UNIVERSITÀ DI NAPOLI PARTHENOPE

DIPARTIMENTO DI SCIENZE E TECNOLOGIA

Laurea Triennale in Informatica

PROGETTO BASI DI DATI

GESTIONE PETSTORE

Relatore:

Prof: Antonio Maratea

Candidati:

Mattia Danisi - 0124002281

Luca Di Meglio - 0124002377

Riccardo Maione - 0124002391

Categoria:

Gestione Aziendale

Data di consegna:

22/07/2022

ANNO ACCADEMICO 2021/2022

Indice

Capitolo 1

PROGETTAZIONE

1.1 Il Database in sintesi

Il nostro database gestisce i dati relativi ad un **negozio di animali: un Petstore** (modello basato sull'azienda reale Zoomiguana).
Il fulcro del database è l'entità **animale**, di cui registriamo le entrate/uscite relative, i dati riguardanti la provenienza dell'animale e le relative adozioni (o eventuali restituzioni).
Segue l'entità **dipendente** di cui registriamo i principali dati anagrafici e di contatto e le informazioni relative agli stipendi e alle turnazioni di lavoro.
Chiude il cerchio l'entità **cliente** di cui registriamo i principali dati anagrafici e di contatto e le informazioni relative all'acquisto dei prodotti, alle visite effettuate in azienda e ovviamente le varie adozioni (e eventuali restituzioni).

Il database ha 3 scopi principali :

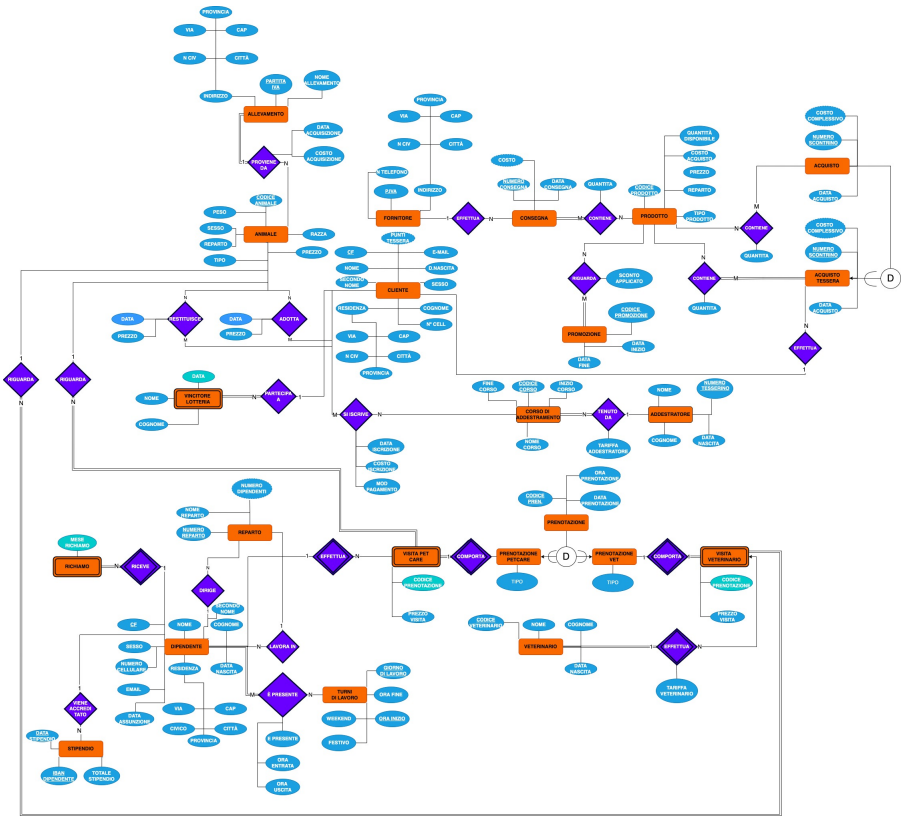
1. Registrare i movimenti relativi agli animali: acquisizioni dall'allevamento, adozioni e restituzioni
2. Gestire le informazioni relative ai beni, dipendenti e professionisti che lavorano in azienda : prodotti, dipendenti, veterinari, addestratori ecc
3. Tracciare le entrate e le uscite monetarie dell'azienda

