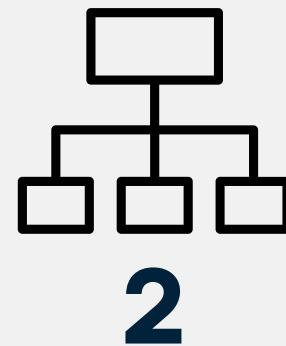
# Implementazione del Progetto con PL/SQL

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



3

Programmare pl/sql



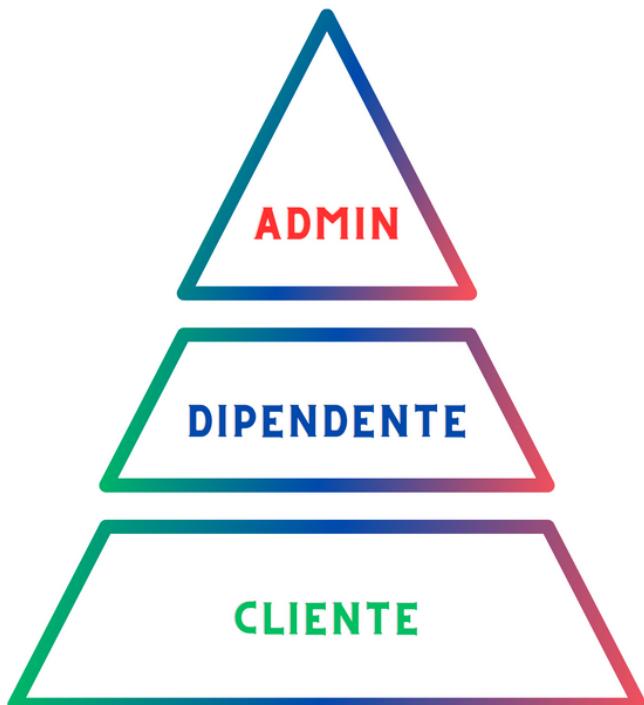
4

Base dati guitar

ci troviamo in questa fase!

## 2.1 UTENTI:

IL NOSTRO DATABASE CONSTA DI 3 UTENTI (ADMIN, DIPENDENTE, CLIENTE) OGNUNO CON IL PROPRIO LIVELLO GERARCHICO come riportato in figura:



descrizione grafico: si legge dal alto verso il basso, ove il più alto ha più permessi nel DB.

## 2.1.a CREAZIONE DEGLI UTENTI:

### ADMIN

QUESTO BLOCCO DI CODICE CREA UN NUOVO UTENTE NEL DATABASE CHIAMATO RESPONSABILE\_ADMIN CON LA PASSWORD ADMIN. SUCCESSIVAMENTE, VENGONO CONCESSI TUTTI I PRIVILEGI A QUESTO UTENTE, IL CHE SIGNIFICA CHE RESPONSABILE\_ADMIN PUÒ ESEGUIRE QUALSIASI OPERAZIONE SUL DATABASE, INCLUSI SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT, REVOKE E COSÌ VIA.

#### CODE

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

### DIPENDENTE

QUESTO BLOCCO DI CODICE CREA UN NUOVO UTENTE NEL DATABASE CHIAMATO DIPENDENTE CON LA PASSWORD PASSWORD\_DIPENDENTE. SUCCESSIVAMENTE, VIENE CONCESSO VARI PRIVILEGI

#### CODE

```
//DIPENDENTE  
CREATE USER DIPENDENTE IDENTIFIED BY Ndipendente;  
--PERMESSI DEFINITI 2.1.b
```

### CLIENTE

QUESTO BLOCCO DI CODICE CREA UN NUOVO UTENTE NEL DATABASE CHIAMATO CLIENTE CON LA PASSWORD PASSWORD\_CLIENTE. SUCCESSIVAMENTE, VIENE CONCESSO VARI PRIVILEGI RIPORTATI IN 2.1.B

#### CODE

```
//CLIENTE  
CREATE USER CLIENTE IDENTIFIED BY Ncliente;  
--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.

### 2.1.a.a PERMESSI ADMIN

**RESPONSABILE\_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, come la creazione, modifica o eliminazione di tabella, viste, indici, procedure memorizzate, ecc.

- GRANT ALL PRIVILEGES to RESPONSABILE\_ADMIN;
- GRANT CONNECT, CREATE SESSION TO RESPONSABILE\_ADMIN;

## 2.1.a.b PERMESSI DIPENDENTE

DIPENDENTE\_USER: Questo utente ha privilegi limitati che gli permettono di eseguire solo determinate operazioni. GRANT CREATE SESSION gli permette di avviare una sessione nel database. Inoltre, ha il permesso di visualizzare (SELECT) i dati da una serie di tabelle specifiche, come INVENTARIO, TURNO, COMPRENDE, ecc. Questo significa che può leggere i dati da queste tabelle, ma non può modificarli o eliminarli.

- GRANT CONNECT, CREATE SESSION TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON INVENTARIO TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON TURNO TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON COMPRENDE TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON CONTIENE TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON COSTITUITO TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON POSSIEDE TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON ABBONAMENTO TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON PRODOTTO TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON SOCIETA TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON PROMOZIONE TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON AUTORE TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON STATO\_ACQUISTO TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON STATO\_SPEDIZIONE TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON SPEDIZIONE TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON ACQUISTO TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON METODO\_DI\_PAGAMENTO TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON INDIRIZZO TO DIPENDENTE;
- GRANT SELECT, DELETE, UPDATE ON CLIENTE TO DIPENDENTE;
- GRANT EXECUTE ON VENDITE\_PRODOTTI TO DIPENDENTE;
- GRANT EXECUTE ON TOTALE\_ACQUISTO TO DIPENDENTE;
- GRANT EXECUTE ON INSERIMENTO\_INDIRIZZO TO DIPENDENTE;
- GRANT EXECUTE ON CANCELLAZIONE\_ACQUISTO TO DIPENDENTE;
- GRANT SELECT VIEW\_ORDINI TO DIPENDENTE;

## 2.1.a.c PERMESSI CLIENTE

**CLIENTE:** 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 VISUALIZZA\_ORDINI, che presumibilmente gli permette di vedere i propri ordini.

- GRANT CONNECT TO CLIENTE;
- GRANT SELECT VIEW\_ORDINI TO CLIENTE;
- GRANT EXECUTE ON INSERIMENTO\_INDIRIZZO TO CLIENTE;
- GRANT EXECUTE ON CANCELLAZIONE\_ACQUISTO TO CLIENTE;

## 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 vendita online di immagazzinare e gestire le informazioni relative ai clienti, ai prodotti, agli acquisti, alle spedizioni e agli abbonamenti; in aggiunta, sono presenti vari vincoli di integrità.

### **CLIENTE:**

La tabella "CLIENTE" memorizza le informazioni relative ai clienti, includendo il loro codice fiscale (che funge da chiave primaria), nome, cognome, email, stato dell'abbonamento (che può essere 'SI' o 'NO') e numero di telefono. Tutti i campi, ad eccezione dell'abbonamento, sono obbligatori.

```
CREATE TABLE CLIENTE(
CODICE_FISCALE CHAR(16) PRIMARY KEY,
NOME VARCHAR2(200) NOT NULL,
COGNOME VARCHAR2(200) NOT NULL,
EMAIL VARCHAR2(200) NOT NULL UNIQUE,
ABBONAMENTO NUMBER(1) NOT NULL,
TELEFONO VARCHAR2(20) NOT NULL,
CONSTRAINT CHECK_TELEFONO CHECK (REGEXP_LIKE(TELEFONO, '^[0-9]+$')),
CONSTRAINT CHECK_ABBONAMENTO CHECK(ABBONAMENTO = 0 OR ABBONAMENTO = 1),
CONSTRAINT CHECK_EMAIL CHECK (EMAIL LIKE '%@%.%' AND EMAIL NOT LIKE '@%' AND EMAIL NOT LIKE '%@%@%')
);
```

## **METODO\_DI\_PAGAMENTO:**

La tabella "METODO\_DI\_PAGAMENTO" memorizza le informazioni relative ai metodi di pagamento dei clienti, includendo un codice identificativo unico (che funge da chiave primaria), il codice di sicurezza della carta (CCV), la data di scadenza della carta, il tipo di carta (che può essere 'CARTA DI CREDITO' o 'CARTA PREPAGATA'), il circuito della carta (che può essere 'VISA', 'MASTERCARD' o 'AMERICAN EXPRESS') e il codice fiscale del cliente associato al metodo di pagamento. Tutti i campi sono obbligatori. Inoltre, il codice fiscale del cliente è una chiave esterna che fa riferimento al campo "CODICE\_FISCALE" nella tabella "CLIENTE".

```
CREATE TABLE METODO_DI_PAGAMENTO(
  CODICE_IDENTIFICATIVO VARCHAR2(31) PRIMARY KEY,
  CCV VARCHAR2(4) NOT NULL,
  DATA_SCADENZA VARCHAR2(7) NOT NULL,
  TIPO_CARTA VARCHAR2(30) NOT NULL,
  TIPO_CIRCUITO VARCHAR2(20) NOT NULL,
  CODICE_FISCALE_MDP CHAR(16) NOT NULL,
  CONSTRAINT FK_CODICE_FISCALE_MDP FOREIGN KEY(CODICE_FISCALE_MDP)
  REFERENCES CLIENTE(CODICE_FISCALE),
  CONSTRAINT CHECK_CODICE CHECK (REGEXP_LIKE(CODICE_IDENTIFICATIVO,
  '^[0-9]+$')),
  CONSTRAINT CHECK_CCV CHECK (REGEXP_LIKE(CCV, '^[0-9]+$')),
  CONSTRAINT CHECK_CARTA CHECK(TIPO_CARTA='CARTA DI CREDITO' OR
  TIPO_CARTA='CARTA PREPAGATA'),
  CONSTRAINT CHECK_CIRCUITO CHECK(TIPO_CIRCUITO='VISA' OR
  TIPO_CIRCUITO='MASTERCARD' OR TIPO_CIRCUITO='AMERICAN EXPRESS')
);
```

**AUTORE:**

Questa tabella memorizza le informazioni sugli autori dei prodotti, come la partita IVA, il nome e il cognome.

