Università degli studi di Napoli Parthenope

Corso di Basi di Dati e Laboratorio

AlimentariParthenope

Progettazione

Sintesi dei requisiti

Un innovativo e-commerce italiano specializzato in box alimentari ha deciso di ottimizzare la propria esperienza utente sviluppando un database. Questo sistema è stato progettato per tenere traccia e organizzare in modo efficiente le composizioni dei box, la lista degli ingredienti disponibili e le preferenze dei clienti. Grazie a questa soluzione, il processo di acquisto diventa più semplice e immediato. I clienti possono esplorare facilmente le opzioni disponibili, visualizzare dettagli sui valori nutrizionali e sugli allergeni, e selezionare la box più adatta alle proprie esigenze con pochi clic. Il database consente anche di personalizzare l'offerta, permettendo all'e-commerce di proporre ricette su misura, ideali per chi segue una dieta specifica o pratica sport. Inoltre, la piattaforma garantisce una gestione fluida degli ordini e delle spedizioni, assicurando tempi rapidi e ingredienti sempre freschi, consegnati direttamente a casa in tutto il territorio italiano. Questo approccio tecnologico non solo migliora l'organizzazione interna dell'e-commerce, ma anche l'esperienza complessiva dei clienti, che possono contare su un servizio pratico, veloce e in linea con le loro esigenze nutrizionali e di stile di vita.

- **Registrazione obbligatoria**: Per acquistare le box dal sito web, i clienti devono essere necessariamente registrati.
- **Identificazione dei clienti**: Il database identifica i clienti tramite il codice fiscale, in quanto il servizio opera esclusivamente sul territorio italiano.
- **Inserimento dati anagrafici**: Dopo la registrazione, ogni cliente deve fornire i propri dati anagrafici utilizzando le credenziali di accesso (codice fiscale, e-mail e password).
- **Indirizzi di spedizione**:
  Ogni cliente deve inserire almeno un indirizzo di spedizione valido.
  Il primo indirizzo registrato viene automaticamente impostato come predefinito e marcato come "attivo". Il sito consente l'aggiunta di ulteriori indirizzi di spedizione.
- **Catalogo prodotti**: Ogni cliente ha la possibilità di visualizzare un catalogo di box, inclusi i dettagli sui **valori nutrizionali** e sugli eventuali **allergeni** di ciascun prodotto.
- **Gestione del carrello**:
  Ogni cliente dispone di un carrello personale.
  È possibile aggiungere prodotti, visualizzare le quantità e consultare il totale complessivo del carrello.
- **Ordini**: Il cliente può effettuare un ordine e visualizzarne tutti i dettagli.
  Ad ogni ordine sono associate una fattura e una spedizione e la lista dettagliata dell'ordine.
  Durante l'acquisto, è possibile applicare un codice sconto valido.

Glossario

TERMINE	DEFINIZIONE	SINONIMI	OMONIMI
Cliente	Persona registrata e abilitata a effettuare acquisti sul sito	Utente, acquirente	--
Box	Confezione contenente prodotti selezionati	Scatola, confezione	--
Allergeni	Sostanze che possono provocare reazioni allergiche	Agenti allergenici	--
Valori Nutrizionali	Informazioni relative al contenuto nutrizionale di un prodotto	Proprietà alimentari, dati nutrizionali	--
Carrello	Sezione in cui il cliente aggiunge i prodotti prima dell'acquisto	Lista acquisti, shopping bag	--
Ordine	Richiesta di acquisto di uno o più prodotti da parte del cliente	Acquisto, transazione	--
Composizione Box	Elenco degli ingredienti inclusi in una specifica box	Contenuto, assortimento	--
Ingredienti	Materie prime che compongono un prodotto/box	Componenti, materie prime	--
Registrazione	Procedura per creare un account sul sito	Iscrizione, abilitazione	--

Diagramma EE/R

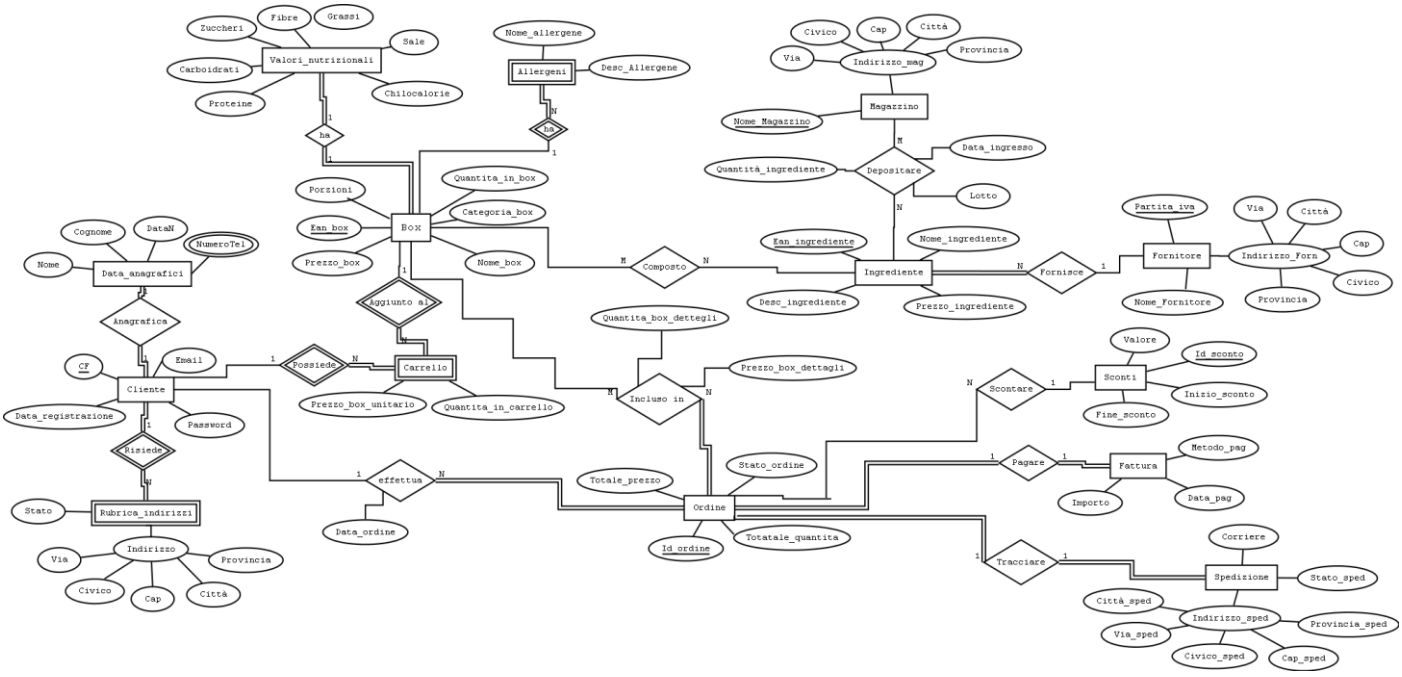
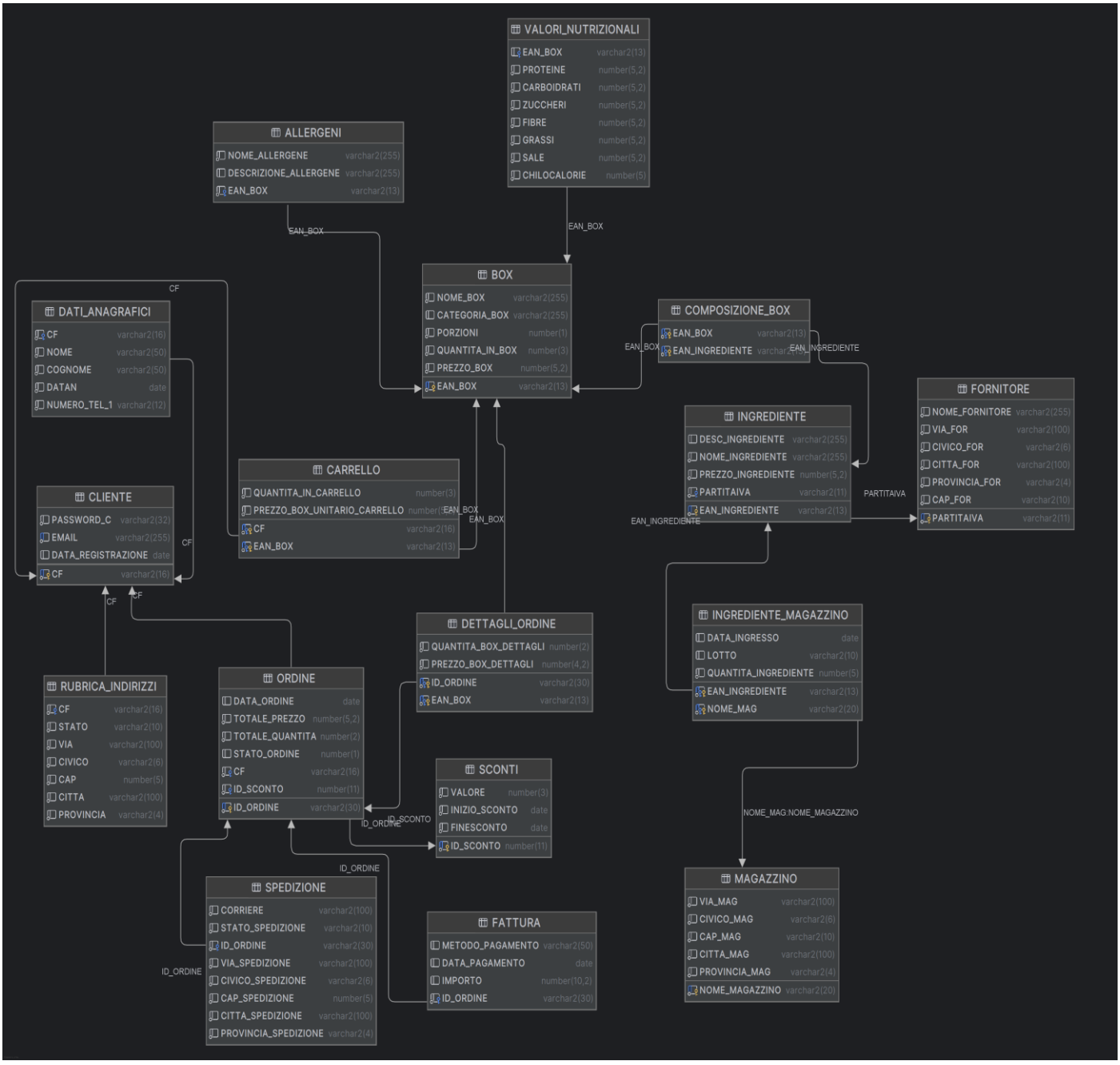


Diagramma Relazionale



Utenti

Utente: Admin

```
CREATE USER admin_ alimentari IDENTIFIED BY admin;
GRANT ALL PRIVILEGES TO admin_alimentari;
```

L'utente **admin** è l'utente che rappresenta gli amministratori del sistema dell'e-commerce.

Gli admin hanno **accesso completo** a tutte le tabelle e funzionalità del database. Possono visualizzare, inserire, aggiornare e cancellare dati in qualsiasi tabella, garantendo il controllo totale sull'intero sistema. Questo include la gestione dei clienti, dei dipendenti e dei fornitori, nonché la supervisione degli ordini, del magazzino e delle spedizioni. Gli admin possono gestire i profili dei clienti e dei dipendenti, modificandone attributi come password, indirizzi e altri dettagli personali o lavorativi. Inoltre, hanno la possibilità di creare, aggiornare e rimuovere account di dipendenti e clienti.

Per quanto riguarda il catalogo, gli admin possono inserire nuovi prodotti, aggiornare quelli esistenti e modificarne i dettagli, inclusi valori nutrizionali, allergeni, composizione e prezzo. Possono gestire le tabelle relative al magazzino (**MAGAZZINO e INGREDIENTE_MAGAZZINO**), aggiornando le scorte o modificando i dettagli dei fornitori.

Relativamente agli ordini, gli admin hanno accesso completo sia alla gestione dei carrelli che a quella degli ordini effettuati, con la possibilità di intervenire su dettagli, stato degli ordini e informazioni di fatturazione. Hanno anche privilegi completi sulle spedizioni, potendo modificarne lo stato o i dettagli. Gli admin gestiscono la tabella degli sconti, potendo creare, aggiornare o rimuovere promozioni, e hanno accesso alla gestione finanziaria, comprese le fatture (**FATTURA**) e i pagamenti.

Oltre alla gestione operativa, gli admin possono anche definire e modificare i ruoli e i privilegi degli utenti del database, garantendo la sicurezza e l'adeguata distribuzione dei permessi.

Utente: Dipendente

```
CREATE USER dipendente IDENTIFIED BY Password_dipendente;
GRANT CONNECT TO dipendente;
GRANT CREATE SESSION TO dipendente;
GRANT SELECT, INSERT, UPDATE, DELETE ON BOX TO dipendente;
GRANT SELECT, INSERT, UPDATE, DELETE ON VALORI_NUTRIZIONALI TO dipendente;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALLERGENI TO dipendente;
GRANT SELECT, INSERT, UPDATE, DELETE ON COMPOSIZIONE_BOX TO dipendente;
GRANT SELECT, INSERT, UPDATE, DELETE ON INGREDIENTE TO dipendente;
GRANT SELECT, INSERT, UPDATE, DELETE ON INGREDIENTE_MAGAZZINO TO dipendente;
GRANT SELECT, INSERT, UPDATE, DELETE ON MAGAZZINO TO dipendente;
GRANT SELECT, UPDATE ON FORNITORE TO dipendente;
GRANT SELECT, UPDATE ON ORDINE TO dipendente;
GRANT SELECT ON DETTAGLI_ORDINE TO dipendente;
GRANT SELECT, UPDATE ON SPEDIZIONE TO dipendente;
```

L'utente **dipendente** è l'utente che rappresenta i dipendenti operativi dell'e-commerce.

I dipendenti hanno accesso alle tabelle che gestiscono il catalogo dei prodotti, inclusi i dettagli relativi ai valori nutrizionali, agli allergeni e alla composizione dei box. Possono inserire nuovi prodotti, aggiornare quelli esistenti e rimuovere prodotti obsoleti. Questo accesso consente loro di mantenere aggiornato il catalogo in base alle esigenze aziendali.