1.2 Glossario

TERMINE	DEFINIZIONE
ALLEVAMENTO	allevamento da cui viene acquisito l'animale
ANIMALE	animale che viene reso disponibile per l'adozione
CLIENTE	cliente che adotta animale
REPARTO	reparto dove risiedono animali e prodotti
FORNITORE	fornitore che fornisce il prodotto
CONSEGNA	consegna effettuata dal fornitore
PRODOTTO	prodotto messo in vendita per la cura dell'animale
CORSO DI ADDESTRAMENTO	corso di formazione tenuto da addestratore per il cliente
ADDESTRATORE	colui che tiene il corso di addestramento
DIPENDENTE	singolo lavoratore aziendale
TURNO DI LAVORO	turno lavorativo specifico che riguarda il dipendente
PRENOTAZIONE	prenotazione utile per la visita petcare o veterinariaria, divisa in due specializzazioni
VETERINARIO	medico esterno chiamato per visitare l'animale
VISITA VETERINARIO	visita effettuata il veterinario in giorni ed orari specifici
VISITA PETCARE	visita effettuata dal dipendente per la cura dell'animale
PROMOZIONE	promozione sul prodotto

1.3 Diagramma E/R

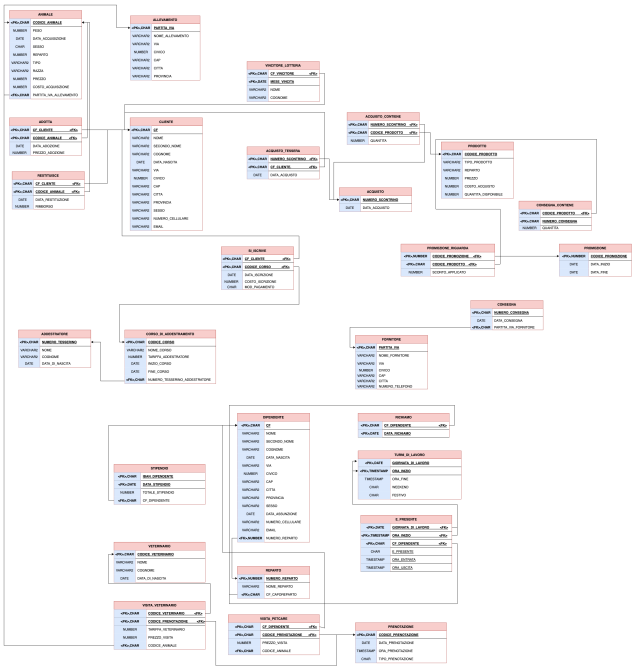
Il diagramma EE/R detto anche modello concettuale, costituisce il modello grafico su cui si basa l'intero database. Nel diagramma le entità sono collegate tramite una relazione e una molteplicità associata. Ad ogni entità sono associati dei campi detti attributi.