```
CREATE TABLE AUTORE(
PARTITA_IVA_AUTORE VARCHAR2(12) PRIMARY KEY,
NOME_AUTORE VARCHAR2(60) NOT NULL,
COGNOME_AUTORE VARCHAR2(60) NOT NULL
);
```

**PROMOZIONE:**

Questa tabella contiene le informazioni relative alle promozioni disponibili, come il codice della promozione, il nome, il valore e le date di inizio e fine.

```
CREATE TABLE PROMOZIONE(
CODICE_PROMOZIONE VARCHAR2(20) PRIMARY KEY,
NOME_PROMOZIONE VARCHAR2(20) NOT NULL,
VALORE_PROMOZIONE NUMBER NOT NULL,
DATA_INIZIO_PROMOZIONE DATE NOT NULL,
DATA_FINE_PROMOZIONE DATE NOT NULL
);
```

**ACQUISTO:**

Questa tabella registra le informazioni sugli acquisti effettuati dai clienti, tra cui il numero dell'acquisto, i riferimenti al cliente che ha effettuato l'acquisto, il metodo di pagamento utilizzato e la promozione applicata, se presente.

```
CREATE TABLE ACQUISTO(
N_ACQUISTO NUMBER PRIMARY KEY,
CODICE_FISCALE_ACQUISTO CHAR(16) NOT NULL,
CONSTRAINT FK_CLIENTE_ACQUISTO FOREIGN KEY(CODICE_FISCALE_ACQUISTO)
REFERENCES CLIENTE(CODICE_FISCALE),
CODICE_IDENTIFICATIVO_ACQUISTO VARCHAR(31) NOT NULL,
CONSTRAINT FK_MDP_ACQUISTO FOREIGN
KEY(CODICE_IDENTIFICATIVO_ACQUISTO) REFERENCES
METODO_DI_PAGAMENTO(CODICE_IDENTIFICATIVO),
CODICE_PROMOZIONE_ACQUISTO VARCHAR(20),
CONSTRAINT FK_CODICE_PROMOZIONE FOREIGN
KEY(CODICE_PROMOZIONE_ACQUISTO) REFERENCES
PROMOZIONE(CODICE_PROMOZIONE)
);
```

**STATO\_ACQUISTO:**

Questa tabella tiene traccia dello stato di ogni acquisto in un dato momento. Esso comprende la data dell'acquisto, lo stato (confermato, respinto, in elaborazione, cancellato) e un riferimento all'acquisto correlato.

```
CREATE TABLE STATO_ACQUISTO(
    DATA_ACQUISTO DATE NOT NULL,
    STATO_ACQUISTO VARCHAR2(15) NOT NULL,
    N_ACQUISTO_STATO_ACQUISTO NUMBER NOT NULL,
    CONSTRAINT PK_STATO_ACQUISTO PRIMARY KEY(DATA_ACQUISTO,
    N_ACQUISTO_STATO_ACQUISTO),
    CONSTRAINT FK_ACQUISTO_STATO_ACQUISTO FOREIGN
    KEY(N_ACQUISTO_STATO_ACQUISTO) REFERENCES ACQUISTO(N_ACQUISTO),
    CONSTRAINT CHECK_STATO_A CHECK(STATO_ACQUISTO='CONFERMATO' OR
    STATO_ACQUISTO='RESPINTO' OR STATO_ACQUISTO='IN ELABORAZIONE' OR
    STATO_ACQUISTO='CANCELLATO')
);
```

**SOCIETA:**

Questa tabella memorizza le informazioni sulle società associate ai prodotti, come la partita IVA, il nome e il tipo di società (casa discografica, casa editrice, etc.).

```
CREATE TABLE SOCIETA(
    PARTITA_IVA_SOCIETA VARCHAR2(12) PRIMARY KEY,
    NOME_SOCIETA VARCHAR2(50) NOT NULL,
    TIPO_SOCIETA CHAR(17) NOT NULL,
    CONSTRAINT CHECK_TIPO_S CHECK(TIPO_SOCIETA='CASA DISCOGRAFICA' OR
    TIPO_SOCIETA='CASA EDITRICE' OR TIPO_SOCIETA='GUITART')
);
```

**PRODOTTO:**

Questa tabella contiene le informazioni sui prodotti disponibili per l'acquisto, come l'ID del prodotto, il nome, il prezzo, la quantità disponibile, la tipologia, il tipo (fisico o digitale), la data di pubblicazione e i riferimenti alla società e all'autore associati al prodotto.

```
CREATE TABLE PRODOTTO(
ID_PRODOTTO NUMBER PRIMARY KEY,
NOME_PRODOTTO VARCHAR2(200) NOT NULL,
PREZZO_PRODOTTO NUMBER NOT NULL,
QUANTITA_DISPONIBILE_PRODOTTO NUMBER NOT NULL,
TIPOLOGIA_PRODOTTO VARCHAR2(11) NOT NULL,
TIPO_PRODOTTO VARCHAR2(8) NOT NULL,
DATA_PUBBLICAZIONE DATE NOT NULL,
PARTITA_IVA_PRODOTTO_SOCIETA VARCHAR2(12) NOT NULL,
PARTITA_IVA_PRODOTTO_AUTORE VARCHAR(12) NOT NULL,
CONSTRAINT FK_PARTITA_IVA_PRODOTTO_SOCIETA FOREIGN
KEY(PARTITA_IVA_PRODOTTO_SOCIETA) REFERENCES
SOCIETA(PARTITA_IVA_SOCIETA),
CONSTRAINT FK_PARTITA_IVA_PRODOTTO_AUTORE FOREIGN
KEY(PARTITA_IVA_PRODOTTO_AUTORE) REFERENCES
AUTORE(PARTITA_IVA_AUTORE),
CONSTRAINT CHECK_TIPOLOGIA_PRODOTTO
CHECK(TIPOLOGIA_PRODOTTO='MAGAZINE' OR
TIPOLOGIA_PRODOTTO='MONOGRAFIA' OR TIPOLOGIA_PRODOTTO='ABBONAMENTO' OR
TIPOLOGIA_PRODOTTO='E-BOOK' OR TIPOLOGIA_PRODOTTO='CD'),
CONSTRAINT CHECK_TIPO_PRODOTTO CHECK(TIPO_PRODOTTO='FISICO' OR
TIPO_PRODOTTO='DIGITALE')
);
```

**INDIRIZZO:**

Questa tabella memorizza le informazioni sugli indirizzi dei clienti per la spedizione dei prodotti fisici. Contiene dettagli come il qualificatore, la strada, il numero, la città, la provincia, il CAP e lo stato europeo.

```
CREATE TABLE INDIRIZZO(
ID_INDIRIZZO NUMBER PRIMARY KEY,
QUALIFICATORE VARCHAR2(30) NOT NULL,
STRADA VARCHAR2(50) NOT NULL,
NUMERO VARCHAR2(5) NOT NULL,
CITTA VARCHAR2(58) NOT NULL,
PROVINCIA VARCHAR2(20) NOT NULL,
CAP VARCHAR2(10) NOT NULL,
STATO_EUROPEO CHAR(2) NOT NULL,
CODICE_FISCALE_INDIRIZZO CHAR(16) NOT NULL,
CONSTRAINT FK_CODICE_FISCALE_I FOREIGN
KEY(CODICE_FISCALE_INDIRIZZO) REFERENCES CLIENTE(CODICE_FISCALE),
CONSTRAINT CHECK_STATO_EUROPEO CHECK(STATO_EUROPEO='IT' OR
STATO_EUROPEO='BE' OR STATO_EUROPEO='FR' OR STATO_EUROPEO='DE' OR
STATO_EUROPEO='LU' OR STATO_EUROPEO='NL' OR STATO_EUROPEO='DK' OR
STATO_EUROPEO='IE' OR STATO_EUROPEO='GB' OR STATO_EUROPEO='EL' OR
STATO_EUROPEO='PT' OR STATO_EUROPEO='ES' OR STATO_EUROPEO='AT' OR
STATO_EUROPEO='FI' OR STATO_EUROPEO='SE' OR STATO_EUROPEO='CY' OR
STATO_EUROPEO='EE' OR STATO_EUROPEO='LV' OR STATO_EUROPEO='LT' OR
STATO_EUROPEO='MT' OR STATO_EUROPEO='PL' OR STATO_EUROPEO='CZ' OR
STATO_EUROPEO='SK' OR STATO_EUROPEO='SI' OR STATO_EUROPEO='HU' OR
STATO_EUROPEO='BG' OR STATO_EUROPEO='RO' OR STATO_EUROPEO='CH' OR
STATO_EUROPEO='IS' OR STATO_EUROPEO='LI' OR STATO_EUROPEO='NO' OR
STATO_EUROPEO='HR')
);
```

**SPEDIZIONE:**

Questa tabella tiene traccia delle informazioni relative alle spedizioni, tra cui il numero della spedizione, la data di creazione, l'indirizzo di spedizione e un riferimento all'acquisto correlato.

```
CREATE TABLE SPEDIZIONE(
    N_SPEDIZIONE NUMBER PRIMARY KEY,
    DATA_CREAZIONE_SPEDIZIONE DATE NOT NULL,
    INDIRIZZO_SPEDIZIONE NUMBER NOT NULL,
    N_ACQUISTO_SPEDIZIONE NUMBER UNIQUE NOT NULL,
    CONSTRAINT FK_INDIRIZZO_SPEDIZIONE FOREIGN
    KEY(INDIRIZZO_SPEDIZIONE) REFERENCES INDIRIZZO(ID_INDIRIZZO),
    CONSTRAINT FK_N_ACQUISTO_SPEDIZIONE FOREIGN
    KEY(N_ACQUISTO_SPEDIZIONE) REFERENCES ACQUISTO(N_ACQUISTO)
);
```

**STATO\_SPEDIZIONE:**

Questa tabella registra lo stato di ogni spedizione in un dato momento, incluso la data, lo stato (in elaborazione, in attesa, pronto per la spedizione, etc.) e un riferimento alla spedizione correlata.