I dipendenti possono anche gestire le tabelle relative al magazzino e agli ingredienti, inserendo nuovi stock, aggiornando le quantità disponibili o eliminando ingredienti non più utilizzati. Hanno accesso ai dettagli dei fornitori, con la possibilità di aggiornare informazioni rilevanti come l'indirizzo o il nome del fornitore, ma non possono inserire nuovi fornitori nel sistema.

Relativamente agli ordini, i dipendenti possono consultare le informazioni sugli ordini e modificarne lo stato operativo (ad esempio, aggiornare uno stato d'ordine da "in lavorazione" a "spedito"). Tuttavia, non possono creare nuovi ordini né intervenire direttamente sulle tabelle che gestiscono i carrelli dei clienti.

I dipendenti gestiscono anche le spedizioni, potendo aggiornare lo stato della consegna e i dettagli logistici. Non hanno accesso alle tabelle relative ai clienti, come CLIENTE, RUBRICA_INDIRIZZI, o DATI_ANAGRAFICI, e non possono visualizzare o modificare le informazioni personali dei clienti. Non hanno alcun permesso di modifica sulle tabelle relative alle fatture (FATTURA) e agli sconti (SCONTI), in quanto queste operazioni sono gestite direttamente dal sistema o da utenti con privilegi specifici.

Utente: Cliente

```
CREATE USER cliente IDENTIFIED BY password_cliente;
GRANT CREATE SESSION TO cliente;
GRANT SELECT ON BOX TO cliente;
GRANT SELECT ON VALORI_NUTRIZIONALI TO cliente;
GRANT SELECT ON ALLERGENI TO cliente;
GRANT SELECT ON INGREDIENTI TO cliente
GRANT SELECT ON COMPOSIZIONE_BOX TO cliente
GRANT SELECT ON SCONTI TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON CLIENTE TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON DATI_ANAGRAFICI TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON RUBRICA_INDIRIZZI TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON CARRELLO TO cliente;
GRANT SELECT ON ORDINE TO cliente;
GRANT SELECT ON DETTAGLI_ORDINE TO cliente;
GRANT SELECT ON SPEDIZIONE TO cliente;
GRANT SELECT ON FATTURA TO cliente;
GRANT SELECT ON box_senza_lattosio TO cliente;
GRANT SELECT ON box_alto_contenuto_proteine TO cliente;
GRANT EXECUTE ON effettua_ordine TO cliente;
GRANT EXECUTE ON mostra_box_da_ingrediente TO cliente;
GRANT EXECUTE ON Aggiungi_Dati_Anagrafici TO cliente;
GRANT EXECUTE ON Aggiungi_Indirizzo TO cliente;
GRANT EXECUTE ON mostra_composizione_box TO cliente;
```

L'utente **cliente** è l'utente che rappresenta i clienti registrati dell'e-commerce.
I clienti possono consultare il catalogo dei prodotti disponibili, visualizzandone i dettagli, come le informazioni nutrizionali e gli allergeni associati, ma non possono in alcun modo modificare o inserire nuovi prodotti nel sistema.

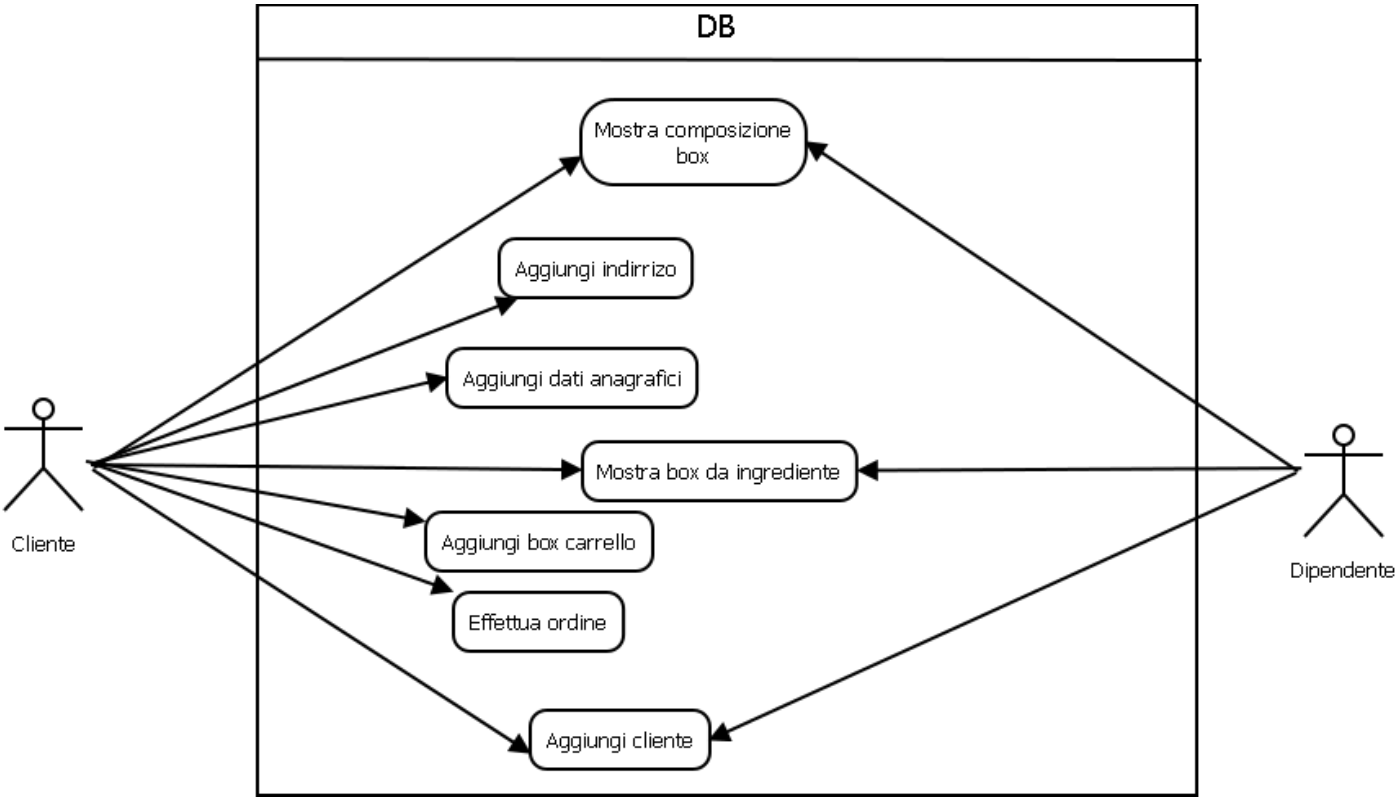
I clienti hanno la possibilità di gestire il proprio profilo personale aggiornando attributi della tabella **CLIENTE** (ad esempio **EMAIL** e **PASSWORD_C**). Possono inoltre inserire, modificare o eliminare indirizzi personali salvati nella tabella **RUBRICA_INDIRIZZI**.

Per quanto riguarda il carrello, i clienti possono aggiungere, modificare o rimuovere articoli dalla tabella **CARRELLO**. Tuttavia, non hanno alcun accesso diretto alla tabella **MAGAZZINO** o al controllo delle disponibilità, che è gestito automaticamente dal sistema.

Relativamente agli ordini, i clienti possono consultare i propri ordini effettuati e i relativi dettagli nelle tabelle **ORDINE** e **DETTAGLI_ORDINE**, ma non possono apportare modifiche agli ordini già effettuati. Inoltre, hanno accesso alle informazioni sulle fatture (**FATTURA**) e alle spedizioni (**SPEDIZIONE**) associate ai propri ordini, senza però la possibilità di intervenire direttamente su queste tabelle.

I clienti non hanno privilegi di modifica o inserimento sulle tabelle relative alla gestione dei fornitori (**FORNITORE, INGREDIENTE, INGREDIENTE_MAGAZZINO**) o sulle tabelle che determinano la composizione dei prodotti (**COMPOSIZIONE_BOX**).

Operazioni degli utenti



Composizione box

Questa procedura fornisce i dettagli relativi alla composizione di un "box", ovvero i prodotti contenuti in un determinato box, con informazioni su quantità e EAN univoco degli ingredienti.
Questa procedura viene eseguita da un cliente o un dipendente che desidera visualizzare le informazioni relative ai prodotti inclusi in un box durante la fase di esplorazione del catalogo.
Codice errore: -20016 / -20017 / -20099

Aggiungi indirizzo

Consente di aggiungere un nuovo indirizzo associato a un cliente, che può essere utilizzato per la spedizione degli ordini. Questa procedura viene eseguita da un cliente che sta effettuando un ordine online o un amministratore che sta gestendo il profilo del cliente.
Codice errore: -20005 / -20020 / -20099

Aggiungi dati anagrafici

Aggiunge o aggiorna i dati anagrafici di un cliente nel sistema, come nome, cognome, data di nascita, e altre informazioni personali. Questa procedura viene eseguita da un **cliente** che si registra o aggiorna il proprio profilo o un **amministratore** che inserisce i dati a seguito di una registrazione manuale.
Codice errore: -20005 / -20022 / -20020 / -20099

Mostra box ingredienti

Permette di visualizzare tutti i "box" che contengono un determinato ingrediente. Questo può essere utile per i clienti che vogliono evitare allergeni o cercare prodotti con specifici ingredienti. Questa procedura viene eseguita da un cliente, dipendente o amministratore che desidera esplorare i prodotti a partire dai singoli ingredienti.
Codice errore: -20023 / -20024 / -20099

Aggiungi box carrello

Aggiunge un box al carrello del cliente per un possibile acquisto.
Questa procedura viene eseguita da un cliente che sta facendo acquisti sul sito e vuole aggiungere uno o più prodotti al proprio carrello.
Codice errore: -20005 / -20003 / -20099

Effettua ordine

Questa procedura gestisce la transazione finale di acquisto. Sposta i prodotti dal carrello a un ordine confermato, includendo la gestione della disponibilità dei prodotti, l'elaborazione del pagamento e la conferma dell'ordine.
Questa procedura viene eseguita da un cliente che finalizza l'acquisto oppure un amministratore in caso di ordini gestiti manualmente.
Codice errore: -20025 / -20026 / -20004 / -20003 / -20099

Aggiungi cliente

Aggiunge un nuovo cliente al sistema, registrando informazioni come nome, codice fiscale, e-mail, indirizzo di spedizione e altre informazioni necessarie.
Questa procedura viene eseguita da un cliente che si registra autonomamente sul sito o un amministratore o dipendente che inserisce i dati di un cliente in fase di registrazione manuale o per motivi amministrativi.
Codice errore: -20018 / -20019 / -20020 / -20021 / -20099

Vincoli di integrità

Statici:

Qui di seguito sono riportati i vincoli statici delle tabelle, in ordine di creazione delle tabelle relative (Non sono riportati i vincoli dinamici e i vincoli di chiavi primarie e chiavi esterne):

- Nella tabella **Cliente** è necessario che i campi CF ,password_c ed e-mail non siano vuoti. Inoltre, l'e-mail deve essere univoco. Il campo data_registrazione viene impostato di default a sysdate.
- Nella tabella **Rubrica_indirizzzi** è necessario che tutti i campi siano non vuoti. Lo stato deve avere un valore compreso tra "attivo" e "inattivo" .
- Nella tabella **Data_anagrafici** tutti i campi non possono essere vuoti e la chiave esterna(cf) deve essere unica e non vuoto.
- Nella tabella **Box** tutti i campi non devono essere vuoti tranne categoria_box che puo essere mancante e verrà messa null di default.
- Nella tabella **Valori_nutrizionali** è necessario che tutti i campi non siano vuoti e devono essere maggiori uguali di 0(non possono avere un valore negativo).
- Nella tabella **Allergeni** l'attributo nome_allergene non puo essere null mentre descrizione_allergene è impostato di default null.
- Nella tabella **Magazzino** tutti i valori non posso essere null e cap_mag deve avere una lunghezza uguale a 5.
- Nella tabella **Fornitore** i valori non posso essere null e cap_for deve avere una lunghezza uguale a 5.
- Nella tabella **Ingrediente_Magazzino** i valori non posso essere null tranne per desc_ingrediente se non viene impostato di default è null.
- Nella tabella **Ingrediente** è necessario che i campi ean_ingrediente, nome_mag e quantita_ingrediente non siano vuoti. I campi data_ingresso e lotto se non impostato vengono posti di default a null.
- Nella tabella **Composizione_box** è necessario che tutti i campi non siano vuoti.
- Nella tabella **Carrello** è necessario che tutti i campi non siano vuoti.
- Nella tabella **Sconti** è necessario che i campi valore_sconto, inizio_sconto e fine_sconto non siano vuoti. Per verificare che lo sconto sia valido: inizio_sconto non può essere maggiore di fine_sconto.
- Nella tabella **Ordine** i campi totale_prezzo e totale_quantità non devono essere vuoti, data_ordine e stato_ordine vengono impostati di default a null poichè vengono impostati

tramite la procedura effettua_ordine. Stato_ordine deve avere un valore compreso tra 1 e 4 .(1= ”in preparazione”, 2= ”spedito”, 3= ”consegnato”, 4= ”cancellato”).

- Nella tabella **Dettagli_ordine** è necessario che tutti i campi non siano vuoti.
- Nella tabella **Fattura** tutti i campi sono impostati di default a null ma vengono inseriti tramite procedure effettua_ordine e la chiave esterna(id_ordine) deve essere unica e non vuoto.
- Nella tabella **Spedizione** è necessario che tutti i campi non siano vuoti e stato spedizione deve avere un valore compreso tra 1 e 4. (1=” in preparazione”, 2= ”spedito”, 3= ”consegnato”, 4= ”cancellato”) e la chiave esterna(id_ordine) deve essere unica e non vuoto.