1.4 Modello Relazionale

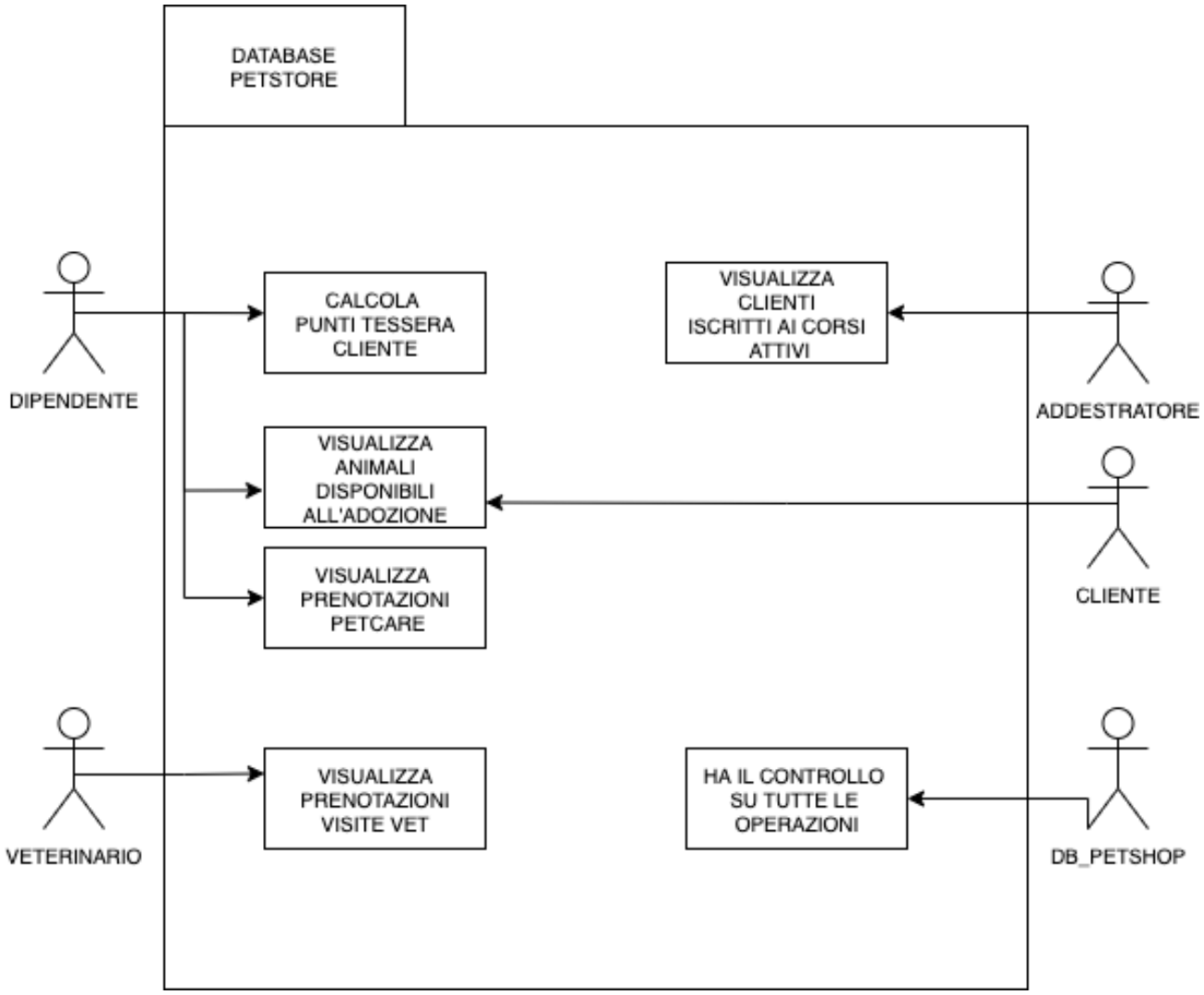
Nel modello relazionale tutto quello che è stato creato precedentemente viene trasfromato in tabelle, definendo tutti i tipi di attributi utilizzati e facendo gli opportuni collegamenti tra entità tramite chiavi esterne, rispettando le seguenti regole di molteplicità :

- Per la tradizione delle specializzazioni (due nel nostro caso) è possibile scegliere tra tre diversi metodi di approccio: specializzazione a tabella unica, partizionamento verticale e partizionamento orizzontale.
- Per le molteplicità 1-N, inseriamo la chiave primaria dell'entità (1) come chiave esterna sull' entità (N).
- Per le molteplicità 1-1, se abbiamo le totalità ad entrambi, l'inserimento della chiave esterna risulta indifferente, ma nel caso la totalità non sia di entrambi è consigliato inserire la chiave esterna dove c'è la totalità, un esempio nel nostro caso è quello che intercorre tra le due tabelle.
- Per le quali esiste una relazione M-N nel modello concettuale, bisogna creare una tabella detta tabella di transizione. Le chiavi primarie delle entità tra le quali esiste una relazione M-N vengono aggiunte alla nuova tabella creata la cui chiave primaria è formata dall'unione di quest'ultime.
- Non ci interessano regole sulla traduzione di attributi multivalore in quanto assenti nel nostro diagramma



1.5 Utenti e Loro Categorie

Il nostro Database è composto da 5 utenti di cui 4 di tipo comune (addestratore,veterinario,cliente e dipendente) e 1 di tipo amministratore (db_petshop) al quale sono consentite tutte le operazioni



1.5.1 Volumi

TABELLA	TIPO	VOLUME	INCREMENTO	PERIODO
ALLEVAMENTO	E	16	5	anno
DIPENDENTE	E	17	5	semestre
ANIMALE	E	20	20	mese
CLIENTE	E	24	20	settimana
ADOTTA	A	20	10	settimana
RESTITUISCE	A	17	2	settimana
REPARTO	E	15	1	anno
FORNITORE	E	20	5	anno
CONSEGNA	E	20	60	settimana
PRODOTTO	E	20	20	bimestre
ACQUISTO	E	30	300	settimana
ACQUISTO_TESSERA	E	15	250	settimana
ACQUISTO_CONTIENE	A	50	600	settimana
VINCITORE_LOTTERIA	ED	0	1	mese
CONSEGNA_CONTIENE	A	30	300	settimana
ADDESTRATORE	E	20	20	anno
CORSO_DI_ADDESTRAMENTO	E	15	10	mese
SI_ISCRIVE	A	15	50	mese
STIPENDIO	E	17	17	mese
TURNI_DI_LAVORO	E	15	12	settimana
E_PRESENTE	A	85	85	settimana
VETERINARIO	E	15	10	anno
PRENOTAZIONE	E	30	30	settimana
VISITA_VETERINARIO	ED	15	15	settimana
VISITA_PETCARE	ED	15	15	settimana
PROMOZIONE	E	20	10	settimana
PROMOZIONE_RIGUARDA	A	20	10	settimana
RICHIAMO	ED	0	1	mese