```
CREATE TABLE STATO_SPEDIZIONE(
    DATA_STATO_SPEDIZIONE DATE NOT NULL,
    STATO_SPEDIZIONE VARCHAR2(24) NOT NULL,
    N_SPEDIZIONE_STATO_SPEDIZIONE NUMBER NOT NULL,
    CONSTRAINT PK_STATO_SPEDIZIONE PRIMARY KEY(DATA_STATO_SPEDIZIONE,
    N_SPEDIZIONE_STATO_SPEDIZIONE),
    CONSTRAINT FK_N_SPEDIZIONE_STATO_SPEDIZIONE FOREIGN
    KEY(N_SPEDIZIONE_STATO_SPEDIZIONE) REFERENCES
    SPEDIZIONE(N_SPEDIZIONE),
    CONSTRAINT CHECK_STATO_SPEDIZIONE CHECK(STATO_SPEDIZIONE='IN
    ELABORAZIONE' OR STATO_SPEDIZIONE='IN ATTESA' OR
    STATO_SPEDIZIONE='PRONTO PER LA SPEDIZIONE' OR
    STATO_SPEDIZIONE='SPEDITO' OR STATO_SPEDIZIONE='IN TRANSITO' OR
    STATO_SPEDIZIONE='IN CONSEGNA' OR STATO_SPEDIZIONE='RESTITUITO' OR
    STATO_SPEDIZIONE='RESPINTO' OR STATO_SPEDIZIONE='CONSEGNATO' OR
    STATO_SPEDIZIONE='CANCELLATO')
);
```

**ABBONAMENTO:**

Questa tabella contiene dettagli sugli abbonamenti disponibili, come il codice dell'abbonamento, il nome e le date di inizio e fine.

```
CREATE TABLE ABBONAMENTO(
CODICE_ABBONAMENTO VARCHAR2(10) PRIMARY KEY,
NOME_ABBONAMENTO VARCHAR2(15) NOT NULL,
DATA_INIZIO_ABBONAMENTO DATE NOT NULL,
DATA_FINE_ABBONAMENTO DATE NOT NULL,
CONSTRAINT CHECK_DATA CHECK (DATA_INIZIO_ABBONAMENTO <
DATA_FINE_ABBONAMENTO)
);
```

**CONTIENE:**

Questa tabella di relazione registra quali prodotti sono inclusi in ogni acquisto e in quale quantità. Nel diagramma EE/R corrisponde alla associazione Contiene serve per dare vita alla molteciplità Molti-a-molti(m a n) tra la relazione Acquisto e Prodotto.

```
CREATE TABLE CONTIENE(
QUANTITA NUMBER NOT NULL,
N_ACQUISTO_CONTIENE NUMBER NOT NULL,
ID_PRODOTTO_CONTIENE NUMBER NOT NULL,
CONSTRAINT PK_CONTIENE PRIMARY KEY(N_ACQUISTO_CONTIENE,
ID_PRODOTTO_CONTIENE),
CONSTRAINT FK_ACQUISTO_CONTIENE FOREIGN KEY(N_ACQUISTO_CONTIENE)
REFERENCES ACQUISTO(N_ACQUISTO),
CONSTRAINT FK_PRODOTTO_CONTIENE FOREIGN KEY(ID_PRODOTTO_CONTIENE)
REFERENCES PRODOTTO(ID_PRODOTTO)
);
```

**POSSIENE(N-a-M):**

Questa tabella di relazione registra quali clienti possiedono quali abbonamenti.  
Nel diagramma EE/R corrisponde alla associazione Possiede e serve per dare vita alla **molteplicità** Molti-a-molti(m a n) tra la relazione Abbonamento e Cliente

```
CREATE TABLE POSSIEDE(
    CODICE_ABBONAMENTO_POSSIEDE VARCHAR2(10) NOT NULL,
    CODICE_FISCALE_POSSIEDE CHAR(16) NOT NULL,
    CONSTRAINT PK_POSSIEDE PRIMARY KEY(CODICE_ABBONAMENTO_POSSIEDE,
    CODICE_FISCALE_POSSIEDE),
    CONSTRAINT FK_ABBONAMENTO_POSSIEDE FOREIGN
    KEY(CODICE_ABBONAMENTO_POSSIEDE) REFERENCES
    ABBONAMENTO(CODICE_ABBONAMENTO),
    CONSTRAINT FK_CLIENTE_POSSIEDE FOREIGN
    KEY(CODICE_FISCALE_POSSIEDE) REFERENCES CLIENTE(CODICE_FISCALE)
);
```

- ATTENZIONE: non confondere con associazione POSSIENDE tra Cliente e Metodo\_di\_pagamento con molteplicità (1 a N) nel diagramma EE/R

**COSTITUITO:**

Questa tabella di relazione registra quali prodotti sono inclusi in ogni abbonamento.  
nel diagramma EE/R corrisponde alla associazione Costituito e serve per dare vita alla **molteplicità** Molti-a-molti(m a n) tra la relazione Abbonamento e Prodotto

```
CREATE TABLE COSTITUITO(
    CODICE_ABBONAMENTO_COSTITUITO VARCHAR2(10) NOT NULL,
    ID_PRODOTTO_COSTITUITO NUMBER NOT NULL,
    CONSTRAINT PK_COSTITUITO PRIMARY KEY(CODICE_ABBONAMENTO_COSTITUITO,
    ID_PRODOTTO_COSTITUITO),
    CONSTRAINT FK_ABBONAMENTO_COSTITUITO FOREIGN
    KEY(CODICE_ABBONAMENTO_COSTITUITO) REFERENCES
    ABBONAMENTO(CODICE_ABBONAMENTO),
    CONSTRAINT FK_PRODOTTO_COSTITUITO FOREIGN KEY(ID_PRODOTTO_COSTITUITO)
    REFERENCES PRODOTTO(ID_PRODOTTO)
);
```

## 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.

**CLIENTE**

```

INSERT INTO CLIENTE
(CODICE_FISCALE,NOME,COGNOME,EMAIL,ABBONAMENTO,TELEFONO)
VALUES
('SGSGNT80A01E654Y','Giacinto','Sagese','GiacintoSagese@armyspy.com','0','
3783173761');

INSERT INTO CLIENTE
(CODICE_FISCALE,NOME,COGNOME,EMAIL,ABBONAMENTO,TELEFONO)
VALUES
('MNFPLS92S04A892C','Manfredo','Pugliesi','ManfredoPugliesi@armyspy.com','
0','3924301632');

INSERT INTO CLIENTE
(CODICE_FISCALE,NOME,COGNOME,EMAIL,ABBONAMENTO,TELEFONO)
VALUES
('RSSDBR87D11D612M','Adalberto','Rossi','AdalbertoRossi@armyspy.com','0','
3372682684');

INSERT INTO CLIENTE
(CODICE_FISCALE,NOME,COGNOME,EMAIL,ABBONAMENTO,TELEFONO)
VALUES
('DMNNLT94L31H501X','Edmondo','Napolitano','EdmondoNapolitano@jourrapide.c
om','0','3634024381');

INSERT INTO CLIENTE
(CODICE_FISCALE,NOME,COGNOME,EMAIL,ABBONAMENTO,TELEFONO)
VALUES
('DLCRNT83S06D810W','Renato','DeLuca','RenatoDeLuca@teleworm.us','1','3820
411398');

INSERT INTO CLIENTE
(CODICE_FISCALE,NOME,COGNOME,EMAIL,ABBONAMENTO,TELEFONO)
VALUES
('TRVMRA62M13D150J','Maria','Trevisano','MariaTrevisano@armyspy.com','0','
3803322094');

INSERT INTO CLIENTE
(CODICE_FISCALE,NOME,COGNOME,EMAIL,ABBONAMENTO,TELEFONO)
VALUES
('CLBRNT82S70F205G','Renata','Calabrese','RenataCalabrese@tmall.com','1','
3114429983');

INSERT INTO CLIENTE
(CODICE_FISCALE,NOME,COGNOME,EMAIL,ABBONAMENTO,TELEFONO)
VALUES
('PLRRSR41B41E517N','Rosaria','Palermo','RosariaPalermo@jigsy.com','1','33
06736572');

INSERT INTO CLIENTE
(CODICE_FISCALE,NOME,COGNOME,EMAIL,ABBONAMENTO,TELEFONO)
VALUES
('FRNLFA83T14A390H','Alfio','Fiorentino','AlfioFiorentino@rhyta.com','1','
3392088632');

INSERT INTO CLIENTE
(CODICE_FISCALE,NOME,COGNOME,EMAIL,ABBONAMENTO,TELEFONO)
VALUES
('LMBLNS92H07F8390','Alfonsino','Lombardo','AlfonsinoLombardo@armyspy.com'
,'1','3677728814');

```

## ABBONAMENTO

```
INSERT INTO ABBONAMENTO
(CODICE_ABBONAMENTO,NOME_ABBONAMENTO,DATA_INIZIO_ABBONAMENTO,DATA_FINE_ABBONAMENTO)
VALUES
('ABB.CLS','CLASSIC','01-FEB-2023','30-NOV-2023');

INSERT INTO ABBONAMENTO
(CODICE_ABBONAMENTO,NOME_ABBONAMENTO,DATA_INIZIO_ABBONAMENTO,DATA_FINE_ABBONAMENTO)
VALUES
('ABB.DLX','DELUXE','01-FEB-2023','30-NOV-2023');
```

## AUTORE