Dinamici:

Qui di seguito sono riportati i vincoli dinamici delle tabelle, ovvero vincoli implementati tramite triggers, in ordine di implementazione nel file *triggers.sql:*

- *Prezzo_unitario_carrello*

  Questo trigger viene eseguito prima di un'operazione di inserimento nella tabella CARRELLO Calcola e assegna automaticamente il prezzo unitario del prodotto (PREZZO_BOX_UNITARIO_CARRELLO) al record che sta per essere inserito nella tabella CARRELLO, recuperando il prezzo dalla tabella BOX.

- *Controllo_disponibilità_box*

  Questo trigger viene eseguito **prima di un'operazione di inserimento** nella tabella CARRELLO. Controlla la validità dei dati del nuovo record prima che venga inserito.

- Prevenzione_cancellazione_cliente

  Questo trigger si attiva **prima dell'eliminazione di un record** dalla tabella CLIENTE. Impedisce l'eliminazione di un cliente se ha ordini associati nella tabella ORDINE

- Cancellazione_ordine

  Questo trigger si attiva **dopo l'aggiornamento di un record** nella tabella ORDINE.
  Gestisce la **cancellazione logica di un ordine** (quando Stato_ordine passa a 4) aggiornando la disponibilità dei box associati.

- Inizializza_spedizione

  Questo trigger si attiva **dopo l'inserimento di un nuovo record** nella tabella ORDINE. La sua funzione principale è creare una spedizione associata all'ordine appena inserito, utilizzando i dettagli dell'indirizzo attivo del cliente.

- Verifica_registrazione_clienti

  Questo trigger si attiva **prima dell'inserimento** di un nuovo record nella tabella cliente e ha il compito di garantire l'unicità dei campi: **Codice fiscale (CF) e Email.**

Verifica di normalità

Prima forma normale

- *Il database è normalizzato nella **prima forma normale**, poiché soddisfa i requisiti fondamentali di questa forma. In particolare, tutte le tabelle del database contengono valori atomici, ovvero ogni campo è progettato per contenere un singolo valore per riga, senza la presenza di gruppi o liste di valori.*

  Ad esempio, nelle tabelle CLIENTE, BOX, ORDINE e in tutte le altre, le colonne come CF, EMAIL, NOME_BOX contengono singoli valori distinti per ogni riga. Inoltre, ogni tabella dispone di una chiave primaria univoca, che identifica in modo inequivocabile ogni record. Per esempio, la colonna CF della tabella CLIENTE è una chiave primaria che garantisce l'unicità di ciascun cliente, mentre la tabella BOX è identificata dalla colonna EAN_BOX. Infine, non sono presenti righe duplicate, poiché tutte le chiavi primarie sono uniche e non si ripetono. Pertanto, il database è conforme alla prima forma normale, evitando la presenza di strutture complesse o non atomiche.

Seconda forma normale

- *Il database è normalizzato nella **seconda forma normale**, poiché soddisfa i requisiti della prima forma normale e, inoltre, ogni attributo non chiave dipende completamente dalla chiave primaria. In altre parole, non esistono dipendenze parziali, ovvero attributi che dipendono solo da una parte della chiave primaria nelle tabelle che utilizzano una chiave primaria composta.*

  Ad esempio, nella tabella DETTAGLI_ORDINE, che ha come chiave primaria composta ID_ORDINE e EAN_BOX, ogni attributo, come QUANTITA_BOX_DETTAGLI e PREZZO_BOX_DETTAGLI, dipende completamente dalla chiave composta, senza alcuna dipendenza parziale. Allo stesso modo, nella tabella CARRELLO, la chiave primaria è costituita dalla combinazione di CF e EAN_BOX, e ogni attributo (come QUANTITA_IN_CARRELLO e PREZZO_BOX_UNITARIO_CARRELLO) dipende interamente da entrambe le colonne chiave. Inoltre, tutte le altre tabelle, come CLIENTE, RUBRICA_INDIRIZZI e ORDINE, non presentano attributi che dipendono parzialmente da una porzione della chiave primaria. In questo modo, il database è stato progettato per eliminare la ridondanza e le anomalie di aggiornamento che potrebbero derivare da dipendenze parziali. Pertanto, il database è conforme alla seconda forma normale.

Terza forma normale

- *Il database è normalizzato nella **terza forma normale** , in quanto soddisfa tutte le condizioni della prima e della seconda forma normale e, inoltre, elimina le dipendenze transitive tra gli*

attributi. In altre parole, ogni attributo non chiave è direttamente dipendente dalla chiave primaria e non da altri attributi non chiave.

Per esempio, nella tabella CLIENTE, l'attributo CF è la chiave primaria e gli altri attributi (come PASSWORD_C, EMAIL e DATA_REGISTRAZIONE) dipendono direttamente da essa. Non ci sono attributi che dipendono da altri non chiave.

In altre tabelle come ORDINE, DETTAGLI_ORDINE e SCONTI, tutte le colonne che non fanno parte della chiave primaria dipendono unicamente dalla chiave primaria e non da altri attributi non chiave. Ad esempio, in ORDINE, l'attributo TOTALE_PREZZO dipende direttamente dall'ID ordine (la chiave primaria) e non da altri attributi come STATO_ORDINE, che è anch'esso dipendente dall'ID ordine.

Inoltre, nelle tabelle BOX, VALORI_NUTRIZIONALI e ALLERGENI, i dati nutrizionali e gli allergeni sono separati in tabelle distinte con relazioni chiare e senza dipendenze transitive. Ogni valore nutrizionale o allergene dipende direttamente dal codice del prodotto (EAN_BOX), che è la chiave primaria.

Tutti questi esempi dimostrano che il database è progettato per evitare ridondanze e anomalie di aggiornamento, inserimento e cancellazione, garantendo così la coerenza e l'integrità dei dati. Pertanto, il database è normalizzato correttamente nella terza forma.

Implementazione

Data Definition Language (DDL):

```
CREATE TABLE CLIENTE
(
   CF       VARCHAR2(16)  not null
      primary key,
   PASSWORD_C VARCHAR2(32)  not null,
   EMAIL     VARCHAR2(255) not null UNIQUE,
   DATA_REGISTRAZIONE DATE DEFAULT SYSDATE
);
```

La tabella CLIENTE rappresenta i clienti registrati sulla piattaforma. Ogni cliente è identificato in modo univoco tramite il proprio codice fiscale (CF), che funge da chiave primaria. Oltre al codice fiscale, sono memorizzate le informazioni necessarie per l'accesso, come la password (PASSWORD_C) e l'e-mail (EMAIL), che deve essere univoca per ciascun utente. La colonna DATA_REGISTRAZIONE registra la data in cui il cliente ha completato la registrazione, con un valore predefinito pari alla data corrente. Questa tabella costituisce il punto di partenza per la gestione dei dati dei clienti, utilizzata per autenticazione e gestione degli ordini.

```
CREATE TABLE RUBRICA_INDIRIZZI
(
   CF       VARCHAR2(16)  not null,
   STATO    VARCHAR2(10) not NULL
      check (Stato IN ('attivo', 'inattivo')),
   VIA      VARCHAR2(100)          not null,
   CIVICO   VARCHAR2(6)            not null,
   CAP      NUMBER(5)              not null,
   CITTA    VARCHAR2(100)          not null,
   PROVINCIA VARCHAR2(4)           not null,
  CONSTRAINT FK_RUBRICA_CLIENTE FOREIGN KEY (CF) REFERENCES CLIENTE(CF)
);
```

La tabella RUBRICA_INDIRIZZI gestisce gli indirizzi associati ai clienti registrati nel sistema. Ogni indirizzo è legato in modo univoco a un cliente attraverso la chiave esterna CF, che fa riferimento al codice fiscale della tabella CLIENTE. La colonna STATO specifica se l'indirizzo è attualmente "attivo" o "inattivo", permettendo di distinguere gli indirizzi utilizzabili per ordini da quelli non più validi. Vengono inoltre memorizzati i dettagli completi dell'indirizzo, tra cui VIA, CIVICO, CAP, CITTÀ e

PROVINCIA, garantendo informazioni sufficienti per la gestione logistica. Questa tabella è fondamentale per la gestione di spedizioni e corrispondenze legate ai clienti.

```
CREATE TABLE DATI_ANAGRAFICI
(
   CF         VARCHAR2(16) not NULL UNIQUE ,
   NOME       VARCHAR2(50)  not null,
   COGNOME    VARCHAR2(50)  not null,
   DATAN      DATE          not null,
   NUMERO_TEL_1 VARCHAR2(12)  not null,
  CONSTRAINT FK_DATI_ANAGRAFICI_CLIENTE FOREIGN KEY (CF) REFERENCES CLIENTE(CF)
);
```

La tabella DATI_ANAGRAFICI raccoglie le informazioni personali essenziali dei clienti registrati nel sistema. Ogni record è associato a un cliente specifico tramite la chiave esterna CF, che fa riferimento alla colonna corrispondente della tabella CLIENTE. Vengono memorizzati il nome (NOME), il cognome (COGNOME), la data di nascita (DATAN) e un numero di telefono principale (NUMERO_TEL_1). Questi dati sono fondamentali per identificare e contattare i clienti, offrendo un'integrazione tra la gestione anagrafica e le altre funzionalità del sistema.

```
CREATE TABLE BOX
(
EAN_BOX       VARCHAR2(13) PRIMARY KEY,
NOME_BOX      VARCHAR2(255) NOT NULL,
CATEGORIA_BOX  VARCHAR2(255) DEFAULT NULL,
PORZIONI      NUMBER(1)    NOT NULL,
QUANTITA_IN_BOX NUMBER(3)    NOT NULL,
PREZZO_BOX     NUMBER(5, 2)  NOT NULL
);
```

La tabella BOX rappresenta l'insieme dei prodotti offerti dall'e-commerce, con ogni prodotto identificato in modo univoco tramite un codice riconosciuto a livello internazionale (EAN_BOX), che funge da chiave primaria. Per ogni prodotto vengono memorizzati il nome (NOME_BOX), la categoria merceologica (CATEGORIA_BOX), e il numero di porzioni contenute nel box (PORZIONI). Inoltre, sono specificati la quantità di unità disponibili in ogni box (QUANTITA_IN_BOX) e il prezzo unitario del box (PREZZO_BOX). Questa tabella è essenziale per la gestione dell'inventario e per le operazioni di vendita, permettendo un’organizzazione chiara dei prodotti.

```
CREATE TABLE VALORI_NUTRIZIONALI (
EAN_BOX  VARCHAR2(13) not null unique,
PROTEINE NUMBER(5, 2) CHECK (PROTEINE >= 0) not null,
CARBOIDRATI NUMBER(5, 2) CHECK (CARBOIDRATI >= 0) not null,
ZUCCHERI NUMBER(5, 2) CHECK (ZUCCHERI >= 0) not null,
FIBRE NUMBER(5, 2) CHECK (FIBRE >= 0) not null,
GRASSI NUMBER(5, 2) CHECK (GRASSI >= 0) not null,
SALE NUMBER(5, 2) CHECK (SALE >= 0) not null,
CHILOCALORIE NUMBER(5) CHECK (CHILOCALORIE >= 0) not null,
CONSTRAINT FK_VALORI_NUTRIZIONALI_BOX FOREIGN KEY (EAN_BOX) REFERENCES
  BOX(EAN_BOX)
);
```

La tabella VALORI_NUTRIZIONALI contiene i dettagli relativi al contenuto nutrizionale dei prodotti presenti nel catalogo dell'e-commerce. Ogni record è associato a un prodotto specifico tramite la chiave esterna EAN_BOX, che fa riferimento alla tabella BOX. Per ciascun prodotto vengono memorizzate le informazioni relative a proteine (PROTEINE), carboidrati (CARBOIDRATI), zuccheri (ZUCCHERI), fibre (FIBRE), grassi (GRASSI), sale (SALE) e chilocalorie (CHILOCALORIE), con vincoli che assicurano che tutti i valori siano maggiori o uguali a zero. Questa tabella è fondamentale per fornire informazioni nutrizionali ai clienti e per garantire trasparenza sui prodotti venduti.

```
CREATE TABLE ALLERGENI
(
NOME_ALLERGENE      VARCHAR2(255) NOT NULL ,
DESCRIZIONE_ALLERGENE VARCHAR2(255) DEFAULT NULL,
EAN_BOX  VARCHAR2(13)  NOT NULL,
CONSTRAINT FK_ALLERGENI_EAN_BOX FOREIGN KEY (EAN_BOX) REFERENCES BOX(EAN_BOX)
);
```

La tabella ALLERGENI raccoglie le informazioni sugli allergeni presenti nei prodotti offerti dall'e-commerce. Ogni record specifica il nome dell'allergene (NOME_ALLERGENE) e, opzionalmente, una descrizione dettagliata (DESCRIZIONE_ALLERGENE). La chiave esterna EAN_BOX collega ciascun allergene al prodotto corrispondente nella tabella BOX, garantendo che ogni allergene sia associato a un prodotto esistente. Questa tabella è essenziale per la gestione delle informazioni alimentari, offrendo ai clienti dati chiari e completi per fare scelte consapevoli in base alle loro esigenze dietetiche o restrizioni mediche.

```
CREATE TABLE MAGAZZINO
(
   NOME_MAGAZZINO VARCHAR2(20)
      constraint PK_MAGAZZINO
         primary key,
   VIA_MAG        VARCHAR2(100) not null,
   CIVICO_MAG     VARCHAR2(6)   not null,
   CAP_MAG VARCHAR2(10) CHECK (LENGTH(CAP_MAG) = 5) not null,
   CITTA_MAG      VARCHAR2(100) not null,
   PROVINCIA_MAG   VARCHAR2(4)   not null
);
```

La tabella MAGAZZINO rappresenta i punti di stoccaggio fisici utilizzati per conservare e distribuire i prodotti dell'e-commerce. Ogni magazzino è identificato in modo univoco tramite il suo nome (NOME_MAGAZZINO), che funge da chiave primaria. Per ogni magazzino vengono memorizzati l'indirizzo completo, comprensivo di via (VIA_MAG), numero civico (CIVICO_MAG), CAP (CAP_MAG), città (CITTA_MAG) e provincia (PROVINCIA_MAG). È previsto un vincolo di controllo sul CAP_MAG per assicurare che abbia una lunghezza di 5 caratteri. Questa tabella è fondamentale per la gestione della logistica e per tracciare la posizione dei prodotti stoccati.

```
CREATE TABLE FORNITORE
(
   PARTITAIVA   VARCHAR2(11)  not null
      constraint PK_FORNITORE
         primary key,
   NOME_FORNITORE   VARCHAR2(255) not null,
   VIA_FOR     VARCHAR2(100) not NULL,
   CIVICO_FOR   VARCHAR2(6)   not NULL,
   CITTA_FOR    VARCHAR2(100) not NULL,
   PROVINCIA_FOR VARCHAR2(4)   not NULL,
   CAP_FOR VARCHAR2(10) CHECK (LENGTH(CAP_FOR) = 5) not null
);
```