1.6 Vincoli di Integrità

I vincoli sono detti statici, se non variano per l'intero ciclo di vita del database e dinamici se possono subire variazioni con il passare del tempo

1.6.1 Vincoli Statici

- Un cliente puo adottare un animale solo se maggiorenne
- Ogni dipendente assunto deve avere un eta compresa tra i 18 e i 60 anni
- Sono ammessi due tipi di prenotazioni: petcare e veterinaria
- Un animale adottato da un Cliente A non può essere restituito da un cliente B
- Un animale adottato dal cliente A non può essere adottato dal cliente B se il cliente A non ne ha effettuato la restituzione
- Lo sconto massimo applicabile non può mai essere tale da impedire un guadagno superiore al 30
- Ogni reparto ha massimo un direttore
- È ammesso un solo vincitore lotteria per mese

1.6.2 Vincoli Dinamici

- Ogni tipo X di animale ha un proprio reparto che condivide con un altro tipo diverso da X
- Un dipendente con meno di 2 anni di esperienza in azienda non può ricevere più di 1300 euro di stipendio
- Non posso eliminare un reparto se dentro ci sono ancora animali, dipendenti o prodotti con quantità disponibile maggiore di zero
- Il numero massimo di consegne in un giorno è 2
- Non si può registrare un adozione con data diversa da quella odierna
- Un cliente che ha un numero di restituzioni maggiore di due non può adottare un animale
- Non è possibile eliminare uno stipendio se il dipendente a esso associato lavora per l'azienda o se la data di emissione dello stipendio appartiene all'anno corrente
- È ammesso un solo richiamo al mese

1.7 Verifica Di Normalità

1.7.1 Prima Forma Normale

Il DB rispetta la prima forma normale, data la presenza di tutti campi unici

1.7.2 Seconda Forma Normale

IL DB rispetta la seconda forma normale, data la mancanza di attributi che dipendono solo parzialmente dalla chiave composta

1.7.3 Terza Forma Normale

Il DB rispetta la terza forma normale data la mancanza di attributi non-chiave generici dipendenti funzionalmente da un altro attributo non-chiave generico.
Il DB è in 3FN e in BCNF (Boys-Codd Normal Form)

Capitolo 2

IMPLEMENTAZIONE

**LA SPIEGAZIONE DEL CODICE SI TROVA NELLA CARTELLA 'CODICE' ALLEGATA ALLA RELAZIONE

La fase di progettazione è ormai finita e bisogna scrivere il codice che darà vita al nostro DB. Il linguaggio utilizzato è il PL/SQL

2.1 Creazione Utenti

Fase di creazione degli utenti che andranno ad accedere al Database

```
CREATE USER C##DB_PETSHOP IDENTIFIED BY doglover111;
CREATE USER C##CLIENTE IDENTIFIED BY superclient222;
CREATE USER C##DIPENDENTE IDENTIFIED BY bestworkerever333;
CREATE USER C##ADDESTRATORE IDENTIFIED BY cesarmilan444;
CREATE USER C##VET IDENTIFIED BY cheaperdrhouse555;
```

2.2 Data Definition Language

Il DDL riflette il modello relazionale: per creare le tabelle usiamo il comamdo PL/SQL CREATE TABLE e attraverso FOREIGN KEY, PRIMARY KEY e CONSTRAINT specifichiamo le chivi primarie, le chiavi esterne e i vincoli