```

INSERT INTO AUTORE
(PARTITA_IVA_AUTORE, NOME_AUTORE, COGNOME_AUTORE)
VALUES
('39999320052', 'GUITART', '-');

INSERT INTO AUTORE
(PARTITA_IVA_AUTORE, NOME_AUTORE, COGNOME_AUTORE)
VALUES
('23589960691', 'OSCAR', 'BELLLOMO');

INSERT INTO AUTORE
(PARTITA_IVA_AUTORE, NOME_AUTORE, COGNOME_AUTORE)
VALUES
('72650340796', 'GIANVITO', 'PULZONE');

INSERT INTO AUTORE
(PARTITA_IVA_AUTORE, NOME_AUTORE, COGNOME_AUTORE)
VALUES
('58128990544', 'FERDINANDO', 'CARULLI');

INSERT INTO AUTORE
(PARTITA_IVA_AUTORE, NOME_AUTORE, COGNOME_AUTORE)
VALUES
('63798360192', 'RAFFAELE', 'IERVOLINO');

INSERT INTO AUTORE
(PARTITA_IVA_AUTORE, NOME_AUTORE, COGNOME_AUTORE)
VALUES
('51619000519', 'MARIO', 'CARREIRA');

```

## SOCIETÀ

```

INSERT INTO SOCIETA
(PARTITA_IVA_SOCIETA, NOME_SOCIETA, TIPO_SOCIETA)
VALUES
('39999320052', 'GUITART', 'GUITART');

INSERT INTO SOCIETA
(PARTITA_IVA_SOCIETA, NOME_SOCIETA, TIPO_SOCIETA)
VALUES
('45350740630', 'RIZZOLI', 'CASA EDITRICE');

INSERT INTO SOCIETA
(PARTITA_IVA_SOCIETA, NOME_SOCIETA, TIPO_SOCIETA)
VALUES
('18977850363', 'MARSILIO', 'CASA EDITRICE');

INSERT INTO SOCIETA
(PARTITA_IVA_SOCIETA, NOME_SOCIETA, TIPO_SOCIETA)
VALUES
('34789260295', 'SELLERIO', 'CASA EDITRICE');

INSERT INTO SOCIETA
(PARTITA_IVA_SOCIETA, NOME_SOCIETA, TIPO_SOCIETA)
VALUES
('28146030045', 'FANUCCI', 'CASA EDITRICE');

INSERT INTO SOCIETA
(PARTITA_IVA_SOCIETA, NOME_SOCIETA, TIPO_SOCIETA)
VALUES
('64905180770', 'IPERBOREA', 'CASA EDITRICE');

```

## PRODOTTO

```

INSERT INTO PRODOTTO
(ID_PRODOTTO, NOME_PRODOTTO, PREZZO_PRODOTTO, QUANTITA_DISPONIBILE_PRODOTTO,
TIPOLOGIA_PRODOTTO, TIPO_PRODOTTO, DATA_PUBBLICAZIONE,
PARTITA_IVA_PRODOTTO_SOCIETA, PARTITA_IVA_PRODOTTO_AUTORE)
VALUES
('0042', 'ABBONAMENTO CLASSIC', '53,00', '10000', 'ABBONAMENTO', 'DIGITALE', '13-
GEN-2023', '39999320052', '39999320052');

INSERT INTO PRODOTTO
(ID_PRODOTTO, NOME_PRODOTTO, PREZZO_PRODOTTO, QUANTITA_DISPONIBILE_PRODOTTO,
TIPOLOGIA_PRODOTTO, TIPO_PRODOTTO, DATA_PUBBLICAZIONE,
PARTITA_IVA_PRODOTTO_SOCIETA, PARTITA_IVA_PRODOTTO_AUTORE)
VALUES
('0039', 'ABBONAMENTO DELUXE', '79,00', '10000', 'ABBONAMENTO', 'DIGITALE', '13-
GEN-2023', '39999320052', '39999320052');

INSERT INTO PRODOTTO
(ID_PRODOTTO, NOME_PRODOTTO, PREZZO_PRODOTTO, QUANTITA_DISPONIBILE_PRODOTTO,
TIPOLOGIA_PRODOTTO, TIPO_PRODOTTO, DATA_PUBBLICAZIONE,
PARTITA_IVA_PRODOTTO_SOCIETA, PARTITA_IVA_PRODOTTO_AUTORE)
VALUES
('0061', 'GUITART 102', '19,00', '150', 'MAGAZINE', 'FISICO', '27-LUG-
2017', '39999320052', '39999320052');

INSERT INTO PRODOTTO
(ID_PRODOTTO, NOME_PRODOTTO, PREZZO_PRODOTTO, QUANTITA_DISPONIBILE_PRODOTTO,
TIPOLOGIA_PRODOTTO, TIPO_PRODOTTO, DATA_PUBBLICAZIONE,
PARTITA_IVA_PRODOTTO_SOCIETA, PARTITA_IVA_PRODOTTO_AUTORE)
VALUES
('0043', 'GUITART 101', '19,00', '150', 'MAGAZINE', 'FISICO', '17-APR-
2018', '39999320052', '39999320052');

INSERT INTO PRODOTTO
(ID_PRODOTTO, NOME_PRODOTTO, PREZZO_PRODOTTO, QUANTITA_DISPONIBILE_PRODOTTO,
TIPOLOGIA_PRODOTTO, TIPO_PRODOTTO, DATA_PUBBLICAZIONE,
PARTITA_IVA_PRODOTTO_SOCIETA, PARTITA_IVA_PRODOTTO_AUTORE)
VALUES
('0015', 'LEO BROUWER', '29,90', '150', 'MONOGRAFIA', 'FISICO', '8-AGO-
2018', '45350740630', '72650340796');

INSERT INTO PRODOTTO
(ID_PRODOTTO, NOME_PRODOTTO, PREZZO_PRODOTTO, QUANTITA_DISPONIBILE_PRODOTTO,
TIPOLOGIA_PRODOTTO, TIPO_PRODOTTO, DATA_PUBBLICAZIONE,
PARTITA_IVA_PRODOTTO_SOCIETA, PARTITA_IVA_PRODOTTO_AUTORE)
VALUES
('0009', 'FERDINANDO CARULLI', '29,90', '150', 'MONOGRAFIA', 'FISICO', '22-AGO-
2018', '18977850363', '58128990544');

INSERT INTO PRODOTTO
(ID_PRODOTTO, NOME_PRODOTTO, PREZZO_PRODOTTO, QUANTITA_DISPONIBILE_PRODOTTO,
TIPOLOGIA_PRODOTTO, TIPO_PRODOTTO, DATA_PUBBLICAZIONE,
PARTITA_IVA_PRODOTTO_SOCIETA, PARTITA_IVA_PRODOTTO_AUTORE)
VALUES
('0069', '66 NOTTURNI', '0', '150', 'E-BOOK', 'DIGITALE', '18-GIU-
2019', '34789260295', '63798360192');

```

```
INSERT INTO PRODOTTO
(ID_PRODOTTO, NOME_PRODOTTO, PREZZO_PRODOTTO,
QUANTITA_DISPONIBILE_PRODOTTO, TIPOLOGIA_PRODOTTO, TIPO_PRODOTTO,
DATA_PUBBLICAZIONE, PARTITA_IVA_PRODOTTO_SOCIETA,
PARTITA_IVA_PRODOTTO_AUTORE)
VALUES
('0059','DUE STUDI','0','150','E-BOOK','DIGITALE','3-AGO-
2020','28146030045','23589960691');

INSERT INTO PRODOTTO
(ID_PRODOTTO, NOME_PRODOTTO, PREZZO_PRODOTTO,
QUANTITA_DISPONIBILE_PRODOTTO, TIPOLOGIA_PRODOTTO, TIPO_PRODOTTO,
DATA_PUBBLICAZIONE, PARTITA_IVA_PRODOTTO_SOCIETA,
PARTITA_IVA_PRODOTTO_AUTORE)
VALUES
('0014','EMILIO PUJOL','29,90','150','MONOGRAFIA','FISICO','17-SET-
2021','64905180770','51619000519');

INSERT INTO PRODOTTO
(ID_PRODOTTO, NOME_PRODOTTO, PREZZO_PRODOTTO,
QUANTITA_DISPONIBILE_PRODOTTO, TIPOLOGIA_PRODOTTO, TIPO_PRODOTTO,
DATA_PUBBLICAZIONE, PARTITA_IVA_PRODOTTO_SOCIETA,
PARTITA_IVA_PRODOTTO_AUTORE)
VALUES
('0045','A WINTER DAY','0','150','E-BOOK','DIGITALE','2-GIU-
2022','28146030045','23589960691');
```

## METODO\_DI\_PAGAMENTO

```

INSERT INTO METODO_DI_PAGAMENTO
(CODICE_IDENTIFICATIVO, CCV, DATA_SCADENZA, TIPO_CARTA, TIPO_CIRCUITO,
CODICE_FISCALE_MDP)
VALUES
('4830636800349531', '245', '08-2026', 'CARTA DI
CREDITO', 'VISA', 'SGSGNT80A01E654Y');

INSERT INTO METODO_DI_PAGAMENTO
(CODICE_IDENTIFICATIVO, CCV, DATA_SCADENZA, TIPO_CARTA, TIPO_CIRCUITO,
CODICE_FISCALE_MDP)
VALUES
('5268176090182993', '107', '03-2025', 'CARTA
PREPAGATA', 'MASTERCARD', 'MNFPLS92S04A892C');

INSERT INTO METODO_DI_PAGAMENTO
(CODICE_IDENTIFICATIVO, CCV, DATA_SCADENZA, TIPO_CARTA, TIPO_CIRCUITO,
CODICE_FISCALE_MDP)
VALUES
('4830639851337660', '557', '08-2029', 'CARTA
PREPAGATA', 'VISA', 'RSSDBR87D11D612M');