La tabella FORNITORE rappresenta i fornitori che riforniscono l’e-commerce degli ingredienti per comporre dei box. Ogni fornitore è identificato in modo univoco dalla propria partita IVA (PARTITAIVA), che funge da chiave primaria. Sono registrati il nome del fornitore (NOME_FORNITORE) e l'indirizzo completo, che include via (VIA_FOR), numero civico (CIVICO_FOR), città (CITTA_FOR), provincia (PROVINCIA_FOR) e CAP (CAP_FOR), con un vincolo che garantisce che il CAP sia sempre di 5 caratteri. Questa tabella è cruciale per mantenere una tracciabilità chiara dei fornitori e per facilitare la gestione degli ordini di approvvigionamento e delle relazioni commerciali.

```
CREATE TABLE INGREDIENTE
(
   EAN_INGREDIENTE    VARCHAR2(13) not null
      constraint PK_INGREDIENTE
         primary key,
   DESC_INGREDIENTE   VARCHAR2(255) default NULL,
   NOME_INGREDIENTE   VARCHAR2(255) not NULL,
   PREZZO_INGREDIENTE NUMBER(5, 2)  not NULL,
   PARTITAIVA         VARCHAR2(11)  not NULL,
    CONSTRAINT FK_INGREDIENTE  FOREIGN KEY (PARTITAIVA) REFERENCES
     FORNITORE(PARTITAIVA)
);
```

La tabella INGREDIENTE rappresenta gli ingredienti utilizzati per la composizione dei prodotti offerti dall'e-commerce (BOX). Ogni ingrediente è identificato in modo univoco tramite un codice univoco (EAN_INGREDIENTE), che funge da chiave primaria. Per ciascun ingrediente vengono memorizzati la descrizione opzionale (DESC_INGREDIENTE), il nome (NOME_INGREDIENTE) e il prezzo unitario (PREZZO_INGREDIENTE). Inoltre, ogni ingrediente è associato a un fornitore specifico attraverso la chiave esterna PARTITAIVA, che fa riferimento alla tabella FORNITORE. Questa tabella è fondamentale per gestire le materie prime, monitorare i costi e mantenere una tracciabilità completa delle relazioni con i fornitori.

```
CREATE TABLE INGREDIENTE_MAGAZZINO (
   DATA_INGRESSO DATE DEFAULT NULL,
   LOTTO VARCHAR2(10) DEFAULT NULL,
   QUANTITA_INGREDIENTE NUMBER(5) NOT NULL,
   EAN_INGREDIENTE VARCHAR2(13) NOT NULL,
   NOME_MAG VARCHAR2(20) NOT NULL,
   CONSTRAINT FK_INGREDIENTE_MAG_EAN_INGREDIENTE FOREIGN KEY (EAN_INGREDIENTE)
     REFERENCES INGREDIENTE(EAN_INGREDIENTE),
   CONSTRAINT FK_INGREDIENTE_MAG_NOME_MAG FOREIGN KEY (NOME_MAG) REFERENCES
     MAGAZZINO(NOME_MAGAZZINO),
   CONSTRAINT PK_INGREDIENTE_MAGAZZINO PRIMARY KEY (EAN_INGREDIENTE, NOME_MAG)
);
```

La tabella INGREDIENTE_MAGAZZINO gestisce la relazione tra gli ingredienti e i magazzini dove sono stoccati. Ogni record rappresenta una specifica quantità di un ingrediente presente in un determinato magazzino. La chiave primaria composta da EAN_INGREDIENTE e NOME_MAG identifica univocamente l'ingresso di un ingrediente in un magazzino. La colonna DATA_INGRESSO memorizza la data in cui l'ingrediente è stato immesso nel magazzino, mentre LOTTO tiene traccia di eventuali lotti di produzione. La quantità disponibile dell'ingrediente in magazzino è registrata nella colonna

QUANTITA_INGREDIENTE. Le chiavi esterne EAN_INGREDIENTE e NOME_MAG collegano questa tabella rispettivamente alla tabella INGREDIENTE e alla tabella MAGAZZINO. Questa tabella è cruciale per gestire l'inventario degli ingredienti nei vari magazzini.

```
CREATE TABLE COMPOSIZIONE_BOX (
  EAN_BOX VARCHAR2(13) NOT NULL,
  EAN_INGREDIENTE VARCHAR2(13) NOT NULL,
  CONSTRAINT FK_COMP_EAN_BOX FOREIGN KEY (EAN_BOX) REFERENCES BOX(EAN_BOX),
  CONSTRAINT FK_COMP_EAN_INGREDIENTE FOREIGN KEY (EAN_INGREDIENTE) REFERENCES
    INGREDIENTE(EAN_INGREDIENTE),
  CONSTRAINT PK_COMPOSIZIONE_BOX PRIMARY KEY (EAN_BOX, EAN_INGREDIENTE)
);
```

La tabella COMPOSIZIONE_BOX definisce la relazione tra i prodotti (BOX) e gli ingredienti che li compongono. Ogni record rappresenta un ingrediente specifico che fa parte di un determinato box. La chiave primaria composta da EAN_BOX e EAN_INGREDIENTE assicura che ogni combinazione di box e ingrediente sia univoca. Le chiavi esterne EAN_BOX e EAN_INGREDIENTE collegano rispettivamente questa tabella alla tabella BOX e alla tabella INGREDIENTE, stabilendo la composizione di ciascun prodotto. Questa tabella è essenziale per la gestione delle informazioni sui prodotti, indicando quali ingredienti sono contenuti in ciascun box venduto nell'e-commerce.

```
CREATE TABLE CARRELLO (
  CF                  VARCHAR2(16)   NOT NULL,
  EAN_BOX             VARCHAR2(13)   NOT NULL,
  QUANTITA_IN_CARRELLO     NUMBER(3)     NOT NULL,
  PREZZO_BOX_UNITARIO_CARRELLO NUMBER(5, 2) NOT NULL,
  CONSTRAINT FK_CARRELLO_CF  FOREIGN KEY (CF) REFERENCES CLIENTE(CF),
  CONSTRAINT FK_CARRELLO_EAN_BOX FOREIGN KEY (EAN_BOX) REFERENCES BOX(EAN_BOX),
  CONSTRAINT PK_CARRELLO PRIMARY KEY (CF, EAN_BOX)
);
```

La tabella CARRELLO rappresenta il carrello degli acquisti dei clienti nel sistema di e-commerce. Ogni record contiene informazioni relative a un prodotto (EAN_BOX) e alla sua quantità (QUANTITA_IN_CARRELLO) nel carrello di un cliente identificato tramite il suo codice fiscale (CF). La colonna PREZZO_BOX_UNITARIO_CARRELLO memorizza il prezzo unitario del prodotto al momento dell'inserimento nel carrello. La chiave primaria è composta da CF e EAN_BOX, assicurando che ogni cliente possa avere una sola quantità di ciascun prodotto nel proprio carrello. Le chiavi esterne collegano questa tabella rispettivamente alle tabelle CLIENTE e BOX, garantendo che solo clienti registrati possano aggiungere prodotti al carrello e che i prodotti esistano nel catalogo. Questa tabella è fondamentale per la gestione del processo di acquisto e per il calcolo del totale dell'ordine.

```
CREATE TABLE SCONTI
(
  ID_SCONTO    NUMBER(11)
    CONSTRAINT PK_SCONTI
      PRIMARY KEY,
  VALORE       NUMBER(3) not NULL,
  INIZIO_SCONTO DATE      not NULL,
  FINESCONTO   DATE      not NULL,
  CONSTRAINT CHK_DATE_SCONTI
    CHECK (INIZIO_SCONTO < FINESCONTO)
);
```

La tabella SCONTI memorizza le informazioni relative agli sconti applicabili agli ordini. Ogni sconto è identificato in modo univoco tramite un identificativo (ID_SCONTO), che funge da chiave primaria. Il valore dello sconto è rappresentato dalla colonna VALORE, che indica la percentuale o l'importo dello sconto applicato. Le date di inizio (INIZIO_SCONTO) e fine (FINESCONTO) definiscono il periodo di validità dello sconto, con un vincolo che assicura che la data di inizio sia sempre precedente a quella di fine. Questa tabella è essenziale per la gestione delle promozioni e per l'applicazione automatica degli sconti durante il processo di acquisto.

```
CREATE TABLE ORDINE
(
  ID_ORDINE     VARCHAR2(30)
constraint PK_ORDINE
primary key,
  DATA_ORDINE   DATE       default NULL,
  TOTALE_PREZZO  NUMBER(5, 2) not null,
  TOTALE_QUANTITA NUMBER(2)   not null,
  STATO_ORDINE   NUMBER(1)   default NULL check (Stato_ordine IN (1, 2, 3, 4)),
  CF  VARCHAR2(16) NOT NULL,
  CONSTRAINT FK_ID_ORDINE_ORDINE  FOREIGN KEY (CF) REFERENCES CLIENTE(CF),
  ID_SCONTO     NUMBER(11)  NOT NULL,
  CONSTRAINT FK_ORDINE_ID_SCONTO  FOREIGN KEY (ID_SCONTO ) REFERENCES
    SCONTI(ID_SCONTO )
);
```

La tabella ORDINE memorizza le informazioni relative agli ordini effettuati dai clienti. Ogni ordine è identificato in modo univoco tramite l'ID ordine (ID_ORDINE), che funge da chiave primaria. La colonna DATA_ORDINE registra la data in cui l'ordine è stato effettuato, mentre TOTALE_PREZZO e TOTALE_QUANTITA rappresentano rispettivamente il prezzo totale e la quantità totale degli articoli nell'ordine. La colonna STATO_ORDINE indica lo stato dell'ordine con valori numerici che

corrispondono a diversi stadi del processo (1 (per in attesa) , 2 (per in elaborazione), 3 (per spedito), 4 (per completato). Il CF del cliente (codice fiscale) è utilizzato per associare l'ordine al cliente che l'ha effettuato, tramite una chiave esterna che fa riferimento alla tabella CLIENTE. Inoltre, l'ID dello sconto applicato, rappresentato dalla colonna ID_SCONTO, è legato alla tabella SCONTI per identificare quale sconto è stato applicato all'ordine. Questa tabella è cruciale per tracciare lo stato degli ordini, calcolare i totali e gestire le promozioni applicate.

```
CREATE TABLE DETTAGLI_ORDINE (
  ID_ORDINE VARCHAR2(30) NOT NULL,
  EAN_BOX VARCHAR2(13) NOT NULL,
  QUANTITA_BOX_DETTAGLI NUMBER(2) NOT NULL,
  PREZZO_BOX_DETTAGLI NUMBER(4, 2) NOT NULL,
  CONSTRAINT FK_DETTAGLI_ORDINE_ID_ORDINE FOREIGN KEY (ID_ORDINE) REFERENCES
    ORDINE(ID_ORDINE),
  CONSTRAINT FK_DETTAGLI_ORDINE_EAN_BOX FOREIGN KEY (EAN_BOX) REFERENCES
    BOX(EAN_BOX),
  CONSTRAINT PK_DETTAGLI_ORDINE PRIMARY KEY (ID_ORDINE, EAN_BOX)
);
```

La tabella DETTAGLI_ORDINE contiene le informazioni dettagliate sugli articoli inclusi in ogni ordine. Ogni record rappresenta un prodotto specifico (identificato dal codice EAN_BOX) all'interno di un ordine (identificato da ID_ORDINE). La chiave primaria composta da ID_ORDINE e EAN_BOX garantisce che per ogni ordine ci sia una sola riga per ciascun prodotto. La colonna QUANTITA_BOX_DETTAGLI memorizza la quantità di ciascun prodotto ordinato, mentre PREZZO_BOX_DETTAGLI registra il prezzo unitario del prodotto al momento dell'ordine. Le chiavi esterne ID_ORDINE e EAN_BOX stabiliscono un collegamento tra questa tabella e le tabelle ORDINE (per l'associazione all'ordine) e BOX (per l'associazione al prodotto specifico). Questa tabella è fondamentale per tracciare i prodotti specifici di ciascun ordine e per calcolare i totali dell'ordine.

```
CREATE TABLE FATTURA
(
  METODO_PAGAMENTO VARCHAR2(50)  default NULL,
  DATA_PAGAMENTO   DATE        default NULL,
  IMPORTO        NUMBER(10, 2) default NULL,
  ID_ORDINE      VARCHAR2(30)   not NULL unique ,
  CONSTRAINT FK_FATTURA_ID_ORDINE FOREIGN KEY (ID_ORDINE) REFERENCES
    ORDINE(ID_ORDINE)
);
```

La tabella FATTURA memorizza le informazioni relative ai pagamenti degli ordini. Ogni fattura è associata a un ordine tramite la colonna ID_ORDINE, che è una chiave esterna che fa riferimento alla

tabella ORDINE. La colonna METODO_PAGAMENTO indica il metodo utilizzato per il pagamento (ad esempio, carta di credito, bonifico, PayPal, ecc.). La colonna DATA_PAGAMENTO registra la data in cui il pagamento è stato effettuato, mentre IMPORTO rappresenta l'importo totale pagato per l'ordine. La tabella è essenziale per tracciare le transazioni finanziarie associate agli ordini e garantire la corretta gestione dei pagamenti nel sistema.

```
CREATE TABLE SPEDIZIONE
(
   CORRIERE       VARCHAR2(100) not NULL,
   STATO_SPEDIZIONE VARCHAR2(10)  not NULL check (Stato_spedizione IN ('1', '2', '3','4')),
   ID_ORDINE      VARCHAR2(30)    not NULL unique,
   CONSTRAINT FK_SPEDIZIONE_ID_ORDINE FOREIGN KEY (ID_ORDINE) REFERENCES
     ORDINE(ID_ORDINE),
   VIA_SPEDIZIONE            VARCHAR2(100) not null,
   CIVICO_SPEDIZIONE         VARCHAR2(6)   not null,
   CAP_SPEDIZIONE            NUMBER(5)     not null,
   CITTA_SPEDIZIONE          VARCHAR2(100) not null,
   PROVINCIA_SPEDIZIONE    VARCHAR2(4)   not null
);
```

La tabella SPEDIZIONE contiene informazioni relative alla spedizione di un ordine. Ogni spedizione è identificata dall'ID_ORDINE, che collega la spedizione all'ordine associato tramite una chiave esterna. La colonna CORRIERE indica il nome del corriere incaricato della spedizione. La colonna STATO_SPEDIZIONE rappresenta lo stato della spedizione e può assumere valori numerici che descrivono il processo della spedizione (1 (per in lavorazione), 2 (per spedito), 3 (per consegnato), 4 (per cancellata)). Le colonne relative all'indirizzo (VIA_SPEDIZIONE, CIVICO_SPEDIZIONE, CAP_SPEDIZIONE, CITTA_SPEDIZIONE, PROVINCIA_SPEDIZIONE) memorizzano i dettagli dell'indirizzo di destinazione della spedizione. Questa tabella è essenziale per monitorare lo stato delle spedizioni e per gestire i dettagli logistici legati alla consegna degli ordini.

