Tipografia-Stampa

Studenti: Alessandro Massadoro, Riccardo Andrea Spinosa

Matricole: 0124002450,0124002253

Sintesi

Il progetto è una richiesta reale,ma solo un prototipo. E' stato effettuato un colloquio presso la cartolibreria spaccanapoli per poter inquadrare la richiesta e cimentarsi nella progettazione per il superamento dell'esame universitario Tecnologie-Web. Da tale colloquio è emerso che:

Si necessità lo sviluppo grafico front-end del sito attraverso html, css e javascript. Lo sviluppo back-end attraverso Sql per database di supporto e pyton per sviluppare algoritmi di ricerca e tutto ciò che è necessario al sito.

Il database deve gestire la vendita di articoli di un sito. I prodotti tipografici sono comprati online da dei clienti, tali prodotti verranno comprati esternamente attraverso delle api/web scripting da un altro fornitore e poi rivenduti al cliente finale.

Si necessita tener traccia degli **ordini** dei **prodotti** acquistati dal **client**e e delle sue informazioni necessarie all'acquisto. Il cliente deve necessariamente effettuare il login attraverso facebook,google o altri strumenti messi a disposizione ed una volta effettuato il login il cliente ha la possibilità di acquistare. L'utente registrato può allegare file insieme al numero ordine da personalizzare. Se l'utente risulta registrato, i dati di spedizione non verranno richiesti dato che coincideranno a quelli al momento della registrazione, viceversa se l'utente non è registrato potrà effettuare sempre l'ordine come account pubblico, che all'interno nel database sara cliente non registrato e ne verrà inserito 1 solo (regola di business). Gli studenti affrontano anche il tema Sicurezza.

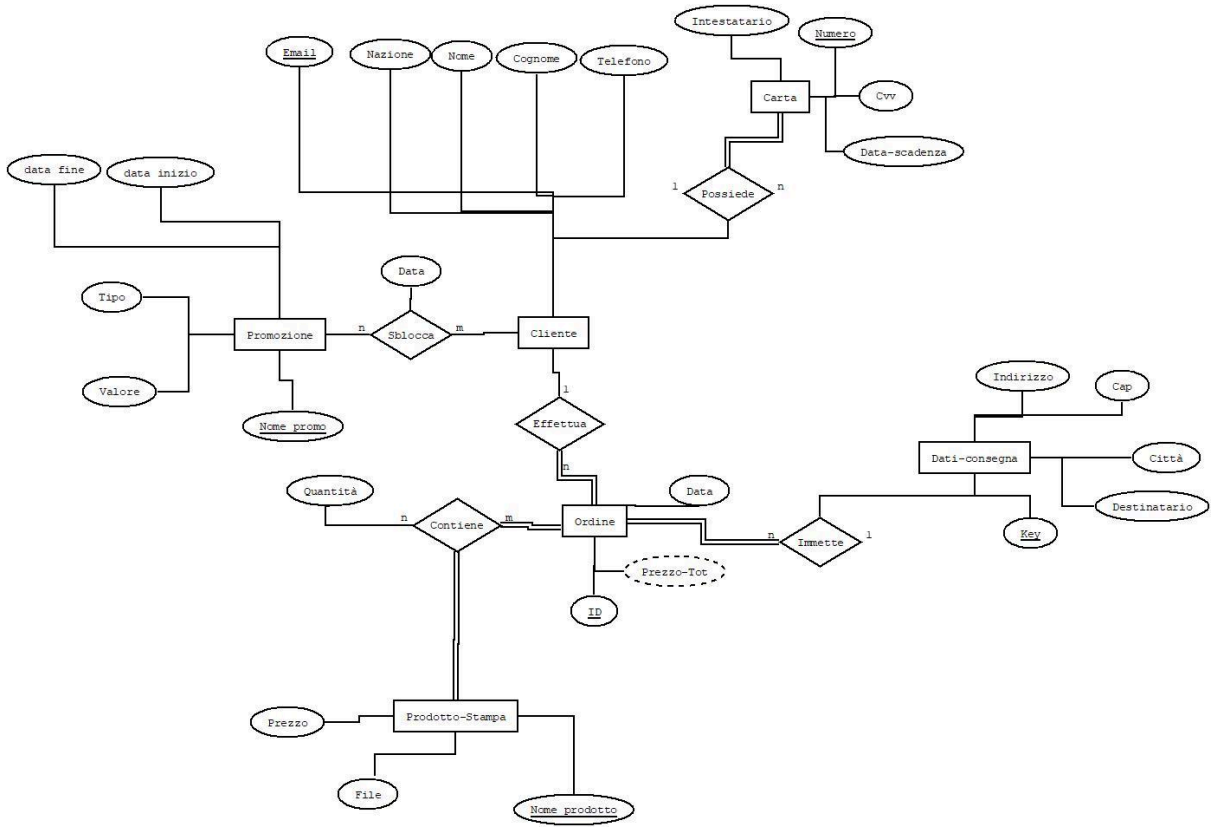
Glossario

Il glossario ha lo scopo fondamentale di chiarire il gergo tecnico usato nella descrizione dei requisiti e di evidenziare eventuali sinonimie e omonimie. In questo caso la terminologia è immediata e sono semplicemente evidenziati alcuni sinonimi.