INSERT INTO METODO_DI_PAGAMENTO
(CODICE_IDENTIFICATIVO, CCV, DATA_SCADENZA, TIPO_CARTA, TIPO_CIRCUITO,
CODICE_FISCALE_MDP)
VALUES
('5220691605273043', '656', '03-2029', 'CARTA DI
CREDITO', 'MASTERCARD', 'DMNNLT94L31H501X');

INSERT INTO METODO_DI_PAGAMENTO
(CODICE_IDENTIFICATIVO, CCV, DATA_SCADENZA, TIPO_CARTA, TIPO_CIRCUITO,
CODICE_FISCALE_MDP)
VALUES
('378009218690464', '5781', '07-2024', 'CARTA DI CREDITO', 'AMERICAN
EXPRESS', 'DLCRNT83S06D810W');

INSERT INTO METODO_DI_PAGAMENTO
(CODICE_IDENTIFICATIVO, CCV, DATA_SCADENZA, TIPO_CARTA, TIPO_CIRCUITO,
CODICE_FISCALE_MDP)
VALUES
('4830636927541432', '894', '04-2028', 'CARTA
PREPAGATA', 'VISA', 'TRVMRA62M13D150J');

INSERT INTO METODO_DI_PAGAMENTO
(CODICE_IDENTIFICATIVO, CCV, DATA_SCADENZA, TIPO_CARTA, TIPO_CIRCUITO,
CODICE_FISCALE_MDP)
VALUES
('4830633328759713', '425', '02-2027', 'CARTA DI
CREDITO', 'VISA', 'CLBRNT82S70F205G');

INSERT INTO METODO_DI_PAGAMENTO
(CODICE_IDENTIFICATIVO, CCV, DATA_SCADENZA, TIPO_CARTA, TIPO_CIRCUITO,
CODICE_FISCALE_MDP)
VALUES
('378005624560374', '8585', '11-2025', 'CARTA DI CREDITO', 'AMERICAN
EXPRESS', 'PLRRSR41B41E517N');

INSERT INTO METODO_DI_PAGAMENTO
(CODICE_IDENTIFICATIVO, CCV, DATA_SCADENZA, TIPO_CARTA, TIPO_CIRCUITO,
CODICE_FISCALE_MDP)
VALUES
('4830639629177182', '466', '01-2026', 'CARTA
PREPAGATA', 'VISA', 'FRNLFA83T14A390H');

INSERT INTO METODO_DI_PAGAMENTO
(CODICE_IDENTIFICATIVO, CCV, DATA_SCADENZA, TIPO_CARTA, TIPO_CIRCUITO,
CODICE_FISCALE_MDP)
VALUES
('5101857379719743', '383', '10-2029', 'CARTA DI
CREDITO', 'MASTERCARD', 'LMBLNS92H07F8390');

```

## PROMOZIONE

```
INSERT INTO PROMOZIONE
(CODICE_PROMOZIONE, NOME_PROMOZIONE, VALORE_PROMOZIONE,
DATA_INIZIO_PROMOZIONE, DATA_FINE_PROMOZIONE)
VALUES
('HS241X', 'PROM01', '5', '27-GEN-2023', '14-APR-2023');

INSERT INTO PROMOZIONE
(CODICE_PROMOZIONE, NOME_PROMOZIONE, VALORE_PROMOZIONE,
DATA_INIZIO_PROMOZIONE, DATA_FINE_PROMOZIONE)
VALUES
('HS242X', 'PROM02', '10', '05-MAG-2023', '08-AGO-2023');

INSERT INTO PROMOZIONE
(CODICE_PROMOZIONE, NOME_PROMOZIONE, VALORE_PROMOZIONE,
DATA_INIZIO_PROMOZIONE, DATA_FINE_PROMOZIONE)
VALUES
('HS243X', 'PROM03', '15', '20-SET-2023', '04-OTT-2023');

INSERT INTO PROMOZIONE
(CODICE_PROMOZIONE, NOME_PROMOZIONE, VALORE_PROMOZIONE,
DATA_INIZIO_PROMOZIONE, DATA_FINE_PROMOZIONE)
VALUES
('HS244X', 'PROM04', '20', '18-OTT-2023', '06-NOV-2023');

INSERT INTO PROMOZIONE
(CODICE_PROMOZIONE, NOME_PROMOZIONE, VALORE_PROMOZIONE,
DATA_INIZIO_PROMOZIONE, DATA_FINE_PROMOZIONE)
VALUES
('HS245X', 'PROM05', '25', '18-DIC-2023', '28-DIC-2023');
```

## ACQUISTO

```
INSERT INTO ACQUISTO
(N_ACQUISTO, CODICE_FISCALE_ACQUISTO, CODICE_IDENTIFICATIVO_ACQUISTO,
CODICE_PROMOZIONE_ACQUISTO)
VALUES
('1001','SGSGNT80A01E654Y','4830636800349531','');
INSERT INTO ACQUISTO
(N_ACQUISTO, CODICE_FISCALE_ACQUISTO, CODICE_IDENTIFICATIVO_ACQUISTO,
CODICE_PROMOZIONE_ACQUISTO)
VALUES
('1002','MNFPLS92S04A892C','5268176090182993','');
INSERT INTO ACQUISTO
(N_ACQUISTO, CODICE_FISCALE_ACQUISTO, CODICE_IDENTIFICATIVO_ACQUISTO,
CODICE_PROMOZIONE_ACQUISTO)
VALUES
('1003','RSSDBR87D11D612M','4830639851337660','');
INSERT INTO ACQUISTO
(N_ACQUISTO, CODICE_FISCALE_ACQUISTO, CODICE_IDENTIFICATIVO_ACQUISTO,
CODICE_PROMOZIONE_ACQUISTO)
VALUES
('1004','DLCRNT83S06D810W','378009218690464','');
INSERT INTO ACQUISTO
(N_ACQUISTO, CODICE_FISCALE_ACQUISTO, CODICE_IDENTIFICATIVO_ACQUISTO,
CODICE_PROMOZIONE_ACQUISTO)
VALUES
('1005','CLBRNT82S70F205G','4830633328759713','');
INSERT INTO ACQUISTO
(N_ACQUISTO, CODICE_FISCALE_ACQUISTO, CODICE_IDENTIFICATIVO_ACQUISTO,
CODICE_PROMOZIONE_ACQUISTO)
VALUES
('1006','FRNLFA83T14A390H','4830639629177182','');
INSERT INTO ACQUISTO
(N_ACQUISTO, CODICE_FISCALE_ACQUISTO, CODICE_IDENTIFICATIVO_ACQUISTO,
CODICE_PROMOZIONE_ACQUISTO)
VALUES
('1007','LMBLNS92H07F8390','5101857379719743','');
INSERT INTO ACQUISTO
(N_ACQUISTO, CODICE_FISCALE_ACQUISTO, CODICE_IDENTIFICATIVO_ACQUISTO,
CODICE_PROMOZIONE_ACQUISTO)
VALUES
('1008','DMNNLT94L31H501X','5220691605273043','');
INSERT INTO ACQUISTO
(N_ACQUISTO, CODICE_FISCALE_ACQUISTO, CODICE_IDENTIFICATIVO_ACQUISTO,
CODICE_PROMOZIONE_ACQUISTO)
VALUES
('1009','TRVMRA62M13D150J','4830636927541432','HS241X');
INSERT INTO ACQUISTO
(N_ACQUISTO, CODICE_FISCALE_ACQUISTO, CODICE_IDENTIFICATIVO_ACQUISTO,
CODICE_PROMOZIONE_ACQUISTO)
VALUES
('1010','PLRRSR41B41E517N','378005624560374','HS242X');
```

## INDIRIZZO

```
INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('11','VIA','CASTELFIDARDO','120','CROPALATI','COSENZA','87060','IT','SGSGN
T80A01E654Y');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('12','VIA','TORINO','27','PASSO
PENICE','PIACENZA','29020','IT','SGSGNT80A01E654Y');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('13','VIA','PARTENOPE','23','CAIRA','FROSINONE','03040','IT','SGSGNT80A01E
654Y');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('14','VIA','DONNALBINA','144','VAPRIO
D''AGOGNA','NOVARA','28010','IT','SGSGNT80A01E654Y');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('15','VIA','VOLTO SAN
LUCA','133','MEUGLIANO','TORINO','10080','IT','SGSGNT80A01E654Y');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('21','VIA','DEL VIMINALE','138','SAN
LEONE','CAGLIARI','09012','IT','MNFPLS92S04A892C');
```

```
INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('61', 'CORSO', 'NOVARA', '148', 'OROSEI', 'NUORO', '08028', 'IT', 'FRNLFA83T14A39
0H');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('62', 'VIA', 'TORRICELLI', '69', 'LEVICO
TERME', 'TRENTO', '38056', 'IT', 'FRNLFA83T14A390H');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('71', 'VIA', 'LEOPARDI', '70', 'FRACISCO', 'SONDRIO', '23021', 'IT', 'LMBLNS92H0
7F8390');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('81', 'VIA', 'NAZARIO
SAURO', '128', 'LISSONE', 'MILANO', '20035', 'IT', 'DMNNLT94L31H501X');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('91', 'VIA', 'DELLE MURA
GIANICOLENSI', '53', 'PIETRACUPA', 'CAMPOBASSO', '86020', 'IT', 'TRVMRA62M13D150
J');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('101', 'VIA', 'CATULLO', '56', 'SAN BENEDETTO DEI
MARTI', 'L'AQUILA', '67058', 'IT', 'PLRRSR41B41E517N');
```

```
INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA,
PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('31','VIColo','CALCIRELLI','111','SAN VITO LO
CAPO','TRAPANI','91010','IT','RSSDBR87D11D612M');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA,
PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('32','VIA','ARCHIMEDE','144','GRANAGLIONE','BOLOGNA','40045',
'IT','RSSDBR8 7D11D612M');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA,
PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('41','STRADA PROVINCIALE','65','56','CASIRATE
D' ADDA','BERGAMO','24040','IT','DLCRNT83S06D810W');

INSERT INTO INDIRIZZO
(ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO, CITTA,
PROVINCIA, CAP,
STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
VALUES
('51','VIA','RAFFAELE
CONFORTI','105','CAMPOTOSTO','L'AQUILA','67013','IT','CLBRNT8
2S70F205G');
```

## STATO\_ACQUISTO

La tabella viene popolata dopo ACQUISTO in quanto è debole rispetto alla seconda.