Data Manipulation Language(DML):

Per evitare di avere molte righe di inserimento, abbiamo deciso di riportare solo gli “insert” essenziali per poter testare in modo basilare la base di dati. Per visionare lo script completo, riferirsi al file *popolamento.sql*

```
INSERT INTO CLIENTE (CF, PASSWORD_C, EMAIL) VALUES  ('RSSMRC85M01H501X', 'A3!rM@2024', 'marco.rossi@email.com');
INSERT INTO RUBRICA_INDIRIZZI (CF, STATO, VIA, CIVICO, CAP, CITTA, PROVINCIA) VALUES ('RSSMRC85M01H501X', 'attivo', 'Via Roma', '15', 00100, 'Roma', 'RM');
INSERT INTO DATI_ANAGRAFICI (CF, NOME, COGNOME, DATAN, NUMERO_TEL_1) VALUES ('RSSMRC85M01H501X', 'Marco', 'Rossi', TO_DATE('1985-03-01', 'YYYY-MM-DD'), '3281234567');
INSERT INTO FORNITORE (PARTITAIVA, NOME_FORNITORE, VIA_FOR, CIVICO_FOR, CITTA_FOR, PROVINCIA_FOR, CAP_FOR) VALUES ('43210987654', 'Latteria Centrale', 'Via del Latte', '3', 'Parma', 'PR', '43121');
INSERT INTO FORNITORE (PARTITAIVA, NOME_FORNITORE, VIA_FOR, CIVICO_FOR, CITTA_FOR, PROVINCIA_FOR, CAP_FOR) VALUES ('51234567890', 'Passione Asia', 'Via delle Vigne', '10', 'Pavia', 'PV', '27100');
INSERT INTO FORNITORE (PARTITAIVA, NOME_FORNITORE, VIA_FOR, CIVICO_FOR, CITTA_FOR, PROVINCIA_FOR, CAP_FOR) VALUES ('61234567891', 'Pescheria Marinara', 'Lungomare', '25', 'Catania', 'CT', '95100');
INSERT INTO MAGAZZINO (NOME_MAGAZZINO, VIA_MAG, CIVICO_MAG, CAP_MAG, CITTA_MAG, PROVINCIA_MAG) VALUES ('Magazzino Centrale', 'Via Roma', '10', '00100', 'Roma', 'RM');
INSERT INTO MAGAZZINO (NOME_MAGAZZINO, VIA_MAG, CIVICO_MAG, CAP_MAG, CITTA_MAG, PROVINCIA_MAG) VALUES ('Deposito Milano', 'Corso Buenos Aires', '23', '20100', 'Milano', 'MI');
INSERT INTO MAGAZZINO (NOME_MAGAZZINO, VIA_MAG, CIVICO_MAG, CAP_MAG, CITTA_MAG, PROVINCIA_MAG) VALUES ('Logistica Napoli', 'Via Toledo', '45B', '80100', 'Napoli', 'NA');
INSERT INTO BOX (EAN_BOX, NOME_BOX, CATEGORIA_BOX, PORZIONI, QUANTITA_IN_BOX, PREZZO_BOX) VALUES('8001234567890', 'Box Tiramisù', 'Dolci', 4, 5, 12.50);
INSERT INTO INGREDIENTE (EAN_INGREDIENTE, NOME_INGREDIENTE, DESC_INGREDIENTE, PREZZO_INGREDIENTE, PARTITAIVA) VALUES ('1234567890123', 'Zucchero semolato 100g', 'Zucchero', 2.00, '43210987654');
INSERT INTO INGREDIENTE (EAN_INGREDIENTE, NOME_INGREDIENTE, DESC_INGREDIENTE, PREZZO_INGREDIENTE, PARTITAIVA) VALUES ('1234567890456', 'Caffè in polvere per moka 80g', 'Caffè', 3.50, '43210987654');
INSERT INTO INGREDIENTE (EAN_INGREDIENTE, NOME_INGREDIENTE, DESC_INGREDIENTE, PREZZO_INGREDIENTE, PARTITAIVA) VALUES ('1234567890789', 'Mascarpone fresco 250g', 'Mascarpone', 4.50, '43210987654');
INSERT INTO INGREDIENTE (EAN_INGREDIENTE, NOME_INGREDIENTE, DESC_INGREDIENTE, PREZZO_INGREDIENTE, PARTITAIVA) VALUES ('1234567890890', 'Savoiardi di alta qualità 250g', 'Savoiardi', 3.00, '43210987654');
INSERT INTO INGREDIENTE (EAN_INGREDIENTE, NOME_INGREDIENTE, DESC_INGREDIENTE, PREZZO_INGREDIENTE, PARTITAIVA) VALUES ('1234567890125', 'Cacao amaro in polvere 200g', 'Cacao', 1.50, '43210987654');
INSERT INTO VALORI_NUTRIZIONALI (PROTEINE, CARBOIDRATI, ZUCCHERI, FIBRE, GRASSI, SALE, CHILOCALORIE, EAN_BOX)
VALUES (4.5, 22.0, 12.0, 2.5, 8.0, 0.5, 150, '8001234567890');
INSERT INTO ALLERGENI (NOME_ALLERGENE, DESCRIZIONE_ALLERGENE, EAN_BOX)
VALUES ('Lattosio', 'Presenza di lattosio', '8001234567890');
INSERT INTO COMPOSIZIONE_BOX ( EAN_BOX, EAN_INGREDIENTE) VALUES ('8001234567890', '1234567890123');
INSERT INTO COMPOSIZIONE_BOX ( EAN_BOX, EAN_INGREDIENTE) VALUES ('8001234567890', '1234567890456');
INSERT INTO COMPOSIZIONE_BOX ( EAN_BOX, EAN_INGREDIENTE) VALUES ('8001234567890', '1234567890789');
INSERT INTO COMPOSIZIONE_BOX ( EAN_BOX, EAN_INGREDIENTE) VALUES ('8001234567890', '1234567890890');
INSERT INTO COMPOSIZIONE_BOX ( EAN_BOX, EAN_INGREDIENTE) VALUES ('8001234567890', '1234567890125');
INSERT INTO INGREDIENTE_MAGAZZINO (DATA_INGRESSO, LOTTO, QUANTITA_INGREDIENTE, EAN_INGREDIENTE, NOME_MAG) VALUES (TO_DATE('2024-11-02', 'YYYY-MM-DD'), 'L1072', 300, '9012345670123', 'Magazzino Centrale');
INSERT INTO INGREDIENTE_MAGAZZINO (DATA_INGRESSO, LOTTO, QUANTITA_INGREDIENTE, EAN_INGREDIENTE, NOME_MAG) VALUES (TO_DATE('2024-11-01', 'YYYY-MM-DD'), 'L1001', 500, '1234567890123', 'Magazzino Centrale');
INSERT INTO INGREDIENTE_MAGAZZINO (DATA_INGRESSO, LOTTO, QUANTITA_INGREDIENTE, EAN_INGREDIENTE, NOME_MAG) VALUES (TO_DATE('2024-11-02', 'YYYY-MM-DD'), 'L1002', 300, '1234567890456', 'Magazzino Centrale');
INSERT INTO INGREDIENTE_MAGAZZINO (DATA_INGRESSO, LOTTO, QUANTITA_INGREDIENTE, EAN_INGREDIENTE, NOME_MAG) VALUES (TO_DATE('2024-11-03', 'YYYY-MM-DD'), 'L1003', 450, '1234567890789', 'Magazzino Centrale');
INSERT INTO INGREDIENTE_MAGAZZINO (DATA_INGRESSO, LOTTO, QUANTITA_INGREDIENTE, EAN_INGREDIENTE, NOME_MAG) VALUES (TO_DATE('2024-11-03', 'YYYY-MM-DD'), 'L1067', 450, '3456789018901', 'Magazzino Centrale');
INSERT INTO INGREDIENTE_MAGAZZINO (DATA_INGRESSO, LOTTO, QUANTITA_INGREDIENTE, EAN_INGREDIENTE, NOME_MAG) VALUES (TO_DATE('2024-11-03', 'YYYY-MM-DD'), 'L1997', 450, '3456789019012', 'Magazzino Centrale');
INSERT INTO SCONTI (ID_SCONTO, VALORE, INIZIO_SCONTO, FINESCONTO)
VALUES (1, 10, TO_DATE('2024-11-01', 'YYYY-MM-DD'), TO_DATE('2025-02-15', 'YYYY-MM-DD'));
 INSERT INTO SCONTI (ID_SCONTO, VALORE, INIZIO_SCONTO, FINESCONTO)
```

```
VALUES (2, 15, TO_DATE('2024-12-01', 'YYYY-MM-DD'), TO_DATE('2025-02-20', 'YYYY-MM-DD'));
INSERT INTO SCONTI (ID_SCONTO, VALORE, INIZIO_SCONTO, FINESCONTO)
VALUES (3, 20, TO_DATE('2024-11-10', 'YYYY-MM-DD'), TO_DATE('2025-03-05', 'YYYY-MM-DD'));
INSERT INTO CARRELLO(CF,EAN_BOX,QUANTITA_IN_CARRELLO) values (' RSSMRC85M01H501X ','8001234567890',1);
INSERT INTO
ORDINE(ID_ORDINE,DATA_ORDINE,TOTALE_PREZZO,TOTALE_QUANTITA,STATO_ORDINE,CF,ID_SCONTO)values('RSSMRC85M01H501X2024
1120011759',TO_DATE('2024-11-20', 'YYYY-MM-DD'),58.5,4,1,'RSSMRC85M01H501X',1);
INSERT INTO
DETTAGLI_ORDINE(ID_ORDINE,EAN_BOX,QUANTITA_BOX_DETTAGLI,PREZZO_BOX_DETTAGLI)values('RSSMRC85M01H501X2024112001175
9',8001234567890,1,12.5);
INSERT INTO FATTURA(METODO_PAGAMENTO,DATA_PAGAMENTO,IMPORTO,ID_ORDINE)values('carta',TO_DATE('2024-11-20', 'YYYY-MM-
DD'),52.65,'RSSMRC85M01H501X20241120011759');
INSERT INTO
SPEDIZIONE(CORRIERE,STATO_SPEDIZIONE,ID_ORDINE,VIA_SPEDIZIONE,CIVICO_SPEDIZIONE,CAP_SPEDIZIONE,CITTA_SPEDIZIONE,PROVI
NCIA_SPEDIZIONE)values('Bartolini',1,'RSSMRC85M01H501X20241120011759','Via Roma','15',100,'Roma','RM');
```

Funzioni

```
CREATE OR REPLACE FUNCTION calcola_importo_scontato(
  prezzo_iniziale   IN NUMBER,
  sconto_percentuale IN NUMBER
)
RETURN NUMBER
IS
  prezzo_finale NUMBER;
BEGIN
  -- Verifica se lo sconto è valido (deve essere compreso tra 0 e 100)
  IF sconto_percentuale < 0 OR sconto_percentuale > 100 THEN
    RETURN NULL;
  ELSE
    -- Calcolo del prezzo finale applicando lo sconto
    prezzo_finale := prezzo_iniziale - (prezzo_iniziale * (sconto_percentuale / 100));
    RETURN prezzo_finale;
  END IF;
END;
```

La funzione **calcola_importo_scontato** serve a calcolare il prezzo finale di un prodotto o servizio applicando uno sconto percentuale sul prezzo iniziale.
Per esempio, se un prodotto costa 100 euro e viene applicato uno sconto del 20%, la funzione calcolerà e restituirà 80 euro come prezzo finale. Se lo sconto inserito è maggiore di 100 o minore di 0, la funzione restituirà NULL, segnalando un errore.

```
CREATE OR REPLACE FUNCTION Genera_Id_Ordine(p_cf VARCHAR2, p_data DATE)
RETURN VARCHAR2
IS
 v_id_ordine VARCHAR2(50);
BEGIN
 v_id_ordine := p_cf || TO_CHAR(p_data, 'YYYYMMDDHHMISS');
 RETURN v_id_ordine;
 END;
```

La funzione **Genera_Id_Ordine** serve a generare un identificatore unico per un ordine, combinando il **codice fiscale** del cliente (p_cf) e la **data e ora** dell'ordine (p_data). L'ID generato è una concatenazione del codice fiscale del cliente e della data e ora dell'ordine in un formato specifico. Questa funzione è utile in un sistema di gestione degli ordini dove è necessario generare un ID univoco per ogni ordine. Questo tipo di ID può essere utilizzato per tracciare facilmente ogni ordine in modo univoco, garantendo che due ordini dello stesso cliente, anche se effettuati lo stesso giorno, abbiano ID distinti grazie all'inclusione dell'orario preciso.