Termine	Definizione	Sinonimi	Omonimi
Cliente	Colui che acquista	Acquirente	-
Ordini	Ordini eseguiti	-	-
Prodotti	Prodotti solo	Merce, articoli	

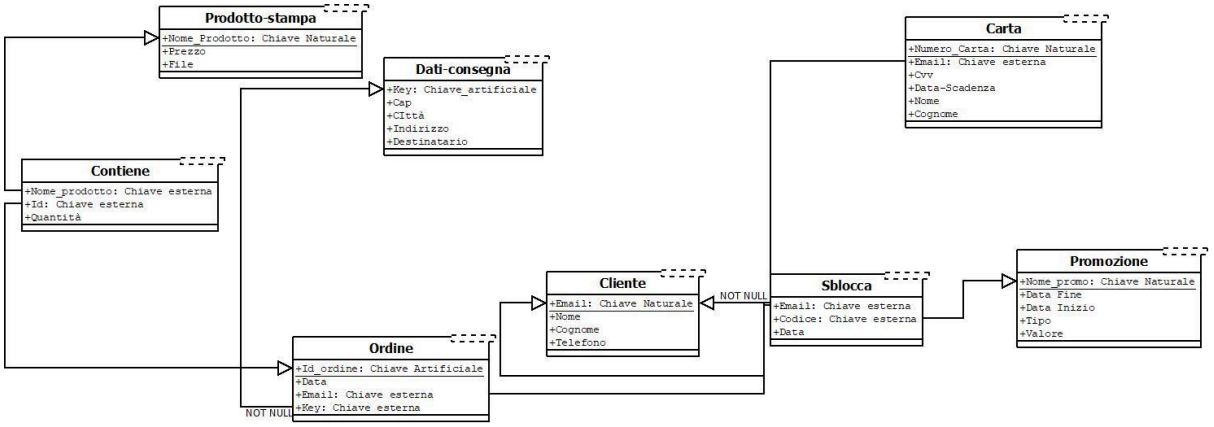
Termine	Definizione	Sinonimi	Omonimi
	acquistati		

Diagramma EE/R



Il codice di promozione risulta essere il equivalente al nome che gli assegniamo, per esempio promo10!.

Diagramma relazionale



Utenti e loro categorie

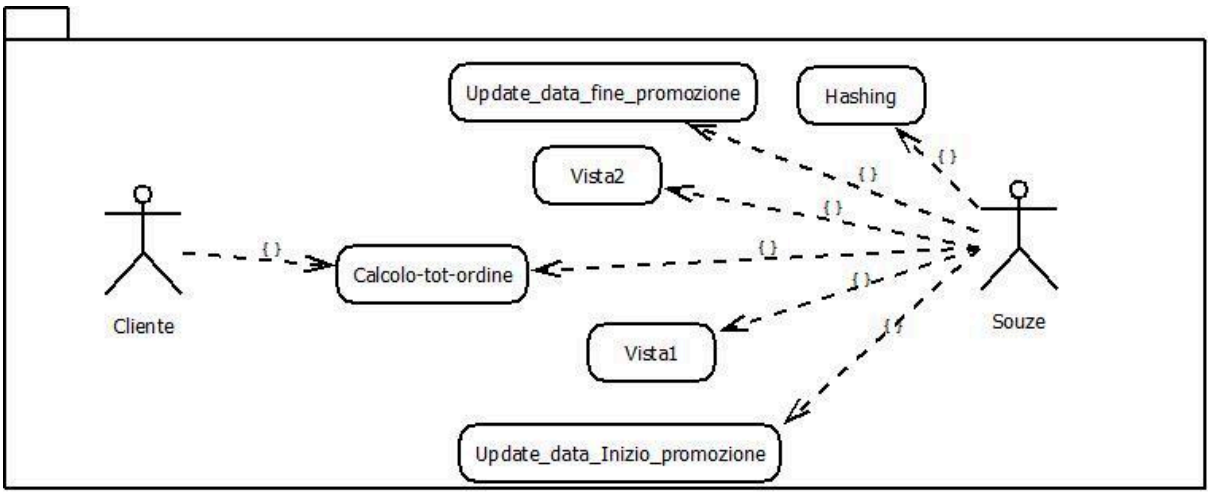
Nel nostro caso Abbiamo un amministratore di sistema che gestisce il database che ha tutti i permessi equivale a colui che esegue l'operazione di vendita alla richiesta del cliente ed n clienti che possono accedere mediante propri dati e registrarsi oppure accedere come utente

Utente	Tipo	Volume	Permessi
Souze	**Amministratore**	**1**	**All**
Cliente	**Comune**	**N**	**Calcolo-tot-ordine**

Operazione Degli Utenti

Le operazioni di base sono quelle eseguibili con un semplice comando DML. Le operazioni degli utenti sono quelle più complesse che vanno implementate tramite procedure.

Questo database si può notare che è maggiormente riversato all'amministratore del sito tipografo che è stato realizzato e non sul cliente che acquista a quello ci pensa il lato software attraverso le api ricavando tutto ciò che non si necessita tener traccia. infatti noteremo che il cliente l'unica operazione che potrà eseguire sul database da esterno è la funzione tot ordine per sapere il totale del carrelo. tutte le info rilevanti per il cliente non vengono salvate nel databse ma viene gestito solo l'essenziale utile per l'amministratore per tenere traccia di ciò che serve gli serve. tutte le info per la vendita vera è propria dei prodotti sono ricavate attraverso le API di javascript. Allora perchè non gestiamo tutto tramite software? Con il database teniamo traccia delle informazioni necessarie come gli ordini ricevuti ed i clienti registrati inoltre l'sql è nettamente più rapido ed efficiente, il restante gestito tramite software non può essere gestito direttamente con il database in quanto si necessitano delle api.