```
INSERT INTO STATO_ACQUISTO  
(DATA_ACQUISTO, STATO_ACQUISTO, N_ACQUISTO_STATO_ACQUISTO)  
VALUES  
('11-SET-2019', 'CONFERMATO', '1001');  
  
INSERT INTO STATO_ACQUISTO  
(DATA_ACQUISTO, STATO_ACQUISTO, N_ACQUISTO_STATO_ACQUISTO)  
VALUES  
('26-SET-2019', 'RESPINTO', '1002');  
  
INSERT INTO STATO_ACQUISTO  
(DATA_ACQUISTO, STATO_ACQUISTO, N_ACQUISTO_STATO_ACQUISTO)  
VALUES  
('7-GEN-2019', 'CONFERMATO', '1003');  
  
INSERT INTO STATO_ACQUISTO  
(DATA_ACQUISTO, STATO_ACQUISTO, N_ACQUISTO_STATO_ACQUISTO)  
VALUES  
('28-MAR-2020', 'CONFERMATO', '1004');  
  
INSERT INTO STATO_ACQUISTO  
(DATA_ACQUISTO, STATO_ACQUISTO, N_ACQUISTO_STATO_ACQUISTO)  
VALUES  
('22-MAR-2021', 'RESPINTO', '1005');  
  
INSERT INTO STATO_ACQUISTO  
(DATA_ACQUISTO, STATO_ACQUISTO, N_ACQUISTO_STATO_ACQUISTO)  
VALUES  
('10-OTT-2021', 'CONFERMATO', '1006');  
  
INSERT INTO STATO_ACQUISTO  
(DATA_ACQUISTO, STATO_ACQUISTO, N_ACQUISTO_STATO_ACQUISTO)  
VALUES  
('12-NOV-2021', 'CONFERMATO', '1007');  
  
INSERT INTO STATO_ACQUISTO  
(DATA_ACQUISTO, STATO_ACQUISTO, N_ACQUISTO_STATO_ACQUISTO)  
VALUES  
('18-GEN-2022', 'IN ELABORAZIONE', '1008');  
  
INSERT INTO STATO_ACQUISTO  
(DATA_ACQUISTO, STATO_ACQUISTO, N_ACQUISTO_STATO_ACQUISTO)  
VALUES  
('31-GEN-2023', 'IN ELABORAZIONE', '1009');  
  
INSERT INTO STATO_ACQUISTO  
(DATA_ACQUISTO, STATO_ACQUISTO, N_ACQUISTO_STATO_ACQUISTO)  
VALUES  
('23-GIU-2023', 'CONFERMATO', '1010');
```

## SPEDIZIONE

```
INSERT INTO SPEDIZIONE
(N_SPEDIZIONE, DATA_CREAZIONE_SPEDIZIONE, INDIRIZZO_SPEDIZIONE,
N_ACQUISTO_SPEDIZIONE)
VALUES
('0001','11-SET-2019','12','1001');

INSERT INTO SPEDIZIONE
(N_SPEDIZIONE, DATA_CREAZIONE_SPEDIZIONE, INDIRIZZO_SPEDIZIONE,
N_ACQUISTO_SPEDIZIONE)
VALUES
('0002','26-SET-2019','21','1002');

INSERT INTO SPEDIZIONE
(N_SPEDIZIONE, DATA_CREAZIONE_SPEDIZIONE, INDIRIZZO_SPEDIZIONE,
N_ACQUISTO_SPEDIZIONE)
VALUES
('0003','7-GEN-2019','31','1003');

INSERT INTO SPEDIZIONE
(N_SPEDIZIONE, DATA_CREAZIONE_SPEDIZIONE, INDIRIZZO_SPEDIZIONE,
N_ACQUISTO_SPEDIZIONE)
VALUES
('0004','28-MAR-2020','41','1004');

INSERT INTO SPEDIZIONE
(N_SPEDIZIONE, DATA_CREAZIONE_SPEDIZIONE, INDIRIZZO_SPEDIZIONE,
N_ACQUISTO_SPEDIZIONE)
VALUES
('0005','22-MAR-2021','51','1005');

INSERT INTO SPEDIZIONE
(N_SPEDIZIONE, DATA_CREAZIONE_SPEDIZIONE, INDIRIZZO_SPEDIZIONE,
N_ACQUISTO_SPEDIZIONE)
VALUES
('0006','10-OTT-2021','62','1006');

INSERT INTO SPEDIZIONE
(N_SPEDIZIONE, DATA_CREAZIONE_SPEDIZIONE, INDIRIZZO_SPEDIZIONE,
N_ACQUISTO_SPEDIZIONE)
VALUES
('0007','12-NOV-2021','71','1007');

INSERT INTO SPEDIZIONE
(N_SPEDIZIONE, DATA_CREAZIONE_SPEDIZIONE, INDIRIZZO_SPEDIZIONE,
N_ACQUISTO_SPEDIZIONE)
VALUES
('0008','18-GEN-2022','81','1008');

INSERT INTO SPEDIZIONE
(N_SPEDIZIONE, DATA_CREAZIONE_SPEDIZIONE, INDIRIZZO_SPEDIZIONE,
N_ACQUISTO_SPEDIZIONE)
VALUES
('0009','31-GEN-2022','91','1009');

INSERT INTO SPEDIZIONE
(N_SPEDIZIONE, DATA_CREAZIONE_SPEDIZIONE, INDIRIZZO_SPEDIZIONE,
N_ACQUISTO_SPEDIZIONE)
VALUES
('0010','23-GIU-2023','101','1010');
```

## STATO\_SPEDIZIONE

La tabella viene popolata dopo SPEDIZIONE in quanto è debole rispetto alla seconda.

```
INSERT INTO STATO_SPEDIZIONE
(DATA_STATO_SPEDIZIONE, STATO_SPEDIZIONE,
N_SPEDIZIONE_STATO_SPEDIZIONE)
VALUES
('11-SET-2019','IN ELABORAZIONE','0001');
INSERT INTO STATO_SPEDIZIONE
(DATA_STATO_SPEDIZIONE, STATO_SPEDIZIONE,
N_SPEDIZIONE_STATO_SPEDIZIONE)
VALUES
('26-SET-2019','RESPINTO','0002');
INSERT INTO STATO_SPEDIZIONE
(DATA_STATO_SPEDIZIONE, STATO_SPEDIZIONE,
N_SPEDIZIONE_STATO_SPEDIZIONE)
VALUES
('7-GEN-2019','PRONTO PER LA SPEDIZIONE','0003');
INSERT INTO STATO_SPEDIZIONE
(DATA_STATO_SPEDIZIONE, STATO_SPEDIZIONE,
N_SPEDIZIONE_STATO_SPEDIZIONE)
VALUES
('28-MAR-2020','CONSEGNATO','0004');
INSERT INTO STATO_SPEDIZIONE
(DATA_STATO_SPEDIZIONE, STATO_SPEDIZIONE,
N_SPEDIZIONE_STATO_SPEDIZIONE)
VALUES
('22-MAR-2021','RESPINTO','0005');
INSERT INTO STATO_SPEDIZIONE
(DATA_STATO_SPEDIZIONE, STATO_SPEDIZIONE,
N_SPEDIZIONE_STATO_SPEDIZIONE)
VALUES
('10-OTT-2021','SPEDITO','0006');
INSERT INTO STATO_SPEDIZIONE
(DATA_STATO_SPEDIZIONE, STATO_SPEDIZIONE,
N_SPEDIZIONE_STATO_SPEDIZIONE)
VALUES
('12-NOV-2021','IN CONSEGNA','0007');
INSERT INTO STATO_SPEDIZIONE
(DATA_STATO_SPEDIZIONE, STATO_SPEDIZIONE,
N_SPEDIZIONE_STATO_SPEDIZIONE)
VALUES
('18-GEN-2022','IN ELABORAZIONE','0008');
INSERT INTO STATO_SPEDIZIONE
(DATA_STATO_SPEDIZIONE, STATO_SPEDIZIONE,
N_SPEDIZIONE_STATO_SPEDIZIONE)
VALUES
('31-GEN-2022','IN ELABORAZIONE','0009');
INSERT INTO STATO_SPEDIZIONE
(DATA_STATO_SPEDIZIONE, STATO_SPEDIZIONE,
N_SPEDIZIONE_STATO_SPEDIZIONE)
VALUES
('23-GIU-2023','RESTITUITO','0010');
```

## CONTIENE

La tabella viene popolata dopo ACQUISTO e PRODOTTO in quanto è una tabella di transizione.

```
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('2','1001','0061');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1005','0061');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1001','0015');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1003','0015');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1006','0015');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1002','0043');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1005','0043');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1010','0043');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1003','0009');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1009','0009');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1003','0059');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1006','0059');
```

```
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1010','0059');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1008','0045');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1010','0045');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1004','0042');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1006','0042');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1007','0042');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1009','0014');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1007','0014');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1005','0039');
INSERT INTO CONTIENE
(QUANTITA, N_ACQUISTO_CONTIENE, ID_PRODOTTO_CONTIENE)
VALUES
('1','1010','0039');
```

## COSTITUITO

La tabella viene popolata dopo ABBONAMENTO e PRODOTTO in quanto è una tabella di transizione.