Trigger

```
CREATE OR REPLACE TRIGGER PREZZO_UNITARIO_CARRELLO
BEFORE INSERT
ON CARRELLO
FOR EACH ROW
BEGIN
 -- Ricerca del prezzo del box per l'EAN specificato
 SELECT Prezzo_box
INTO :NEW.PREZZO_BOX_UNITARIO_CARRELLO
FROM BOX
WHERE Ean_box = :NEW.Ean_box;

EXCEPTION
WHEN NO_DATA_FOUND THEN
  -- Se l'EAN_BOX non esiste nella tabella BOX
  RAISE_APPLICATION_ERROR(-20000, 'Errore: EAN della box non trovato nella tabella BOX.');

 WHEN TOO_MANY_ROWS THEN
  -- Se la query restituisce più di un risultato (dovrebbe essere un errore di integrita dei dati)
  RAISE_APPLICATION_ERROR(-20001, 'Errore: EAN della box non è univoco nella tabella BOX.');

 WHEN OTHERS THEN
  -- Gestione di eventuali altri errori con un messaggio generico
  RAISE_APPLICATION_ERROR(-20099, 'Errore imprevisto: ' || SQLERRM);
 END;
 END PREZZO_UNITARIO_CARRELLO;
```

Il trigger **PREZZO_UNITARIO_CARRELLO** è progettato per **automatizzare l'inserimento del prezzo unitario di un prodotto nel carrello**, prelevandolo dalla tabella BOX prima che avvenga un'operazione di inserimento sulla tabella CARRELLO.

```
CREATE OR REPLACE TRIGGER CONTROLLO_DISPONIBILITA_BOX
before insert
on CARRELLO
for each row
DECLARE
BOX_DISPONIBILI NUMBER;
box_non_disp EXCEPTION;
box_ins EXCEPTION;
ean_non_valido EXCEPTION;
cf_non_valido EXCEPTION;
ean_count NUMBER;
BEGIN
-- Controlla se l'EAN fornito esiste
SELECT COUNT(*)
INTO ean_count
FROM box
WHERE EAN_BOX = :NEW.EAN_BOX;

IF ean_count = 0 THEN
 RAISE ean_non_valido;
 else
 SELECT QUANTITA_IN_BOX
 INTO BOX_DISPONIBILI
 FROM box
 WHERE :NEW.EAN_BOX = EAN_BOX;
  -- Se BOX_DISPONIBILI è nullo, l'EAN non esiste
 IF BOX_DISPONIBILI = 0  THEN
    RAISE box_non_disp;
 ELSIF :NEW.QUANTITA_IN_CARRELLO > BOX_DISPONIBILI THEN
    RAISE box_ins;
 END IF;
 END IF;

 -- Controllo se il CF fornito è valido (esiste nella tabella cliente)
DECLARE
 v_count NUMBER;
 BEGIN
 SELECT COUNT(*)
 INTO v_count
 FROM cliente
 WHERE cf = :NEW.CF;

 IF v_count = 0 THEN
    RAISE cf_non_valido;
 END IF;
 END;

 EXCEPTION
 WHEN ean_non_valido THEN
  RAISE_APPLICATION_ERROR(-20002, 'Box non trovato per l EAN fornito.');
 WHEN box_non_disp THEN
```

```
RAISE_APPLICATION_ERROR(-20003, 'Il box selezionato non è disponibile al momento.');
WHEN box_ins THEN
RAISE_APPLICATION_ERROR(-20004, 'QuantitA insufficiente del box selezionato.');
WHEN cf_non_valido THEN
RAISE_APPLICATION_ERROR(-20005, 'Codice fiscale non valido o non trovato.');
WHEN OTHERS THEN
RAISE_APPLICATION_ERROR(-20099, 'Errore imprevisto: ' || SQLERRM);
END;
```

Il trigger **CONTROLLO_DISPONIBILITA_BOX** è progettato per garantire l'integrità dei dati e la coerenza durante l'inserimento di un nuovo elemento nel carrello, effettuando controlli sulla disponibilità del box e sulla validità del codice fiscale del cliente.
Si attiva **prima di un'operazione di INSERT** sulla tabella CARRELLO

```
CREATE OR REPLACE TRIGGER PREVENZIONE_CANCELLAZIONE_CLIENTE
BEFORE DELETE
ON CLIENTE
FOR EACH ROW
DECLARE
v_count NUMBER;
non_eliminare EXCEPTION;
BEGIN
 -- Verifica se il cliente ha ordini associati
 SELECT COUNT(*)
 INTO v_count
 FROM ORDINE
 WHERE Cf = :OLD.Cf;

 IF v_count > 0 THEN
     RAISE non_eliminare;
 END IF;

 EXCEPTION
 WHEN non_eliminare THEN
  -- Se esistono ordini associati, solleva un'eccezione per impedire l'eliminazione
  RAISE_APPLICATION_ERROR(-20006, 'Non puoi eliminare un cliente con ordini associati.');
 WHEN OTHERS THEN
  -- Gestione di errori imprevisti
  RAISE_APPLICATION_ERROR(-20099, 'Errore durante la verifica degli ordini associati al cliente: ' ||
SQLERRM);
 END PREVENZIONE_CANCELLAZIONE_CLIENTE;
```

Il trigger **PREVENZIONE_CANCELLAZIONE_CLIENTE** è progettato per prevenire l'eliminazione di un cliente dal database se ha ordini associati. Questo garantisce l'integrità referenziale e la coerenza del database, evitando di lasciare ordini "orfani" privi di un cliente associato.

```
CREATE OR REPLACE TRIGGER CANCELLAZIONE_ORDINE
 AFTER UPDATE
 ON ORDINE
 FOR EACH ROW
 BEGIN
  -- Controllo se lo stato dell'ordine è passato a 4 (cancellato)
  IF :NEW.Stato_ordine = 4 AND :OLD.Stato_ordine != 4 THEN
    -- Aggiorna le quantità dei box in base alla cancellazione dell'ordine
    UPDATE BOX
    SET Quantita_in_box = Quantita_in_box +
      (SELECT SUM(do.QUANTITA_BOX_DETTAGLI)
       FROM DETTAGLI_ORDINE do
       WHERE do.Ean_box = BOX.Ean_box
        AND do.Id_ordine = :NEW.Id_ordine)
    WHERE EXISTS (SELECT 1
            FROM DETTAGLI_ORDINE do
            WHERE do.Ean_box = BOX.Ean_box
             AND do.Id_ordine = :NEW.Id_ordine);
END IF;
EXCEPTION
WHEN OTHERS THEN
RAISE_APPLICATION_ERROR(-20099,
 'Errore imprevisto durante l'aggiornamento della quantità dei box: ' || SQLERRM);
END;
```

Il trigger **CANCELLAZIONE_ORDINE** ha lo scopo di gestire automaticamente il ripristino delle quantità disponibili dei box nel magazzino ogni volta che un ordine viene cancellato. Si attiva **dopo un aggiornamento** della tabella ORDINE, controllando se lo stato dell'ordine (STATO_ORDINE) viene impostato su *4* (cancellato).

```
CREATE OR REPLACE TRIGGER INIZIALIZZA_SPEDIZIONE
AFTER INSERT
ON ORDINE
FOR EACH ROW
DECLARE
viaCF VARCHAR2(100);
civicoCF VARCHAR2(6);
capCF NUMBER(5);
cittaCF VARCHAR2(100);
provinciaCF VARCHAR2(4);
CF_VUOTO EXCEPTION;
via_VUOTO EXCEPTION;
civico_VUOTO EXCEPTION;
cap_VUOTO EXCEPTION;
citta_VUOTO EXCEPTION;
provincia_VUOTO EXCEPTION;


BEGIN
```

```
-- Controllo che il CF non sia nullo o vuoto
 IF :NEW.CF IS NULL OR :NEW.CF = '' THEN
  RAISE CF_VUOTO;
 END IF;
   -- Recupero dell'indirizzo attivo per il cliente
  SELECT via, civico, cap, citta, provincia
  INTO viaCF, civicoCF, capCF, cittaCF, provinciaCF
  FROM RUBRICA_INDIRIZZI
  WHERE CF = :NEW.CF AND STATO = 'attivo';

  -- Controllo che i valori dell'indirizzo siano validi
  IF viaCF IS NULL OR TRIM(viaCF) = '' THEN
     RAISE via_VUOTO;
  END IF;

  IF civicoCF IS NULL OR TRIM(civicoCF) = '' THEN
     RAISE civico_VUOTO;
  END IF;

  IF capCF IS NULL OR capCF <= 0 THEN
     RAISE cap_VUOTO;
  END IF;

  IF cittaCF IS NULL OR TRIM(cittaCF) = '' THEN
     RAISE citta_VUOTO;
  END IF;

  IF provinciaCF IS NULL OR TRIM(provinciaCF) = '' THEN
     RAISE provincia_VUOTO;
  END IF;

  -- Inserimento della spedizione con i dettagli recuperati
  INSERT INTO SPEDIZIONE(corriere, stato_spedizione, id_ordine, VIA_SPEDIZIONE, CIVICO_SPEDIZIONE,
  CAP_SPEDIZIONE, CITTA_SPEDIZIONE, PROVINCIA_SPEDIZIONE)
  VALUES ('Bartolini', 1, :NEW.ID_ORDINE, viaCF, civicoCF, capCF, cittaCF, provinciaCF);

 EXCEPTION
  WHEN CF_VUOTO THEN
     RAISE_APPLICATION_ERROR(-20008, 'Errore: Il Codice Fiscale non può essere nullo o vuoto.');
  WHEN via_VUOTO THEN
     RAISE_APPLICATION_ERROR(-20009, 'Errore: La via non può essere nulla o vuota.');
  WHEN civico_VUOTO THEN
     RAISE_APPLICATION_ERROR(-20010, 'Errore: Il civico non può essere nullo o vuoto.');
  WHEN cap_VUOTO THEN
     RAISE_APPLICATION_ERROR(-20011, 'Errore: Il CAP deve essere un numero valido maggiore di zero.');
  WHEN citta_VUOTO THEN
     RAISE_APPLICATION_ERROR(-20012, 'Errore: La città non può essere nulla o vuota.');
  WHEN provincia_VUOTO THEN
     RAISE_APPLICATION_ERROR(-20013, 'Errore: La provincia non può essere nulla o vuota.');
  WHEN NO_DATA_FOUND THEN
     -- Nessun indirizzo attivo trovato per il cliente
     RAISE_APPLICATION_ERROR(-20014, 'Errore: Nessun indirizzo attivo trovato per il cliente con CF ' || :NEW.CF);
  WHEN TOO_MANY_ROWS THEN
     -- Più di un indirizzo attivo trovato per il cliente
     RAISE_APPLICATION_ERROR(-20015, 'Errore: Più di un indirizzo attivo trovato per il cliente con CF ' || :NEW.CF);
  WHEN OTHERS THEN
     -- Gestione di errori imprevisti
```

```
RAISE_APPLICATION_ERROR(-20099, 'Errore imprevisto durante l''inizializzazione della spedizione: ' || SQLERRM);
```

- END INIZIALIZZA_SPEDIZIONE;

Il trigger INIZIALIZZA_SPEDIZIONE automatizza l'inserimento dei dati di spedizione nella tabella SPEDIZIONE subito dopo l'inserimento di un nuovo ordine nella tabella ORDINE. È un meccanismo progettato per recuperare automaticamente l'indirizzo del cliente dalla tabella RUBRICA_INDIRIZZI e popolare i campi della spedizione con i relativi dettagli.
Evita la necessità di creare manualmente una voce nella tabella SPEDIZIONE per ogni nuovo ordine, migliorando l'efficienza del sistema e recupera in modo sistematico i dettagli dell'indirizzo e impedisce che vengano utilizzati dati incompleti o non validi.

```
CREATE OR REPLACE TRIGGER verifica_registrazione_clienti
BEFORE INSERT ON cliente
FOR EACH ROW
DECLARE
  cf_duplicato EXCEPTION;
  email_duplicata EXCEPTION;
  cf_count INTEGER;
  email_count INTEGER;
BEGIN
  -- Verifica l'unicità del CF
  SELECT COUNT(*) INTO cf_count
  FROM cliente
  WHERE cf = :NEW.cf;

  IF cf_count > 0 THEN
    RAISE cf_duplicato;
  END IF;

  -- Verifica l'unicità dell'email
  SELECT COUNT(*) INTO email_count
  FROM cliente
  WHERE email = :NEW.email;

  IF email_count > 0 THEN
    RAISE email_duplicata;
  END IF;

EXCEPTION
  WHEN cf_duplicato THEN
    RAISE_APPLICATION_ERROR(-20022, 'Errore: il codice fiscale inserito è già registrato.');
  WHEN email_duplicata THEN
    RAISE_APPLICATION_ERROR(-20018, 'Errore: l''email inserita è già registrata.');
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20099, 'Errore sconosciuto: ' || SQLERRM);
END;
```

Il trigger **verifica_registrazione_clienti** è progettato per garantire l'unicità del codice fiscale (cf) e dell'email nella tabella cliente prima di inserire un nuovo record nella tabella cliente.