Volumi

La tavola dei volumi è rappresentata nella tabella sottostante. Oltre a riportare il numero verosimile di tuple presenti in ciascuna tabella una volta che il DB sia a regime, rappresenta anche il suo incremento atteso in un periodo di tempo prefissato. Se esiste una politica di aggiornamento dei dati per cui le tuple più vecchie sono rimosse dal DB, oppure se il numero di tuple in una relazione è più o meno costante nel tempo, allora l'incremento è nullo. In questo caso si ipotizza che un discente nella tabella ricevendo non sia mai cancellato, mentre le tuple di ricevimento e prenotazione siano conservate per un anno.

Tabella	Tipo	Volume	Incremento	Periodo
Promozione	E	4	10	Mensile
Sblocca	A	4	100	Mensile
Cliente	E	4	10	Mensile
Ordine	E	4	100	Bimestrale
Contiene	A	5	100	Bimestrale
Prodotto-stampa	E	5	300	Bimestrale
Dati-consegna	E	3	20	Mensile
Carta	E	7	20	Mensile

E sta per entità e ED per entità debole. Per le tabelle di transizione si indica una A, che sta per associazione ed ES per entità specializzate.

I dati inseriti ovvero i volumi sono solo di esempio nel database il minimo indispensabile per la fas di testing

Vincoli di integrità

Sono detti statici i vincoli che limitano i valori assumibili da alcuni attributi indipendentemente dal tempo, mentre sono detti dinamici quelli che riguardano valori che cambiano nel tempo; anche molte regole di business si presentano come vincoli dinamici e la differenza con questi ultimi è molto sottile. Poiché tutti i dati sono soggetti ad evoluzione con velocità diverse, non si può essere attaccati alla lettera di queste definizioni, né si può pretendere di produrre un elenco esaustivo dei vincoli pertinenti ad un dato contesto. Non compaiono quelli ovvi.

Statici

- Cliente prima di effettuare un ordine deve inserire i dati della carta ed i dati di consegna (**trattato con trigger**)
- A tutti gli ordini viene inserito 111 come prime tre cifre (**trattato con default**)
- Solo un cliente registrato può sbloccare una promozione(**trattato già nello schema con chiave esterna**.)
- Quando un cliente sblocca una promozione Controllo se la promozione è scaduta(**trattato con trigger**)
- Tipo di promozione può essere solo P,E,S che indica: percentuale, euro, spedizione-gratis(**trattato con check**)

- Calcolare il prezzo totale dato dal costo dei prodotti (**trattato con funzione**)

- Le data inizio e fine delle promozioni possono essere modificate così da prolungarle o anticiparle o sospenderle (**Trattato con procedura**)
- Solo prodotti appartenenti ad un ordine possono essere inseriti (**trattato con trigger**)
- La data_fine promozione non può essere minore di data_inizio promozione(**trattato con trigger per l'insert e pe update con procedura**)

Dinamici

- La data di sblocco della promozione è sempre quella odierna(**trattato con trigger**)
- La data di ordine della promozione è sempre quella odierna(**trattato con trigger**)
- Ogni tot di tempo si creano automaticamente promozioni(**trattato con scheduler**).

Verifica di normalità

Nella verifica di normalità si combatte la ridondanza attraverso la caccia alle dipendenze funzionali anomale. Mentre il rispetto della prima forma normale è in qualche modo dato per scontato, implicito nel modello relazionale, la seconda e la terza vanno verificate analizzando il significato degli attributi, poiché si tratta di una proprietà intensionale, non estensionale. Non è obbligatorio portare il database in terza forma normale ed è opportuno tenere presente che ci sono casi in cui è

preferibile per ragioni di prestazioni tenere una forma normale più bassa, in ogni caso le scelte operate devono essere motivate.

Prima forma normale

Un'entità è nella prima forma normale se non contiene gruppi ripetuti. In termini relazionali, una tabella è nella prima forma normale se non contiene colonne ripetute. Le colonne ripetute rendono i dati meno flessibili, sprecano spazio su disco e rendono più difficile la ricerca dei dati. Nel nostro caso abbiamo evitato ridondanza dei dati, per esempio tra ordine e prodotto-stampa abbiamo fatto una relazione n a m, così che i prodotti gia inseriti vengano inseriti una sola volta.

S**econda forma normale**

Tale forma risulta rispettata, Nel nostro schema non risultano chiavi multi attributo.

Terza forma normale

una relazione si dice in terza forma normale quando è innanzitutto in seconda forma normale e le colonne della tabelle sono dipendenti non transitivamente dalla chiave primaria ossia non esistono attributi che dipendono da altri attributi non-chiave. Nel nostro caso quest'ultima anche è rispettata in quanto non esistono attributi che dipendono da attributi non chiave.

Una relazione R è in forma normale di Boyce e Codd se e solo se è in terza forma normale e, per ogni dipendenza funzionale non banale, X è una superchiave per R. Quindi, dato un insieme di relazioni, non è possibile garantire sempre il raggiungimento di questa forma normale; in particolare il mancato raggiungimento di questo obiettivo è indice che la base dati è affetta da un'anomalia di cancellazione (ossia è possibile perdere dati a seguito di un'operazione di cancellazione). Tale condizione non è rispettata In quanto la cancellazione di dati potebbe creare anomalia nel database.

Possibili estensioni

Le estensioni possibili sono innumerevoli e il limite è dato solo dalla fantasia:

Scheduler che permette di eliminare le promozioni scadute ormai inutilizzate e mai rese attive da molto tempo.