```
INSERT INTO COSTITUITO
(CODICE_ABBONAMENTO_COSTITUITO, ID_PRODOTTO_COSTITUITO)
VALUES
('ABB.CLS','0061');
INSERT INTO COSTITUITO
(CODICE_ABBONAMENTO_COSTITUITO, ID_PRODOTTO_COSTITUITO)
VALUES
('ABB.CLS','0043');
INSERT INTO COSTITUITO
(CODICE_ABBONAMENTO_COSTITUITO, ID_PRODOTTO_COSTITUITO)
VALUES
('ABB.CLS','0015');
INSERT INTO COSTITUITO
(CODICE_ABBONAMENTO_COSTITUITO, ID_PRODOTTO_COSTITUITO)
VALUES
('ABB.DLX','0061');
INSERT INTO COSTITUITO
(CODICE_ABBONAMENTO_COSTITUITO, ID_PRODOTTO_COSTITUITO)
VALUES
('ABB.DLX','0043');
INSERT INTO COSTITUITO
(CODICE_ABBONAMENTO_COSTITUITO, ID_PRODOTTO_COSTITUITO)
VALUES
('ABB.DLX','0015');
INSERT INTO COSTITUITO
(CODICE_ABBONAMENTO_COSTITUITO, ID_PRODOTTO_COSTITUITO)
VALUES
('ABB.DLX','0009');
INSERT INTO COSTITUITO
(CODICE_ABBONAMENTO_COSTITUITO, ID_PRODOTTO_COSTITUITO)
VALUES
('ABB.DLX','0014');
```

## POSSIEDE

La tabella viene popolata dopo ABBONAMENTO e CLIENTE in quanto è una tabella di transizione.

```
INSERT INTO POSSIEDE
(CODICE_ABBONAMENTO_POSSIEDE, CODICE_FISCALE_POSSIEDE)
VALUES
('ABB.CLS','DLCRNT83S06D810W');

INSERT INTO POSSIEDE
(CODICE_ABBONAMENTO_POSSIEDE, CODICE_FISCALE_POSSIEDE)
VALUES
('ABB.DLX','CLBRNT82S70F205G');

INSERT INTO POSSIEDE
(CODICE_ABBONAMENTO_POSSIEDE, CODICE_FISCALE_POSSIEDE)
VALUES
('ABB.DLX','PLRRSR41B41E517N');

INSERT INTO POSSIEDE
(CODICE_ABBONAMENTO_POSSIEDE, CODICE_FISCALE_POSSIEDE)
VALUES
('ABB.CLS','FRNLFA83T14A390H');

INSERT INTO POSSIEDE
(CODICE_ABBONAMENTO_POSSIEDE, CODICE_FISCALE_POSSIEDE)
VALUES
('ABB.CLS','LMBLNS92H07F8390');
```

## 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 INDIRIZZI

Questo trigger controlla che ogni volta che si aggiunge un indirizzo, questo non superi il limite massimo. Quando si supera il limite di 5 indirizzi, l'inserimento viene annullato e viene restituito un messaggio di errore.

```
CREATE OR REPLACE TRIGGER MAX_INDIRIZZI
BEFORE INSERT ON INDIRIZZO
FOR EACH ROW
DECLARE
  I_COUNT NUMBER :=0;
  I_ERROR EXCEPTION;
BEGIN
  SELECT COUNT (*) INTO I_COUNT FROM INDIRIZZO WHERE
  (:NEW.CODICE_FISCALE_INDIRIZZO=CODICE_FISCALE_INDIRIZZO);
  IF I_COUNT = 5
  THEN RAISE I_ERROR;
  END IF;

EXCEPTION
  WHEN I_ERROR THEN RAISE_APPLICATION_ERROR(-20010,'RAGGIUNTO IL NUMERO
  MASSIMO DI INDIRIZZI');
END;
```

## NUMERO MASSIMO DI TIPO DI PRODOTTO PER ACQUISTO

Questo trigger controlla che ogni volta che si aggiunge la quantità di un singolo prodotto che si vuole acquistare, questo non superi il limite massimo. Quando si supera il limite di 10, l'inserimento viene annullato e viene restituito un messaggio di errore.

```
CREATE OR REPLACE TRIGGER MAX_PRODOTTO
BEFORE INSERT ON CONTIENE
FOR EACH ROW
DECLARE
C_QUANTITA NUMBER :=0;
C_ERROR EXCEPTION;
BEGIN
FOR R IN
  (SELECT QUANTITA INTO C_QUANTITA FROM CONTIENE WHERE
N_ACQUISTO_CONTIENE = :NEW.N_ACQUISTO_CONTIENE)
  LOOP
  IF :NEW.QUANTITA >= 10
  THEN RAISE C_ERROR;
  END IF;
  END LOOP;
EXCEPTION
WHEN C_ERROR THEN RAISE_APPLICATION_ERROR(-20011,'RAGGIUNTO IL NUMERO
MASSIMO DI TIPO DI PRODOTTO CHE E POSSIBILE ACQUISTARE');
END;
```

## CONTROLLO IN CASO DI SCORTE NON DISPONIBILI

Questo trigger controlla che ci siano abbastanza scorte ogni volta che si aggiunge la quantità di un prodotto che si vuole acquistare. Se non ci sono abbastanza scorte, all'inserimento della quantità, restituisce un errore.

```
CREATE OR REPLACE TRIGGER QUANTITA_NON_DISPONIBILE
BEFORE INSERT ON CONTIENE
FOR EACH ROW
DECLARE
C_QUANTITA NUMBER :=0;
C_ERROR EXCEPTION;
BEGIN
  SELECT QUANTITA_DISPONIBILE_PRODOTTO INTO C_QUANTITA FROM PRODOTTO
WHERE ID_PRODOTTO = :NEW.ID_PRODOTTO_CONTIENE;
  IF :NEW.QUANTITA > C_QUANTITA
  THEN RAISE C_ERROR;
  END IF;

EXCEPTION
WHEN C_ERROR THEN RAISE_APPLICATION_ERROR(-20012,'QUANTITA NON
DISPONIBILE');
END;
```

## CONTROLLO LA DATA DI SPEDIZIONE È POSTERIORE ALL'ACQUISTO

Questo trigger controlla che la data di spedizione sia posteriore a quella di acquisto. Nel caso in cui non lo sia, viene restituito un errore.

```
CREATE OR REPLACE TRIGGER DATA_ERRATA_SPEDIZIONE
BEFORE INSERT ON SPEDIZIONE
FOR EACH ROW
DECLARE
A_DATA_ACQUISTO DATE;
DATA_ERRATA EXCEPTION;

BEGIN
SELECT DATA_ACQUISTO INTO A_DATA_ACQUISTO FROM STATO_ACQUISTO WHERE
N_ACQUISTO_STATO_ACQUISTO = :NEW.N_ACQUISTO_SPEDIZIONE;
IF (:NEW.DATA_CREAZIONE_SPEDIZIONE - A_DATA_ACQUISTO)<0 THEN
RAISE DATA_ERRATA;
END IF;

EXCEPTION
WHEN DATA_ERRATA THEN RAISE_APPLICATION_ERROR(-20013, 'DATA SPEDIZIONE
NON VALIDA');
END;
```

## NUMERO MASSIMO DI METODI DI PAGAMENTO

Questo trigger controlla che ogni volta che si aggiunge un metodo di pagamento, questo non superi il limite massimo. Quando si supera il limite di 5, l'inserimento viene annullato e viene restituito un messaggio di errore.

```
CREATE OR REPLACE TRIGGER MAX_METODO_DI_PAGAMENTO
BEFORE INSERT ON METODO_DI_PAGAMENTO
FOR EACH ROW
DECLARE
I_COUNT NUMBER :=0;
I_ERROR EXCEPTION;
BEGIN
SELECT COUNT (*) INTO I_COUNT FROM METODO_DI_PAGAMENTO WHERE
(:NEW.CODICE_FISCALE_MDP=CODICE_FISCALE_MDP);
IF I_COUNT = 5
THEN RAISE I_ERROR;
END IF;

EXCEPTION
WHEN I_ERROR THEN RAISE_APPLICATION_ERROR(-20014, 'RAGGIUNTO IL NUMERO
MASSIMO DI METODI DI PAGAMENTO');
END;
```

## DIMINUZIONE DELLA QUANTITÀ DI PRODOTTO ALL'ACQUISTO

Questo trigger controlla che le scorte vengano diminuite, ogni volta che si acquista un prodotto. Quando si supera il limite di 10, l'inserimento viene annullato e viene restituito un messaggio di errore. Se l'acquisto non è valido o nel caso di un qualunque altro errore, viene restituito un messaggio di errore.

```
CREATE OR REPLACE TRIGGER AGGIORNAMENTO_QUANTITA
AFTER INSERT ON contiene
FOR EACH ROW
DECLARE
    C_ACQUISTO NUMBER := :NEW.N_ACQUISTO_CONTIENE;
    C_QUANTITA NUMBER := :NEW.QUANTITA;
BEGIN
    UPDATE PRODOTTO
    SET QUANTITA_DISPONIBILE_PRODOTTO = QUANTITA_DISPONIBILE_PRODOTTO
    - C_QUANTITA
    WHERE ID_PRODOTTO = :NEW.ID_PRODOTTO_CONTIENE;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20015, 'Acquisto non valido.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20016, 'Errore durante
l''aggiornamento della quantità.');
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.

## VISUALIZZA VENDITE PRODOTTI

Questa procedura calcola il numero di volte che ciascun prodotto è stato venduto.

```
CREATE OR REPLACE PROCEDURE VENDITE_PRODOTTI
IS
v_count NUMBER;
BEGIN
  FOR R IN (SELECT ID_PRODOTTO, NOME_PRODOTTO FROM PRODOTTO)
  LOOP
    SELECT COUNT(*) INTO v_count
    FROM CONTIENE
    WHERE ID_PRODOTTO_CONTIENE = R.ID_PRODOTTO;
    DBMS_OUTPUT.PUT_LINE('Il prodotto ' || R.NOME_PRODOTTO ||
' è stato venduto ' || v_count || ' volte.');
  END LOOP;
END;
```

## TOTALE ACQUISTO

Questa procedura calcola il totale della spesa per un determinato acquisto effettuato da un cliente specifico.