Viste

```
CREATE or replace VIEW storico_ordini_cliente AS

SELECT o.cf AS
cf_cliente,o.id_ordine,d.ean_box,d.QUANTITA_BOX_DETTAGLI,d.PREZZO_BOX_DETTAGLI,o.data_
ordine, o.totale_prezzo,o.totale_quantita,o.stato_ordine,o.id_sconto

FROM ordine o

JOIN dettagli_ordine d ON o.id_ordine = d.id_ordine;
```

La vista **storico_ordini_cliente** è progettata per fornire una panoramica degli ordini effettuati dai clienti, unendo i dettagli della tabella ordine con quelli della tabella dettagli_ordine.

```
CREATE or replace VIEW box_senza_lattosio AS

SELECT b.EAN_BOX, b.NOME_BOX, b.CATEGORIA_BOX, b.PORZIONI, b.QUANTITA_IN_BOX,
b.PREZZO_BOX

FROM box b

WHERE b.EAN_BOX NOT IN (

   SELECT EAN_BOX

   FROM Allergeni

   WHERE UPPER(NOME_ALLERGENE) = 'LATTOSIO'

);
```

La **vista box_senza_lattosio** è progettata per fornire un elenco dei box che non contengono l'allergene "lattosio".

```
CREATE or replace VIEW box_alto_contenuto_proteine AS

SELECT distinct b.EAN_BOX, b.NOME_BOX, b.CATEGORIA_BOX,
b.PORZIONI,v.PROTEINE,b.QUANTITA_IN_BOX, b.PREZZO_BOX

FROM box b

JOIN valori_nutrizionali v ON b.EAN_BOX = v.EAN_BOX

WHERE v.PROTEINE > 10;
```

La vista **box_alto_contenuto_proteine** elenca i box che contengono più di 10 grammi di proteine, basandosi sui dati della tabella valori_nutrizionali.

```
CREATE or replace VIEW dati_cliente AS
SELECT c.CF as
    cf_cliente,c.PASSWORD_C,c.EMAIL,d.NOME,d.COGNOME,d.DATAN,d.NUMERO_TEL_1,r.STAT
    O,r.VIA,r.CIVICO,r.CAP,r.CITTA,r.PROVINCIA
FROM cliente c
JOIN RUBRICA_INDIRIZZI r ON c.CF = r.CF
JOIN DATI_ANAGRAFICI d ON c.CF = d.CF
where r.stato ='attivo';
```

La vista **dati_cliente** combina informazioni provenienti da più tabelle come rubrica_indirizzi, dati_anagrafici per fornire una panoramica completa dei dettagli di un cliente con un indirizzo attivo.

Procedure

```
CREATE OR REPLACE PROCEDURE Effettua_ordine (p_cf IN VARCHAR2,sconto in number,Metodo_pag in varchar2)
IS
   importo_da_scontare NUMBER(10,2);
   percentuale_sconto number;
   v_id_ordine VARCHAR2(50); -- ID con CF e timestamp
   v_totale_prezzo NUMBER(10,2);
   v_totale_quantita NUMBER;
   v_count NUMBER;
   v_count_sconto NUMBER;
   v_data_ordine DATE;
   v_prodotti_non_disponibili VARCHAR2(4000);
   carrello_vuoto EXCEPTION;
   quantita_insufficiente EXCEPTION;
   sconto_non_valido EXCEPTION;
BEGIN
    -- Controllo sconto
   SELECT COUNT(*)
   INTO v_count_sconto
   FROM SCONTI
   WHERE ID_SCONTO = sconto;

   IF v_count_sconto = 0 THEN
      RAISE sconto_non_valido;
   END IF;

   SELECT  VALORE
   INTO percentuale_sconto
   FROM SCONTI
   WHERE ID_SCONTO = sconto;

   -- Verifica carrello vuoto e ottiene totali
   SELECT COUNT(*), SUM(Quantita_in_carrello * PREZZO_BOX_UNITARIO_CARRELLO), SUM(Quantita_in_carrello)
   INTO v_count, v_totale_prezzo, v_totale_quantita
   FROM carrello
   WHERE cf = p_cf;

   IF v_count = 0 THEN
      RAISE carrello_vuoto;
   END IF;

   -- Ottieni data corrente
   v_data_ordine := CURRENT_DATE;
```

```
-- Genera ID ordine con timestamp
v_id_ordine := Genera_Id_Ordine(p_cf, v_data_ordine);

-- Blocco delle righe della tabella box
FOR r IN (SELECT c.Ean_box, c.Quantita_in_carrello AS richiesta
        FROM carrello c
        WHERE c.cf = p_cf)
LOOP
   -- Tentativo di bloccare la riga corrispondente nella tabella box
   UPDATE box b
   SET quantita_in_box = quantita_in_box - r.richiesta
   WHERE b.Ean_box = r.Ean_box
   AND b.quantita_in_box >= r.richiesta;

   -- Verifica se il numero di righe aggiornate è zero
   IF SQL%ROWCOUNT = 0 THEN
      v_prodotti_non_disponibili := v_prodotti_non_disponibili ||
         r.Ean_box || ' (richiesti: ' || r.richiesta || '); ';
   END IF;
END LOOP;

-- Se ci sono prodotti non disponibili, genera un errore
IF v_prodotti_non_disponibili IS NOT NULL THEN
   RAISE quantita_insufficiente;
END IF;


-- Inserisce l'ordine e ottiene l'ID
INSERT INTO ordine (
   id_ordine,Data_ordine, Totale_prezzo, Totale_quantita, Stato_ordine, Cf,ID_SCONTO
) VALUES (
   v_id_ordine,
   CURRENT_DATE,
   NVL(v_totale_prezzo, 0),
   NVL(v_totale_quantita, 0),
   1,
   p_cf,
   sconto
) RETURNING Id_ordine,TOTALE_PREZZO INTO v_id_ordine,importo_da_scontare;

-- Inserisce i dettagli e pulisce il carrello in un'unica transazione
INSERT INTO dettagli_ordine (
   Id_ordine, Ean_box, QUANTITA_BOX_DETTAGLI,PREZZO_BOX_DETTAGLI
)
SELECT v_id_ordine, Ean_box, Quantita_in_carrello, PREZZO_BOX_UNITARIO_CARRELLO
FROM carrello
WHERE cf = p_cf;

DELETE FROM carrello WHERE cf = p_cf;
```

```
  --Creazione fattura
   insert into FATTURA(
        metodo_pagamento, data_pagamento, importo, id_ordine
   ) values(
        Metodo_pag,
        current_date,
        CALCOLA_IMPORTO_SCONTATO(importo_da_scontare ,percentuale_sconto),
        v_id_ordine
  );
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Ordine Id:' || v_id_ordine ||' Cliente: ' || p_cf || ' Completato con successo.');

EXCEPTION
  WHEN sconto_non_valido THEN
    RAISE_APPLICATION_ERROR(-20025, 'Sconto non valido ');
  WHEN carrello_vuoto THEN
    RAISE_APPLICATION_ERROR(-20026, 'Il carrello è vuoto');
  WHEN quantita_insufficiente THEN
    ROLLBACK;
    RAISE_APPLICATION_ERROR(-20004, 'Quantità insufficiente per i seguenti prodotti: ' || v_prodotti_non_disponibili);
  WHEN OTHERS THEN
     ROLLBACK;
     RAISE_APPLICATION_ERROR(-20099, 'Errore durante l''elaborazione dell''ordine: ' || SQLERRM);
END;
```

La procedura **Effettua_ordine** è progettata per gestire l'elaborazione di un ordine da parte di un cliente, tenendo conto di vari aspetti, come la validazione dello sconto, la verifica della disponibilità dei prodotti e la creazione di un ordine con i dettagli corrispondenti.

Input della procedura: p_cf IN VARCHAR2: codice fiscale del cliente, sconto IN NUMBER: ID dello sconto applicato, Metodo_pag IN VARCHAR2: metodo di pagamento scelto dal cliente.

Questa procedura effettua varie operazioni:

Controllo della validità dello sconto:

- Verifica se lo sconto fornito è valido, controllando se esiste un record corrispondente nella tabella SCONTI.
- Se lo sconto non è valido, viene sollevata un'eccezione.
- **Verifica carrello vuoto:**
- Se il carrello del cliente è vuoto, viene sollevata l'eccezione carrello_vuoto.
- Se il carrello contiene articoli, vengono calcolati il totale del prezzo e la quantità.
- **Generazione dell'ID dell'ordine:**
- Viene generato un ID per l'ordine utilizzando la funzione Genera_Id_Ordine, che include il codice fiscale del cliente e la data corrente.
- **Verifica disponibilità dei prodotti:**

- Per ogni prodotto nel carrello del cliente, viene verificata la disponibilità tramite un UPDATE alla tabella box. Se la quantità disponibile non è sufficiente, il prodotto viene aggiunto alla lista di v_prodotti_non_disponibili e viene sollevata un'eccezione quantita_insufficiente.
- **Inserimento dell'ordine:**
- Se tutti i prodotti sono disponibili, viene inserito un record nella tabella ordine, comprensivo di ID, totale prezzo, stato dell'ordine e codice fiscale del cliente.
- **Inserimento dettagli ordine:**
- I dettagli dell'ordine vengono inseriti nella tabella dettagli_ordine, con le informazioni sui prodotti acquistati e le quantità.
- **Pulizia del carrello:**
- Dopo l'inserimento dei dettagli, il carrello del cliente viene svuotato.
- **Creazione della fattura:**
- Una fattura viene creata con il metodo di pagamento, la data di pagamento, l'importo totale scontato e l'ID dell'ordine. L'importo scontato viene calcolato utilizzando la funzione CALCOLA_IMPORTO_SCONTATO.
- **Commit e conclusione:**
- Viene eseguito il COMMIT per confermare tutte le operazioni

```
CREATE OR REPLACE PROCEDURE mostra_composizione_box (
  p_ean_box IN VARCHAR2
)
IS
  ean_box_non_trovato EXCEPTION;
  -- Variabile per la verifica dell'esistenza dell'EAN della box
  v_box_count NUMBER;

  -- Definizione del cursore per gli ingredienti
  CURSOR c_ingredienti IS
    SELECT DISTINCT i.NOME_INGREDIENTE
    FROM COMPOSIZIONE_BOX cb
    JOIN INGREDIENTE i ON cb.EAN_INGREDIENTE = i.EAN_INGREDIENTE
    WHERE cb.EAN_BOX = p_ean_box;

BEGIN
  -- Verifica se l'EAN della box esiste nella tabella COMPOSIZIONE_BOX
  SELECT COUNT(*) INTO v_box_count
  FROM COMPOSIZIONE_BOX
  WHERE EAN_BOX = p_ean_box;

  -- Se non esiste, solleva l'eccezione
  IF v_box_count = 0 THEN
    RAISE ean_box_non_trovato;
  END IF;

  -- Visualizzazione dell'EAN della box
```

```
  DBMS_OUTPUT.PUT_LINE('EAN della box: ' || p_ean_box);

  -- Ciclo sugli ingredienti della box
  FOR rec IN c_ingredienti LOOP
    DBMS_OUTPUT.PUT_LINE('Ingrediente: ' || rec.NOME_INGREDIENTE );
  END LOOP;

  -- Conferma delle operazioni
  COMMIT;

EXCEPTION
  WHEN ean_box_non_trovato THEN
    RAISE_APPLICATION_ERROR(-20016, 'Errore: la box con EAN ' || p_ean_box || ' non esiste.');
    ROLLBACK;
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20017, 'Errore: nessun ingrediente trovato per la box con EAN ' || p_ean_box);
    ROLLBACK;
 WHEN OTHERS THEN
      -- Gestione di eventuali altri errori con un messaggio generico
      RAISE_APPLICATION_ERROR(-20099, 'Errore imprevisto: ' || SQLERRM);
END mostra_composizione_box;
```

La procedura **mostra_composizione_box** ha come scopo di visualizzare gli ingredienti associati a una determinata "box" identificata dal suo EAN_BOX.

```
CREATE OR REPLACE PROCEDURE Aggiungi_Cliente(
  p_CF VARCHAR2,
  p_PASSWORD_C VARCHAR2,
  p_EMAIL VARCHAR2
) AS
  v_email_count NUMBER;
  email_esistente EXCEPTION;
BEGIN
  -- Verifica se l'email è già esistente
  SELECT COUNT(*) INTO v_email_count
  FROM CLIENTE
  WHERE EMAIL = p_EMAIL;

  IF v_email_count > 0 THEN
    RAISE email_esistente;
  ELSE
    -- Se l'email non è presente, inserisce il nuovo cliente
    INSERT INTO CLIENTE (CF, PASSWORD_C, EMAIL)
    VALUES (p_CF, p_PASSWORD_C, p_EMAIL);

    -- Conferma la transazione se l'inserimento ha successo
    COMMIT;
```

```
    END IF;

  EXCEPTION
    WHEN email_esistente THEN
      -- Annulla la transazione e segnala l'email già registrata
      ROLLBACK;
      RAISE_APPLICATION_ERROR(-20018, 'Email già registrata!');

    WHEN DUP_VAL_ON_INDEX THEN
      -- Annulla la transazione in caso di codice fiscale duplicato
      ROLLBACK;
      RAISE_APPLICATION_ERROR(-20019, 'Codice fiscale già presente nel sistema.');

    WHEN VALUE_ERROR THEN
      -- Annulla la transazione in caso di errori di valore
      ROLLBACK;
      RAISE_APPLICATION_ERROR(-20020, 'Errore nei valori inseriti: verificare la lunghezza dei campi.');

    WHEN NO_DATA_FOUND THEN
      -- Annulla la transazione in caso di dati mancanti
      ROLLBACK;
      RAISE_APPLICATION_ERROR(-20021, 'Errore durante la verifica dei dati.');

    WHEN OTHERS THEN
      -- Annulla la transazione in caso di errori non previsti
      ROLLBACK;
      RAISE_APPLICATION_ERROR(-20099,'Errore imprevisto durante l''inserimento del cliente: ' || SQLERRM);
  END Aggiungi_Cliente;
```

La procedura **Aggiungi_Cliente** gestisce l'inserimento di un nuovo cliente nel sistema, verificando prima se l'email fornita è già registrata e gestendo eventuali errori.

```
  CREATE OR REPLACE PROCEDURE Aggiungi_Indirizzo(
    p_CF VARCHAR2,
    p_VIA VARCHAR2,
    p_CIVICO VARCHAR2,
    p_CAP NUMBER,
    p_CITTA VARCHAR2,
    p_PROVINCIA VARCHAR2
  ) AS
    v_cliente_count NUMBER;
    cliente_esistente EXCEPTION;
  BEGIN
    -- Controllo se il cliente esiste nella tabella CLIENTE
    SELECT COUNT(*)
    INTO v_cliente_count
```

```
    FROM CLIENTE
    WHERE CF = p_CF;

    IF v_cliente_count = 0 THEN
        RAISE cliente_esistente;
    ELSE
        -- Disattiva tutti gli indirizzi precedenti del cliente
        UPDATE RUBRICA_INDIRIZZI
        SET STATO = 'inattivo'
        WHERE CF = p_CF AND STATO = 'attivo';

        -- Inserisce il nuovo indirizzo come "attivo"
        INSERT INTO RUBRICA_INDIRIZZI (CF, STATO, VIA, CIVICO, CAP, CITTA, PROVINCIA)
        VALUES (p_CF, 'attivo', p_VIA, p_CIVICO, p_CAP, p_CITTA, p_PROVINCIA);

        -- Conferma la transazione se tutte le operazioni sono eseguite correttamente
        COMMIT;
    END IF;

EXCEPTION
    WHEN cliente_esistente THEN
        -- Annulla la transazione e segnala che il cliente non è stato trovato
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20005, 'Cliente non trovato, impossibile aggiungere indirizzo.');

    WHEN VALUE_ERROR THEN
        -- Annulla la transazione in caso di errori di formato
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20020, 'Errore di formato: Verifica che il CAP e gli altri dati siano corretti.');

    WHEN OTHERS THEN
        -- Annulla la transazione in caso di errori non previsti
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20099, 'Errore sconosciuto: ' || SQLERRM);
END Aggiungi_Indirizzo;
```

La procedura **Aggiungi_Indirizzo** ha lo scopo di aggiungere un nuovo indirizzo per un cliente nel sistema, aggiornando il suo stato e gestendo eventuali errori.