Procedura che permette l'eliminazione di un cliente disiscritto.

Implementazione

Creazione utenti Il primo passo da compiere è accedere al DBMS come amministratore di sistema e creare l'utente proprietario della base di dati. E' possibile creare contestualmente anche altri utenti, sebbene essi non possano ricevere alcun privilegio di oggetto, poiché lo schema ancora non esiste.

CREATE USER SOUZE IDENTIFIED BY admin ;

GRANT ALL PRIVILEGES TO souze ; In seguito occorre disconnettersi e riconnettersi con le credenziali dell'utente souze, che è l'amministratore della nuova base di dati e che avrà i permessi per creare tutti gli oggetti che occorrono.

Creazione utenti

CREATE USER souze IDENTIFIED BY admin ; GRANT ALL PRIVILEGES TO souze ;

Data definition language

DROP TABLE CLIENTE cascade constraints;

```
DROP TABLE CARTA cascade constraints;
DROP TABLE PROMOZIONE cascade constraints;
DROP TABLE SBLOCCA cascade constraints;
DROP TABLE DATI_CONSEGNA cascade constraints;
DROP TABLE PRODOTTO cascade constraints;
DROP TABLE ORDINE cascade constraints;
DROP TABLE CONTIENE cascade constraints;
DROP SEQUENCE SEQ_DATI_CONSEGNA;
DROP SEQUENCE  SEQ_ID;


CREATE TABLE CLIENTE(
EMAIL VARCHAR(40) PRIMARY KEY,
NAZIONE VARCHAR(30) NOT NULL,
NOME VARCHAR(40),
COGNOME VARCHAR(40),
TELEFONO VARCHAR(12)
);

CREATE TABLE CARTA(
NUMERO_CARTA CHAR(16) PRIMARY KEY,
CVV CHAR(3) NOT NULL,
DATA_SCADENZA DATE NOT NULL,
EMAIL VARCHAR(40) NOT NULL,
CONSTRAINT FK_EMAIL FOREIGN KEY (EMAIL) REFERENCES CLIENTE(EMAIL)
);

CREATE TABLE PROMOZIONE(
NOME_PROMO VARCHAR(20) PRIMARY KEY,
DATA_INIZIO DATE NOT NULL,
DATA_FINE DATE NOT NULL,
```

```
VALORE NUMBER(9) NOT NULL,
TIPO CHAR(1) NOT NULL,
CONSTRAINT CHECK_TIPO CHECK(TIPO IN ('E','P','S'))
);

CREATE TABLE SBLOCCA(

EMAIL VARCHAR(40)  NOT NULL,
NOME_PROMO VARCHAR(20) NOT NULL,
DATA_SBLOCCO DATE,

CONSTRAINT PK_SBLOCCA PRIMARY KEY (EMAIL,NOME_PROMO),

CONSTRAINT FK_EMAIL2 FOREIGN KEY (EMAIL) REFERENCES CLIENTE(EMAIL),
CONSTRAINT FK_NOME_PROMO FOREIGN KEY (NOME_PROMO) REFERENCES
PROMOZIONE(NOME_PROMO)
);

CREATE TABLE DATI_CONSEGNA(
KY NUMBER(9)  PRIMARY KEY,
CAP VARCHAR(8) NOT NULL,
CITTA VARCHAR(40) NOT NULL,
INDIRIZZO VARCHAR(70) NOT NULL,
DESTINATARIO VARCHAR(70) NOT NULL

);
```

```
CREATE TABLE ORDINE(

ID_ORDINE NUMBER(9) DEFAULT 111 PRIMARY KEY,
DATA_OGGI DATE,
EMAIL VARCHAR(40) NOT NULL,
KY NUMBER (9) NOT NULL,
CONSTRAINT FK_EMAIL4 FOREIGN KEY (EMAIL) REFERENCES CLIENTE(EMAIL),
CONSTRAINT FK_KY FOREIGN KEY (KY) REFERENCES DATI_CONSEGNA (KY)
);

CREATE TABLE PRODOTTO(
NOME_PRODOTTO VARCHAR(40) PRIMARY KEY,
PREZZO NUMBER(9,3),
DOCUMENTO BLOB
);


CREATE TABLE CONTIENE(

NOME_PRODOTTO VARCHAR(40)  NOT NULL,
ID_ORDINE NUMBER(9,0) NOT NULL,
QUANTITÀ NUMBER(9,0) NOT NULL,

CONSTRAINT PK_CONTIENE PRIMARY KEY (NOME_PRODOTTO,ID_ORDINE),

CONSTRAINT FK_NOME_PRODOTTO FOREIGN KEY (NOME_PRODOTTO)
REFERENCES PRODOTTO(NOME_PRODOTTO),

CONSTRAINT FK_ID_ORDINE FOREIGN KEY (ID_ORDINE) REFERENCES
ORDINE(ID_ORDINE)
);
```

```
CREATE SEQUENCE SEQ_ID
MINVALUE 1
START WITH 1
INCREMENT BY 1
CACHE 10;

CREATE SEQUENCE SEQ_DATI_CONSEGNA
MINVALUE 1
START WITH 1
INCREMENT BY 1
CACHE 10;
```