2.2.1 Esempi di Tabelle

ANIMALE È l'entità su cui si fonda il Database. Come chiave PK ha un tipo CHAR(12) relativo al codice del suo microchip e come FK un tipo CHAR(11) relativo alla partita IVA dell'allevamento da cui proviene. È anche presente un constraint che regola l'immissione dei dati relativi all attributo SESSO

```
CREATE TABLE ANIMALE (
    CODICE_ANIMALE     CHAR(12)  PRIMARY KEY,
    PESO  NUMBER(2,0)  NOT NULL,
    DATA_ACQUISIZIONE DATE   NOT NULL,
    SESSO CHAR(1)  NOT NULL,
    REPARTO     NUMBER(2,0)   NOT NULL,
    TIPO VARCHAR2(20)  NOT NULL,
    RAZZA VARCHAR2(20)  NOT NULL,
    PREZZO NUMBER(4,0)  NOT NULL,
    COSTO_ACQUISIZIONE NUMBER(4,0),
    PARTITA_IVA_ALLEVAMENTO CHAR(11),
    CONSTRAINT FK_ALLEVAMENTO     FOREIGN KEY (PARTITA_IVA_ALLEVAMENTO)
    CONSTRAINT CHECK_SESSO_AN   CHECK (SESSO IN ('M','F'))
);
```

CLIENTE Entità fondamentale da cui provengono gli incassi dell'azienda. CLIENTE ha una PK di tipo CHAR(16) relativo al suo codice fiscale e lo stesso constraint di ANIMALE

```
CCREATE TABLE CLIENTE (
    CF  CHAR(16)  PRIMARY KEY,
    NOME  VARCHAR2(20)  NOT NULL,
    SECONDO_NOME      VARCHAR2(20),
    COGNOME  VARCHAR2(20)  NOT NULL,
    DATA_NASCITA  DATE NOT NULL,
    VIA  VARCHAR2(20),
    CIVICO  NUMBER(5,0),
    CAP  VARCHAR2(6),
    CITTA VARCHAR2(30),
    PROVINCIA VARCHAR2(30),
    SESSO  VARCHAR2(1),
    NUMERO_CELLULARE VARCHAR2(15) NOT NULL,
    EMAIL                    VARCHAR2(40)    NOT NULL,
    CONSTRAINT CHECK_SESSO  CHECK (SESSO IN ('M','F'))
);
```

PRODOTTO Prodotti e accessori per la cura dell'animale. PRODOTTO ha come PK un tipo CHAR(8) relativo al suo codice prodotto

```
CREATE TABLE PRODOTTO (
    CODICE_PRODOTTO  CHAR(8) PRIMARY KEY,
    TIPO_PRODOTTO VARCHAR2(70),
    REPARTO      VARCHAR2(2),
    PREZZO      NUMBER(5,2) NOT NULL,
    COSTO_ACQUISTO NUMBER(5,2),
    QUANTITA_DISPONIBILE NUMBER NOT NULL
);
```

DIPENDENTE Ultima delle 3 entità fondamentali del DB. DIPENDENTE ha come PK un tipo char(16) relativo ha il suo codice fiscale e una FK di tipo NUMBER(2,0) relativo al numero di reparto in cui lavora (PK di REPARTO)

```
CREATE TABLE DIPENDENTE (
  NUMERO_REPARTO                    NUMBER(2,0),
  CF          CHAR(16)  PRIMARY KEY,
NOME  VARCHAR2(20)  NOT NULL,
SECONDO_NOME       VARCHAR2(20),
COGNOME  VARCHAR2(20)  NOT NULL,
DATA_NASCITA  DATE NOT NULL,
VIA  VARCHAR2(20),
CIVICO  NUMBER(5,0) ,
CAP  VARCHAR2(6),
CITTA VARCHAR2(20),
SESSO  VARCHAR2(1),
DATA_ASSUNZIONE DATE,
NUMERO_CELLULARE VARCHAR2(15) NOT NULL,
        EMAIL                     VARCHAR2(35)    NOT NULL,
        CONSTRAINT FK_REPARTO    FOREIGN KEY (NUMERO_REPARTO) REFERENCES REP
CONSTRAINT CHECK_SESSO_DIP   CHECK (SESSO IN ('M','F'))
);
```