L'indirizzo che viene inserito tramite la procedura viene impostato direttamente come "attivo", mentre i precedenti vengono posti in "inattivo".

```
CREATE OR REPLACE PROCEDURE mostra_box_da_ingrediente(p_ean_ing IN VARCHAR2)
IS
```

```
  ingrediente_non_trovato EXCEPTION;
  nessuna_box_trovata EXCEPTION;

  -- Variabili per le verifiche
  v_ingrediente_count NUMBER := 0;
  v_box_count NUMBER := 0;

  -- Cursore per selezionare i box che contengono l'ingrediente
  CURSOR c_box IS
    SELECT i.NOME_INGREDIENTE, cb.EAN_BOX
    FROM COMPOSIZIONE_BOX cb
    JOIN INGREDIENTE i ON cb.EAN_INGREDIENTE = i.EAN_INGREDIENTE
    WHERE cb.EAN_INGREDIENTE = p_ean_ing;

BEGIN
  -- Verifica se l'EAN ingrediente esiste nella tabella INGREDIENTE
  SELECT COUNT(*) INTO v_ingrediente_count
  FROM INGREDIENTE
  WHERE EAN_INGREDIENTE = p_ean_ing;

  IF v_ingrediente_count = 0 THEN
    -- Se l'ingrediente non esiste, solleva un'eccezione
    RAISE ingrediente_non_trovato;
  END IF;

  -- Verifica se l'EAN ingrediente è presente nella tabella COMPOSIZIONE_BOX
  SELECT COUNT(*) INTO v_box_count
  FROM COMPOSIZIONE_BOX
  WHERE EAN_INGREDIENTE = p_ean_ing;

  IF v_box_count = 0 THEN
    -- Se l'ingrediente esiste in INGREDIENTE ma non è in nessun box
    RAISE nessuna_box_trovata;
  END IF;

  -- Visualizzazione dell'EAN dell'ingrediente
  DBMS_OUTPUT.PUT_LINE('EAN di ingrediente: ' || p_ean_ing);
    DBMS_OUTPUT.PUT_LINE('');
    DBMS_OUTPUT.PUT_LINE('----------------------------------------');

  -- Ciclo per visualizzare tutti i box che contengono l'ingrediente
  FOR rec IN c_box LOOP
    DBMS_OUTPUT.PUT_LINE('Nome ingrediente: ' || rec.NOME_INGREDIENTE);
    DBMS_OUTPUT.PUT_LINE('EAN Box: ' || rec.EAN_BOX);


    DBMS_OUTPUT.PUT_LINE('');
    DBMS_OUTPUT.PUT_LINE('----------------------------------------');
END LOOP;
```

```
EXCEPTION
  WHEN ingrediente_non_trovato THEN
    RAISE_APPLICATION_ERROR(-20023, 'Errore: l''ingrediente con EAN ' || p_ean_ing || ' non esiste in
    INGREDIENTE.');
  WHEN nessuna_box_trovata THEN
    RAISE_APPLICATION_ERROR(-20024, 'Errore: l''ingrediente con EAN ' || p_ean_ing || ' esiste in INGREDIENTE ma
    non è presente in alcuna box.');
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20099, 'Errore sconosciuto: ' || SQLERRM);
END mostra_box_da_ingrediente;
```

La procedura **mostra_box_da_ingrediente** ha lo scopo di visualizzare tutti i box che contengono un determinato ingrediente, identificato dal suo EAN_INGREDIENTE.

```
CREATE OR REPLACE PROCEDURE AGGIUNGI_BOX_CARRELLO (P_CF IN VARCHAR2,P_EAN_BOX IN
    VARCHAR2,P_QUANTITA IN number)
 as
 v_cf_count number;
 v_ean_box_count number;
 cliente_non_esistente EXCEPTION;
 ean_box_non_esistente EXCEPTION;
BEGIN

--Controllo se cf esiste
SELECT COUNT(*)
 INTO v_cf_count
 FROM CLIENTE
 WHERE CF = P_CF;

 IF v_cf_count = 0 THEN
   RAISE cliente_non_esistente;
 END IF;

--Controllo se Ean_box esiste
SELECT COUNT(*)
 INTO v_ean_box_count
 FROM Box
 WHERE ean_box = p_EAN_BOX;

 IF v_ean_box_count = 0 THEN
   RAISE ean_box_non_esistente;
 END IF;

 --Aggiungi nel carrello
 insert into carrello(CF,EAN_BOX,QUANTITA_IN_CARRELLO)values
```

```
        (P_CF,P_EAN_BOX,P_QUANTITA);
  COMMIT;
  EXCEPTION
  WHEN cliente_non_esistente THEN
    ROLLBACK;
    RAISE_APPLICATION_ERROR(-20005, 'Cliente non esiste');
  WHEN ean_box_non_esistente THEN
    ROLLBACK;
    RAISE_APPLICATION_ERROR(-20003, 'Ean box non disponibile');
  WHEN OTHERS THEN
    ROLLBACK;
    RAISE_APPLICATION_ERROR(-20099, 'Errore sconosciuto: ' || SQLERRM);
 END AGGIUNGI_BOX_CARRELLO;
```

La procedura **AGGIUNGI_BOX_CARRELLO** ha lo scopo di aggiungere un box specifico al carrello di un cliente.

```
CREATE OR REPLACE PROCEDURE gestisci_clienti_inattivi AS
  -- Variabili contatori
  avvisi_inviati NUMBER := 0;
  clienti_cancellati NUMBER := 0;
BEGIN
  -- Inizio della transazione
  SAVEPOINT inizio_procedura;

  -- Invio avvisi ai clienti inattivi da più di un anno
  FOR cliente IN (
    SELECT CF
    FROM cliente
    WHERE NOT EXISTS (
      SELECT 1
      FROM ordine o
      WHERE o.CF = cliente.CF
    )
    AND data_registrazione <= ADD_MONTHS(SYSDATE, -12)  -- Più di un anno fa
  ) LOOP
    BEGIN
      -- Simulazione di invio avviso
      DBMS_OUTPUT.PUT_LINE('Avviso inviato al cliente ID: ' || cliente.CF);
      avvisi_inviati := avvisi_inviati + 1;
    EXCEPTION
      WHEN OTHERS THEN
```

```
        raise_application_error(-20099,'Errore durante l''invio dell''avviso al cliente ID: ' || cliente.CF || ' - ' || SQLERRM);
    END;
END LOOP;

-- Cancellazione dei clienti inattivi da più di due anni
BEGIN
    -- Cancella i record dalla tabella carrello per i clienti senza ordini e inattivi da più di 2 anni
    DELETE FROM carrello
    WHERE CF IN (
        SELECT CF
        FROM cliente
        WHERE NOT EXISTS (
            SELECT 1
            FROM ordine o
            WHERE o.CF = cliente.CF
        )
        AND data_registrazione <= ADD_MONTHS(SYSDATE, -24)  -- Più di 2 anni fa
    );

    -- Cancella dalla rubrica degli indirizzi per i clienti senza ordini e inattivi da più di 2 anni
    DELETE FROM rubrica_indirizzi
    WHERE CF IN (
        SELECT CF
        FROM cliente
        WHERE NOT EXISTS (
            SELECT 1
            FROM ordine o
            WHERE o.CF = cliente.CF
        )
        AND data_registrazione <= ADD_MONTHS(SYSDATE, -24)  -- Più di 2 anni fa
    );

    -- Cancella dai dati anagrafici per i clienti senza ordini e inattivi da più di 2 anni
    DELETE FROM dati_anagrafici
    WHERE CF IN (
        SELECT CF
        FROM cliente
        WHERE NOT EXISTS (
            SELECT 1
            FROM ordine o
            WHERE o.CF = cliente.CF
        )
        AND data_registrazione <= ADD_MONTHS(SYSDATE, -24)  -- Più di 2 anni fa
    );

    -- Cancella dalla tabella cliente per i clienti senza ordini e inattivi da più di 2 anni
    DELETE FROM cliente
    WHERE NOT EXISTS (
        SELECT 1
        FROM ordine o
```

```
      WHERE o.CF = cliente.CF
    )
    AND data_registrazione <= ADD_MONTHS(SYSDATE, -24);  -- Più di 2 anni fa

    -- Aggiorna il contatore dei clienti cancellati
    clienti_cancellati := SQL%ROWCOUNT;

    -- Se non ci sono cancellazioni
    IF clienti_cancellati = 0 THEN
      DBMS_OUTPUT.PUT_LINE('Nessun cliente inattivo da più di 2 anni da cancellare.');
    END IF;
  EXCEPTION
    WHEN OTHERS THEN
      raise_application_error(-20099,'Errore durante la cancellazione dei clienti inattivi da più di 2 anni: ' || SQLERRM);
      ROLLBACK TO inizio_procedura;
  END;

  -- Conferma delle modifiche
  COMMIT;

  -- Messaggi di riepilogo
  DBMS_OUTPUT.PUT_LINE('Numero di avvisi inviati a clienti inattivi da più di 1 anno: ' || avvisi_inviati);
  DBMS_OUTPUT.PUT_LINE('Numero di clienti inattivi da più di 2 anni cancellati: ' || clienti_cancellati);
EXCEPTION
  WHEN OTHERS THEN
    raise_application_error(-20099,'Errore generale durante l''esecuzione della procedura: ' || SQLERRM);
    ROLLBACK;
END gestisci_clienti_inattivi;
```

La procedura **gestisci_clienti_inattivi** ha l'obiettivo di gestire i clienti che non hanno effettuato ordini per un lungo periodo. In particolare, invia avvisi ai clienti inattivi da più di un anno e cancella i clienti inattivi da più di due anni, rimuovendo anche i loro dati associati.

```
CREATE OR REPLACE PROCEDURE aggiorna_sconti_scaduti IS
BEGIN
  UPDATE SCONTI
  SET VALORE = NULL
  WHERE FINESCONTO < SYSDATE AND VALORE IS NOT NULL;
END;
```

La procedura **aggiorna_sconti_scaduti** esegue l'aggiornamento dei record nella tabella SCONTI per rimuovere i valori degli sconti scaduti.

Scheduler

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB(
    job_name       => 'job_controllo_sconti',
    job_type       => 'PLSQL_BLOCK',
    job_action     => 'BEGIN aggiorna_sconti_scaduti; END;',
    start_date     => SYSTIMESTAMP,
    repeat_interval => 'FREQ=DAILY; BYHOUR=0; BYMINUTE=0; BYSECOND=0',
    enabled        => TRUE,
    comments       => 'Aggiorna sconti scaduti ogni giorno a mezzanotte'
  );
END;
```

Scheduler che effettua automaticamente il controllo_sconti una volta al giorno a mezzanotte.

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB(
    job_name => 'gestisci_clienti_inattivi_job',
    job_type => 'PLSQL_BLOCK',
    job_action => 'BEGIN gestisci_clienti_inattivi; END;',
    start_date => SYSTIMESTAMP,
    repeat_interval => 'FREQ=MONTHLY',
    enabled => TRUE
  );
END;
```

Scheduler che gestisce automaticamente i clienti_inattivi una volta al mese.

Manuale codici errore

-20000: Errore che si ottiene quando il codice Ean_box non esiste.

-20001: Errore che si ottiene quando vengono restituite troppe righe (too many rows).

-20002: Errore che si ottiene quando il codice Ean_box non è valido.

-20003: Errore che si ottiene quando il codice Ean_box non è disponibile al momento.

-20004: Errore che si ottiene quando la quantità è insufficiente per il box selezionato.

-20005: Errore che si ottiene quando il CF non è valido o non è stato trovato.

-20006: Errore che si ottiene quando non puoi eliminare l'ordine perchè è associato ad un cliente.

-20007: Errore che si ottiene quando nessun riga è stata trovata in dettagli ordine con un certo id_ordine(no data found).

-20008: Errore che si ottiene quando il CF è nullo o vuoto.

-20009: Errore che si ottiene quando il campo "Via" è vuoto.

-20010: Errore che si ottiene quando il campo "Civico" è vuoto .

-20011: Errore che si ottiene quando il campo "Cap" è vuoto.

-20012: Errore che si ottiene quando il campo "Citta" è vuoto.

-20013: Errore che si ottiene quando il campo "Provincia" è vuoto.

-20014: Errore che si ottiene quando l' Indirizzo attivo non è stato trovato.

-20015: Errore che si ottiene quando ci sono indirizzi attivi.

-20016: Errore che si ottiene quando il codice Ean_box non viene trovato.

-20017: Errore che si ottiene quando nessun ingrediente è stato trovato per un Ean_box.

-20018: Errore che si ottiene quando l'Emal è già esistente.

-20019: Errore che si ottiene quando il CF è duplicato.

-20020: Errore che si ottiene quando il Formato non è valido.

-20021: Errore che si ottiene quando ci sono dati mancanti.

-20022: Errore che si ottiene quando ci sono valori duplicati.

-20023: Errore che si ottiene quando l'Ingrediente non è stato trovato.

-20024: Errore che si ottiene quando l'ingrediente esiste ma non è presente in un box.

-20025: Errore che si ottiene quando lo sconto non è valido.

-20026: Errore che si ottiene quando il carrello vuoto.

-20099: Errore che si ottiene quando avviene un tipo di errore non previsto(others).