Data manipulation language

```
INSERT INTO CLIENTE(EMAIL,NAZIONE,NOME,COGNOME,TELEFONO) VALUES
('alessandromassadoro@live.it','Italia','Alessandro','Massadoro','3803648645');

INSERT INTO CLIENTE(EMAIL,NAZIONE,NOME,COGNOME,TELEFONO) VALUES
('andreapuzza@gmail.com','Italia','Andrea','Puzza','3803622645');
```

```
INSERT INTO CLIENTE(EMAIL,NAZIONE,NOME,COGNOME) VALUES

('riccardoandreaspinosa@gmail.com','Italia','Riccardo','Spinosa');



INSERT INTO CLIENTE(EMAIL,NAZIONE) VALUES

('ilgattovirgola@live.it','Italia');



INSERT INTO DATI_CONSEGNA(KY, CAP, CITTA,DESTINATARIO, INDIRIZZO) VALUES
(SEQ_DATI_CONSEGNA.nextval,'81012','Caiazzo','ALESSANDRO','Via fontana murata');



INSERT INTO DATI_CONSEGNA(KY, CAP, CITTA,DESTINATARIO, INDIRIZZO) VALUES
(SEQ_DATI_CONSEGNA.nextval,'70015','Napoli','GENNARO','Via non lo so');



INSERT INTO DATI_CONSEGNA(KY, CAP, CITTA, DESTINATARIO,INDIRIZZO) VALUES
(SEQ_DATI_CONSEGNA.nextval,'81120','Burundi','RICCARDO','Via dei puparuoli 3');









INSERT INTO ORDINE (ID_ORDINE,DATA_OGGI,EMAIL,KY)

VALUES (SEQ_ID.nextval,'11/NOV/2022','alessandromassadoro@live.it',1);



INSERT INTO ORDINE (ID_ORDINE,DATA_OGGI,EMAIL,KY)

VALUES (SEQ_ID.nextval,'12/FEB/2021','ilgattovirgola@live.it',2);



INSERT INTO ORDINE (ID_ORDINE,EMAIL,KY)

VALUES (SEQ_ID.nextval,'andreapuzza@gmail.com',3);
```

```
INSERT INTO ORDINE (ID_ORDINE,EMAIL,KY)
VALUES (SEQ_ID.nextval,'riccardoandreaspinosa@gmail.com',1);

INSERT INTO PRODOTTO(NOME_PRODOTTO,PREZZO)
VALUES ('acqua',1);

INSERT INTO PRODOTTO(NOME_PRODOTTO,PREZZO)
VALUES ('taccuino',3);

INSERT INTO PRODOTTO(NOME_PRODOTTO,PREZZO)
VALUES ('caffe',12);

INSERT INTO PRODOTTO(NOME_PRODOTTO,PREZZO)
VALUES ('ciao',12);

INSERT INTO PRODOTTO(NOME_PRODOTTO,PREZZO)
VALUES ('blocco',32);


INSERT INTO CONTIENE(NOME_PRODOTTO,ID_ORDINE,QUANTITÀ)
VALUES ('acqua',1,12);


INSERT INTO CONTIENE(NOME_PRODOTTO,ID_ORDINE,QUANTITÀ)
VALUES ('blocco',1,4);

INSERT INTO CONTIENE(NOME_PRODOTTO,ID_ORDINE,QUANTITÀ)
VALUES ('blocco',2,10);

INSERT INTO CONTIENE(NOME_PRODOTTO,ID_ORDINE,QUANTITÀ)
```

```
VALUES ('blocco',3,10);

INSERT INTO CONTIENE(NOME_PRODOTTO,ID_ORDINE,QUANTITÀ)
VALUES ('caffe',4,10);

INSERT INTO CARTA(NUMERO_CARTA,CVV, DATA_SCADENZA, EMAIL)
VALUES('1234567891234567','123','31/MAR/2024','alessandromassadoro@live.it');

INSERT INTO CARTA(NUMERO_CARTA,CVV, DATA_SCADENZA, EMAIL)
VALUES('1234567891234569','523','31/MAR/2025','alessandromassadoro@live.it');

INSERT INTO CARTA(NUMERO_CARTA,CVV, DATA_SCADENZA, EMAIL)
VALUES('1236667891234569','523','22/NOV/2025','andreapuzza@gmail.com');

INSERT INTO CARTA(NUMERO_CARTA,CVV, DATA_SCADENZA, EMAIL)
VALUES('1236667891210569','773','01/NOV/2023','riccardoandreaspinosa@gmail.com');

INSERT INTO CARTA(NUMERO_CARTA,CVV, DATA_SCADENZA, EMAIL)
VALUES('1236667891210999','773','01/NOV/2023','riccardoandreaspinosa@gmail.com');

INSERT INTO CARTA(NUMERO_CARTA,CVV, DATA_SCADENZA, EMAIL)
VALUES('1236667891220999','773','01/NOV/2023','riccardoandreaspinosa@gmail.com');

INSERT INTO CARTA(NUMERO_CARTA,CVV, DATA_SCADENZA, EMAIL)
VALUES('1786667891220999','925','01/OCT/2023','ilgattovirgola@live.it');
```

```
INSERT INTO PROMOZIONE(NOME_PROMO,DATA_INIZIO, DATA_FINE, VALORE,TIPO)
VALUES('PROMOPROVA','01/OCT/2023','05/OCT/2023',5,'P');
INSERT INTO PROMOZIONE(NOME_PROMO,DATA_INIZIO, DATA_FINE, VALORE,TIPO)
VALUES('PROMOPROVA6','03/OCT/2023','05/OCT/2023',5,'S');

INSERT INTO PROMOZIONE(NOME_PROMO,DATA_INIZIO, DATA_FINE, VALORE,TIPO)
VALUES('PROMOPROVA2','01/OCT/2023','05/OCT/2023',5,'P');

INSERT INTO PROMOZIONE(NOME_PROMO,DATA_INIZIO, DATA_FINE, VALORE,TIPO)
VALUES('PROMOPROVA3','01/OCT/2023','05/OCT/2023',50,'E');

INSERT INTO PROMOZIONE(NOME_PROMO,DATA_INIZIO, DATA_FINE, VALORE,TIPO)
VALUES('ATTIVA','11/OCT/2022','05/OCT/2023',25,'E');
```

```
INSERT INTO SBLOCCA(NOME_PROMO,EMAIL)
VALUES('PROMOPROVA2','alessandromassadoro@live.it');

INSERT INTO SBLOCCA(NOME_PROMO,EMAIL)
VALUES('ATTIVA','riccardoandreaspinosa@gmail.com');

INSERT INTO SBLOCCA(NOME_PROMO,EMAIL)
VALUES('PROMOPROVA3','ilgattovirgola@live.it');

INSERT INTO SBLOCCA(NOME_PROMO,EMAIL)
```