La procedura ha due parametri di input: CODICE\_FISCALE e N\_ACQUISTO. Questi parametri vengono utilizzati per identificare l'acquisto specifico.

```
CREATE OR REPLACE PROCEDURE TOTALE_ACQUISTO (C_CODICE_FISCALE CHAR,
N_N_ACQUISTO NUMBER)
IS
TOT NUMBER := 0;
TOTALESPESA INTEGER :=0;
BEGIN
SELECT SUM(QUANTITA * PREZZO_PRODOTTO) INTO TOT
FROM CLIENTE
    JOIN ACQUISTO ON CLIENTE.CODICE_FISCALE =
ACQUISTO.CODICE_FISCALE_ACQUISTO
    JOIN CONTIENE ON ACQUISTO.N_ACQUISTO =
CONTIENE.N_ACQUISTO_CONTIENE
    JOIN PRODOTTO ON ID_PRODOTTO_CONTIENE = PRODOTTO.ID_PRODOTTO
WHERE CLIENTE.CODICE_FISCALE=C_CODICE_FISCALE AND
ACQUISTO.N_ACQUISTO=N_N_ACQUISTO;
DBMS_OUTPUT.PUT_LINE('Il totale dell''acquisto ' || N_N_ACQUISTO || ' è: ' || TOT);
END;
```

## INSERIMENTO INDIRIZZO

Questa procedura è utilizzata per inserire un nuovo indirizzo nella tabella INDIRIZZO.  
Prende in input i parametri: ID\_INDIRIZZO, CODICE\_FISCALE\_INDIRIZZO,  
QUALIFICATORE, STRADA, NUMERO, CITTA, PROVINCIA, CAP, STATO\_EUROPEO.

```
CREATE OR REPLACE PROCEDURE INSERIMENTO_INDIRIZZO(
    I_ID_INDIRIZZO NUMBER,
    I_CODICE_FISCALE_INDIRIZZO CHAR,
    I_QUALIFICATORE VARCHAR2,
    I_STRADA VARCHAR2,
    I_NUMERO VARCHAR2,
    I_CITTA VARCHAR2,
    I_PROVINCIA VARCHAR2,
    I_CAP VARCHAR2,
    I_STATO_EUROPEO CHAR)
IS
BEGIN
    INSERT INTO INDIRIZZO (ID_INDIRIZZO, QUALIFICATORE, STRADA, NUMERO,
    CITTA, PROVINCIA, CAP, STATO_EUROPEO, CODICE_FISCALE_INDIRIZZO)
    VALUES (I_ID_INDIRIZZO, I_QUALIFICATORE, I_STRADA, I_NUMERO, I_CITTA,
    I_PROVINCIA, I_CAP, I_STATO_EUROPEO, I_CODICE_FISCALE_INDIRIZZO);
    COMMIT;
END;
```

## CANCELLAZIONE ACQUISTO

Questa procedura è utilizzata per cancellare un acquisto. Prende in input i parametri: CODICE\_FISCALE, N\_ACQUISTO, N\_SPEDIZIONE.

```
CREATE OR REPLACE PROCEDURE CANCELLAZIONE_ACQUISTO(
C_CODICE_FISCALE CHAR,
N_N_ACQUISTO NUMBER,
N_N_SPEDIZIONE NUMBER)
IS
NUM_ACQ    NUMBER;
BEGIN
FOR R IN(
SELECT N_ACQUISTO    INTO  NUM_ACQ
      FROM ACQUISTO JOIN SPEDIZIONE ON ACQUISTO.N_ACQUISTO =
SPEDIZIONE.N_ACQUISTO_SPEDIZIONE
      WHERE ACQUISTO.CODICE_FISCALE_ACQUISTO = C_CODICE_FISCALE
      AND ACQUISTO.N_ACQUISTO = N_N_ACQUISTO
      AND SPEDIZIONE.N_SPEDIZIONE = N_N_SPEDIZIONE)
LOOP
UPDATE STATO_ACQUISTO
SET STATO_ACQUISTO = 'CANCELLATO'
WHERE N_ACQUISTO_STATO_ACQUISTO = N_N_ACQUISTO;

UPDATE STATO_SPEDIZIONE
SET STATO_SPEDIZIONE = 'CANCELLATO'
WHERE N_SPEDIZIONE_STATO_SPEDIZIONE = N_N_SPEDIZIONE
      AND STATO_SPEDIZIONE='IN ELABORAZIONE'
      OR STATO_SPEDIZIONE='IN ATTESA'
      OR STATO_SPEDIZIONE='PRONTO PER LA SPEDIZIONE';
END LOOP;
END;
```

## 2.6 VISTE

Le viste rappresentano una rappresentazione virtuale dei dati presenti in una o più tabelle. Le viste consentono di creare una visualizzazione personalizzata dei dati che soddisfa le esigenze di un'applicazione o di un utente (dati meno/più recenti, calcolo di attributi derivati).

Questa vista permette ai clienti e ai dipendenti di vedere informazioni riguardante un acquisto specifico.

```
CREATE OR REPLACE VIEW VIEW_ORDINI AS
SELECT
    CLIENTE.CODICE_FISCALE,
    ACQUISTO.N_ACQUISTO,
    PRODOTTO.ID_PRODOTTO,
    PRODOTTO.NOME_PRODOTTO,
    PRODOTTO.PREZZO_PRODOTTO,
    CONTIENE.QUANTITA,
    METODO_DI_PAGAMENTO.CODICE_IDENTIFICATIVO,
    INDIRIZZO.QUALIFICATORE,
    INDIRIZZO.STRADA,
    INDIRIZZO.NUMERO,
    INDIRIZZO.CITTA,
    INDIRIZZO.PROVINCIA,
    INDIRIZZO.CAP,
    INDIRIZZO.STATO_EUROPEO,
    SPEDIZIONE.N_SPEDIZIONE,
    STATO_SPEDIZIONE.DATA_STATO_SPEDIZIONE,
    STATO_SPEDIZIONE.STATO_SPEDIZIONE
FROM
    CLIENTE
    JOIN METODO_DI_PAGAMENTO ON CLIENTE.CODICE_FISCALE =
        METODO_DI_PAGAMENTO.CODICE_FISCALE_MDP
    JOIN ACQUISTO ON CLIENTE.CODICE_FISCALE =
        ACQUISTO.CODICE_FISCALE_ACQUISTO
    JOIN CONTIENE ON ACQUISTO.N_ACQUISTO = CONTIENE.N_ACQUISTO_CONTIENE
    JOIN SPEDIZIONE ON ACQUISTO.N_ACQUISTO =
        SPEDIZIONE.N_ACQUISTO_SPEDIZIONE
    JOIN INDIRIZZO ON SPEDIZIONE.INDIRIZZO_SPEDIZIONE =
        INDIRIZZO.ID_INDIRIZZO
    JOIN STATO_SPEDIZIONE ON SPEDIZIONE.N_SPEDIZIONE =
        STATO_SPEDIZIONE.N_SPEDIZIONE_STATO_SPEDIZIONE
    JOIN PRODOTTO ON CONTIENE.ID_PRODOTTO_CONTIENE =
        PRODOTTO.ID_PRODOTTO;
```

## 2.7 SCHEDULER

Lo scheduler gestisce l'esecuzione dei job all'interno del database o di azioni periodiche che avvengono in determinati istanti di tempo (una volta al giorno, una volta alla settimana o in orari specifici).

Controlla all'inizio d'ogni mese che tutti gli abbonamenti di un cliente siano scaduti. se lo sono cambia il valore di abbonamento nella cartella cliente da 1 a 0.

```
BEGIN
    DBMS_SCHEDULER.CREATE_JOB(
        job_name      => 'SCADENZA_ABBONAMENTO',
        job_type      => 'PLSQL_BLOCK',
        job_action    => 'BEGIN
                            FOR R IN (SELECT DISTINCT CLIENTE.CODICE_FISCALE
                                       FROM CLIENTE
                                       JOIN POSSIEDE ON CLIENTE.CODICE_FISCALE
= POSSIEDE.CODICE_FISCALE_POSSIEDE
                                       JOIN ABBONAMENTO ON
POSSIEDE.CODICE_ABBONAMENTO_POSSIEDE = ABBONAMENTO.CODICE_ABBONAMENTO
                                       WHERE ABBONAMENTO.DATA_FINE_ABBONAMENTO
<= SYSDATE)
                            LOOP
                                DECLARE
                                    abbonamenti_attivi NUMBER;
                                BEGIN
                                    SELECT COUNT(*) INTO abbonamenti_attivi
                                    FROM POSSIEDE
                                    JOIN ABBONAMENTO ON
POSSIEDE.CODICE_ABBONAMENTO_POSSIEDE = ABBONAMENTO.CODICE_ABBONAMENTO
                                    WHERE POSSIEDE.CODICE_FISCALE_POSSIEDE =
R.CODICE_FISCALE
                                            AND ABBONAMENTO.DATA_FINE_ABBONAMENTO >
SYSDATE;

                                    IF abbonamenti_attivi = 0 THEN
                                        UPDATE CLIENTE
                                        SET ABBONAMENTO = 0
                                        WHERE CODICE_FISCALE = R.CODICE_FISCALE;
                                    END IF;
                                END;
                            END LOOP;
                        END;',
        start_date    => SYSTIMESTAMP,
        repeat_interval => 'FREQ=MONTHLY; BYHOUR=0; BYMINUTE=0; BYSECOND=0',
        enabled       => TRUE);
END;
```

# THANKS

Leonardo and Nicola DB

[www.guitart.it](http://www.guitart.it)