VALUES('PROMOPROVA2','ilgattovirgola@live.it');

Sequence

Le sequence permettono di sviluppare sequenze di autoincremento. La sequence auto-incrementa la chiave artificiale di ordine

SEQUENCE SEQ_ID

sequenza autoincrement per ordine

CREATE SEQUENCE SEQ_ID

MINVALUE 1

START WITH 1

INCREMENT BY 1

CACHE 10;

SEQUENCE SEQ_DATI_CONSEGNA

sequenza autoincrement per dati_consegna

CREATE SEQUENCE SEQ_DATI_CONSEGNA

MINVALUE 1

START WITH 1

INCREMENT BY 1

CACHE 10;

Trigger

I triigger DML sono utili principalmente per il controllo di vincoli dinamici in fase di immissione o aggiornamento dei dati. E’ possibile cum grano salis usarli anche per implementare regole di business o per calcolare, generare o sovrascrivere il valore di alcuni campi.

Controllo_ordine

Il trigger controlla che siano stati inseriti i dati della carta e prima che l’ordine venga effettuato. Il trigger e’ utile così da non far effettuare l’ordine o meglio inserire l’ordine se mancano tali dati. il trigger inserisce all’interno delle due variabili l’email di carta dove l’email è ugualE all’elemento new che stiamo inserendo, se le due query non trovano corrispondenza restituiranno no_data_found cadendo così nell'eccezione bloccando poi l’inserimento

```
CREATE OR REPLACE TRIGGER CONTROLLO_ORDINE
BEFORE INSERT ON ORDINE
FOR EACH ROW
DECLARE
CONTROLLO_CARTA VARCHAR (40);
FOULD EXCEPTION;
BEGIN
SELECT MAX(CARTA.EMAIL) INTO CONTROLLO_CARTA FROM CARTA JOIN CLIENTE
ON CLIENTE.EMAIL = CARTA.EMAIL WHERE CLIENTE.EMAIL = :NEW.EMAIL;
IF CONTROLLO_CARTA IS NULL THEN
        RAISE FOULD;
END IF;
EXCEPTION WHEN FOULD THEN
RAISE_APPLICATION_ERROR(-20005,'ERROR CARTA NON ESISTENTE');
END CONTROLLO_ORDINE;
```

Controllo_sblocca

Il trigger controlla se la promozione che si vuole sbloccare sia scaduta, in tal caso, la promo non sarà sbloccabile.Ciò viene eseguito attraverso l'eccezione, dichiariamo una variabile scadenza ed una eccezione ecce, convertiamo la data_fine della promozione in stringa ed inseriamo essa in scadenza con un if cotrolleremo la scadenza e sysdate convertito in stringa. se scadenza è minore uguale di data odierna implica che la promo è scaduta e ricadiamo nella eccezione bloccando l'inserimento

```
CREATE OR REPLACE TRIGGER CONTROLLO_SBLOCCA

BEFORE INSERT ON SBLOCCA

FOR EACH ROW

DECLARE

SCADENZA VARCHAR(50);

ECCE EXCEPTION;

BEGIN

SELECT to_char (DATA_FINE,'YYYY-MM-DD HH24:MI:SS') INTO

SCADENZA FROM PROMOZIONE WHERE
PROMOZIONE.NOME_PROMO=:NEW.NOME_PROMO;

IF SCADENZA <= to_char (sysdate,'YYYY-MM-DD HH24:MI:SS')  THEN

RAISE ECCE;

END IF;

EXCEPTION

WHEN ECCE THEN RAISE_APPLICATION_ERROR(-20002,'PROMO SCADUTA');

END;
```

Imposta_data_sblocca

Il trigger imposta automaticamente la data della promozione sbloccata in quanto sarà sempre quella odierna ovvero quando inserita. inseriamo sysdate in una apposita variabile che poi inseriremo in data_sblocca

```
CREATE OR REPLACE TRIGGER IMPOSTA_DATA_SBLOCCA

BEFORE INSERT ON SBLOCCA

FOR EACH ROW

DECLARE

DATA_ODIERNA DATE;

BEGIN

SELECT SYSDATE INTO DATA_ODIERNA FROM DUAL;

:NEW.DATA_SBLOCCO := DATA_ODIERNA;

END;
```

Imposta_data_ordine

Il trigger imposta automaticamente la data dell'ordine effettuato in quanto sarà sempre quella odierna ovvero quando inserita. inseriamo sysdate in una apposita variabile che poi inseriremo in data_ordine

```
CREATE OR REPLACE TRIGGER IMPOSTA_DATA_ORDINE

BEFORE INSERT ON ORDINE

FOR EACH ROW

DECLARE

DATA_ODIERNA DATE;

BEGIN

SELECT SYSDATE INTO DATA_ODIERNA FROM DUAL;

:NEW.DATA_OGGI := DATA_ODIERNA;

END;
```

CONTROLLO_DATA_PROMOZIONE

Il trigger si assicura che data_fine promozione non può essere minore di data_inizio promozione. Utile per evitare errori di modifica o di inserimento

```
CREATE OR REPLACE TRIGGER CONTROLLO_DATA_PROMOZIONE

BEFORE INSERT ON PROMOZIONE

FOR EACH ROW

DECLARE

CONTROLLO_DATA_INIZIO DATE;

CONTROLLO_DATA_FINE DATE;

ECCE EXCEPTION ;

BEGIN

IF :NEW.DATA_INIZIO >  :NEW.DATA_FINE THEN

RAISE ECCE;

END IF;

EXCEPTION WHEN ECCE THEN

RAISE_APPLICATION_ERROR(-20003,'DATA_INIZIO_PROMOZIONE MAGGIORE
DI DATA_FINE_PROMOZIONE');

END;
```

FUNZIONI E PROCEDURE

Calcolo_tot_ordine

La funzione restituisce la somma dei prezzi dei prodotti acquistati di un determinato ordine. dunque prende in ingresso un ordine e con una query effettuerà la somma dei prezzo di tali prodotti dove l'ordine passato ingresso esiste. Nel caso non dovesse essere restituira no_data_foud cadendo in eccezione.

```
CREATE OR REPLACE FUNCTION CALCOLO_TOT_ORDINE (ORDINE
NUMBER)
RETURN NUMBER IS
PAGATO_TOT NUMBER(9,2);
```

```
ORDINE_INESISTENTE NUMBER(9,0);
app number (9,0);/* quantità*/
app1 number (9,3);/*prezzo*/

CURSOR c1
IS
(SELECT PREZZO,QUANTITÀ FROM ORDINE
JOIN CONTIENE ON ORDINE.ID_ORDINE =CONTIENE.ID_ORDINE
JOIN PRODOTTO ON CONTIENE.NOME_PRODOTTO =
PRODOTTO.NOME_PRODOTTO
WHERE ORDINE.ID_ORDINE = ORDINE GROUP BY PREZZO,QUANTITÀ);

BEGIN
PAGATO_TOT:=0;
SELECT ORDINE.ID_ORDINE INTO ORDINE_INESISTENTE FROM
ORDINE WHERE ORDINE.ID_ORDINE= ORDINE;


OPEN c1;
FETCH c1 INTO app,app1;


WHILE c1%FOUND LOOP
PAGATO_TOT :=PAGATO_TOT+ app1 * app;
FETCH c1 INTO app,app1;
END LOOP;
RETURN PAGATO_TOT;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20001,'ORDINE INESISTENTE');
END CALCOLO_TOT_ORDINE;
```

Update_data_fine_promozione

La procedura permette di modificare la scadenza di una promozione posticipandola o anticipandola. Verrà gestite l'eccezione in cui la promozione inserita non esista. Utile per sospendere promozioni o riattivarlE. Un trigger ci assicurerà che la data_fine non sia mai minore della data inizio quando viene inserita. La procedura gestisce l'opzione nel caso si voglia inserire una data_fine minore di data_inizio.

```
CREATE OR REPLACE PROCEDURE UPDATE_DATA_FINE_PROMOZIONE(PROMO
VARCHAR, FINE DATE)
```

```
IS

DATA_PROMO VARCHAR(20);

ECCE EXCEPTION;

BEGIN

SELECT DATA_INIZIO INTO DATA_PROMO

FROM PROMOZIONE WHERE PROMO = PROMOZIONE.NOME_PROMO;

IF DATA_PROMO> FINE THEN

RAISE ECCE;

END IF;

UPDATE PROMOZIONE SET DATA_FINE = FINE  WHERE PROMO =
PROMOZIONE.NOME_PROMO;

EXCEPTION

WHEN ECCE THEN RAISE_APPLICATION_ERROR(-10003,'ERRORE');

WHEN NO_DATA_FOUND THEN

RAISE_APPLICATION_ERROR(-20001,'NO_DATA_FOUND');

END UPDATE_DATA_FINE_PROMOZIONE;
```

Update_data_Inizio_promozione

La procedura permette di modificare l'inizio di una promozione posticipandola o anticipandola. Verrà gestite l'eccezione in cui la promozione inserita non esista. Utile per cambiare gli inizi programmati di promozioni. La procedura gestisce l'opzione nel caso si voglia inserire una data_inizio maggiore di data_fine.

```
CREATE OR REPLACE PROCEDURE
UPDATE_DATA_INIZIO_PROMOZIONE(PROMO VARCHAR, INIZIO DATE)

IS

DATA_PROMO VARCHAR(20);

ECCE EXCEPTION;

BEGIN

SELECT DATA_FINE INTO DATA_PROMO
```

```
FROM PROMOZIONE WHERE PROMO = PROMOZIONE.NOME_PROMO;

IF DATA_PROMO< INIZIO THEN

RAISE ECCE;

END IF;

UPDATE PROMOZIONE SET DATA_INIZIO = INIZIO  WHERE PROMO =
PROMOZIONE.NOME_PROMO;

EXCEPTION

WHEN ECCE THEN RAISE_APPLICATION_ERROR(-20003,'ERRORE');

WHEN NO_DATA_FOUND THEN

RAISE_APPLICATION_ERROR(-10001,'NO_DATA_FOUND');

END UPDATE_DATA_INIZIO_PROMOZIONE;
```

Hashing

QUESTA PROCEDURA PERMETTE DI INSERIRE DELLE PROMOZIONI SENZA DECIDERE ALCUN PARAMETRO. LA PROMOZIONE INIZIERà DAL GIORNO CHE VIENE INSERITA E TERMINERà 7 GIORNI DOPO, LA CHIAVE VIENE INVECE GENERATA CASUALMENTE ED ANCHE IL TIPO SARA' SCELTO A SORTE. IL VALORE SARA' CASUALE POSITIVO ED INFERIORE A 10.

```
CREATE OR REPLACE PROCEDURE HASHING

IS

STRINGA VARCHAR(20);

NUMERO NUMBER(1);

TIPO1 CHAR(1);

BEGIN

SELECT DBMS_RANDOM.STRING('z',DBMS_RANDOM.VALUE(10,10)) INTO
STRINGA FROM DUAL;

SELECT dbms_random.value(1,10) INTO NUMERO FROM DUAL;

IF NUMERO < 4 THEN

TIPO1:='E';

END IF;
```

```
IF NUMERO < 6 AND NUMERO > 3  THEN

TIPO1:='S';

END IF;

IF NUMERO < 10 AND NUMERO > 5  THEN

TIPO1:='P';

END IF;

INSERT INTO PROMOZIONE(NOME_PROMO,DATA_INIZIO, DATA_FINE,
VALORE,TIPO)

VALUES(STRINGA,SYSDATE,SYSDATE+7,NUMERO,TIPO1);

END;
```

IMPLEMENTAZIONE VISTE

Viste Le viste sono relazioni virtuali utili a regolamentare l'accesso ed aumentare la fruibilità dei dati per i vari utenti. E' possibile (e auspicabile) usare le viste per mostrare porzioni dei dati (i dati più recenti, meno recenti, quelli di una determinata area etc.), calcolare gli attributi derivati e applicare funzioni più o meno complesse.

Vista1

Tale vista vede tutti i dati che possono risultare utili insieme nel database

```
CREATE VIEW VISTA1 AS

SELECT
CLIENTE.NOME,CLIENTE.COGNOME,ORDINE.ID_ORDINE,ORDINE.DATA_OGGI
,ORDINE.KY,CLIENTE.EMAIL,SBLOCCA.NOME_PROMO,CONTIENE.NOME_PRO
DOTTO,PRODOTTO.PREZZO

FROM PRODOTTO JOIN CONTIENE ON CONTIENE.NOME_PRODOTTO =
PRODOTTO.NOME_PRODOTTO JOIN ORDINE ON CONTIENE.ID_ORDINE =
ORDINE.ID_ORDINE JOIN CLIENTE ON CLIENTE.EMAIL=ORDINE.EMAIL JOIN
SBLOCCA ON SBLOCCA.EMAIL = CLIENTE.EMAIL;
```

Vista2

Tale vista permette di vedere tutte le info riguardante gli ordini del giorno

```
CREATE VIEW VISTA2 AS

SELECT
CLIENTE.NOME,CLIENTE.COGNOME,ORDINE.ID_ORDINE,ORDINE.DATA_OGGI
,CLIENTE.EMAIL,CONTIENE.NOME_PRODOTTO,PRODOTTO.PREZZO

FROM PRODOTTO JOIN CONTIENE ON CONTIENE.NOME_PRODOTTO =
PRODOTTO.NOME_PRODOTTO JOIN ORDINE ON CONTIENE.ID_ORDINE =
ORDINE.ID_ORDINE JOIN CLIENTE ON CLIENTE.EMAIL=ORDINE.EMAIL JOIN
SBLOCCA ON SBLOCCA.EMAIL = CLIENTE.EMAIL

WHERE DATA_OGGI = SYSDATE
```

Scheduler

Lo scheduler serve a programmare azioni, normalmente periodiche, che avvengono in istanti predeterminati. Similmente ad un trigger, vale il paradigma Evento-Condizione-Azione (ECA), ma l'evento è legato allo scorrere del tempo — quindi all'orologio di sistema — piuttosto che ad un'operazione DML.

CREA_PROMO

LO SCHEDULER CREA PROMOZIONI MENSILMENTE RICHIAMANDO LA PROCEDURA HASHING

```
begin
   dbms_scheduler.create_job(job_name        => 'CREA_PROMO',
                             job_type        => 'STORED_PROCEDURE',
                             job_action      => 'HASHING',
                             start_date      => systimestamp,
                             end_date        => null,
                             repeat_interval => 'FREQ = MONTHLY ',
                             enabled         => true,
                             auto_drop       => false,
                             comments        => 'CREA PROMOZIONI MENSILMENTE');
```

end;

/

Tabella degli errori

Numero	Descrizione		
20001	INSERIMENTO FALLITO PER NO_DATA_FOUND		
20002	PROMO SCADUTA		
20003	DATA_INIZIO_PROMOZIONE MAGGIORE DI DATA_FINE_PROMOZIOE		
20004	PRODOTTO NON PRESENTE IN NESSUN ORDINE		
20005	valore null		