2.3 Data Manipulation Language

Popoliamo il database tramite il comando PL/SQL INSERT con dati fittizzi per simulare un vero DB

```
INSERT INTO ALLEVAMENTO
(PARTITA_IVA,NOME_ALLEVAMENTO,VIA,CIVICO,CAP,CITTA)
VALUES ('12656730756','ALLEVAMENTO DEL CAMPIONE','INCROCIO GRASSI',2,'66800','POTENZA');

INSERT INTO ANIMALE
(PARTITA_IVA_ALLEVAMENTO,CODICE_ANIMALE,PESO,DATA_ACQUISIZIONE,SESSO,REPARTO,TIPO,RAZZA,COSTO_ACQUISIZIONE)
VALUES ('17657040949','000000035770',6,TO_DATE('11/12/2021','DD/MM/YYYY'),'M',03,'CANE','AKITA INU');

INSERT INTO CLIENTE
(CF,NOME,SECONDO_NOME,COGNOME,DATA_NASCITA,VIA,CIVICO,CAP,CITTA,SESSO,NUMERO_CELLULARE,EMAIL)
VALUES ('BNCMRA70A20H501B','MARIO',NULL,'BIANCHI',TO_DATE('20/01/1970','DD/MM/YYYY'),'LEOPARDI',12,'80011','ACERR

INSERT INTO ADOTTA
(CODICE_ANIMALE,DATA_ADOZIONE,PREZZO_ADOZIONE,CF_CLIENTE)
VALUES ('000000035770',TO_DATE('15/01/2022','DD/MM/YYYY'),400,'DNSMTT00P23F839I');

INSERT INTO REPARTO
(NUMERO_REPARTO,NOME_REPARTO)
VALUES (1,'CANI TAGLIA PICCOLA');

INSERT INTO FORNITORE
(PARTITA_IVA,NOME_FORNITORE,VIA,CIVICO,CAP,CITTA,NUMERO_TELEFONO)
VALUES ('57156000135','ZOOCANA ANIMAL','VIA REGINA ELENA',33,'80077','ISCHIA','3315651295');

INSERT INTO CONSEGNA
(NUMERO_CONSEGNA, DATA_CONSEGNA, PARTITA_IVA_FORNITORE)
VALUES ('37842802841',TO_DATE('2022-01-05','YYYY-MM-DD'),'57156000135');

INSERT INTO PRODOTTO
(CODICE_PRODOTTO, TIPO_PRODOTTO, REPARTO, PREZZO, COSTO_ACQUISTO, QUANTITA_DISPONIBILE)
VALUES ('67763013','CIOTOLA PER CANI',10,4.99,1,200);

INSERT INTO CONSEGNA_CONTIENE
(NUMERO_CONSEGNA, CODICE_PRODOTTO, QUANTITA)
VALUES ('37842802841','67763013',200);

INSERT INTO ACQUISTO
(NUMERO_SCONTRINO,DATA_ACQUISTO)
VALUES ('72549302184',TO_DATE('2022-01-03','YYYY-MM-DD'));

INSERT INTO ACQUISTO_TESSERA
(NUMERO_SCONTRINO, DATA_ACQUISTO, CF_CLIENTE)
VALUES ('72549302184',TO_DATE('2022-01-03','YYYY-MM-DD'),'DNSMTT00P23F839I');

INSERT INTO ACQUISTO_CONTIENE
(NUMERO_SCONTRINO,CODICE_PRODOTTO,QUANTITA)
VALUES ('34601774170','59022433',4);

INSERT INTO ADDESTRATORE
(NUMERO_TESSERINO,NOME,COGNOME,DATA_DI_NASCITA)
VALUES ('010235100','GERARDO','TOSCANO',TO_DATE('03/03/1993','DD/MM/YYYY'));

INSERT INTO CORSO_DI_ADDESTRAMENTO
(CODICE_CORSO,INIZIO_CORSO,FINE_CORSO, NOME_CORSO,TARIFFA_ADDESTRATORE,NUMERO_TESSERINO_ADDESTRATORE)
VALUES ('011',TO_DATE('01/07/2022','DD/MM/YYYY'),TO_DATE('01/08/2022','DD/MM/YYYY'),'MINIGIOCHI PER DIVERTIMENTO
```

```
INSERT INTO SI_ISCRIVE
(CF_CLIENTE, CODICE_CORSO, DATA_ISCRIZIONE,COSTO_ISCRIZIONE,MOD_PAGAMENTO)
VALUES ('SPSFNC83L51A509B','005',TO_DATE('18/02/2022','DD/MM/YY'),400,1);

INSERT INTO DIPENDENTE
(NUMERO_REPARTO,CF,NOME,SECONDO_NOME,COGNOME,DATA_NASCITA,VIA,CIVICO,CAP,CITTA,SESSO,DATA_ASSUNZIONE,NUMERO_CELLU
VALUES (04,'1111111111111000','TERESA','BIANCA','BIANCO',TO_DATE('20/11/2001','DD/MM/YYYY'),'CALTANISSETTA',2,'81

INSERT INTO STIPENDIO
(CF_DIPENDENTE,DATA_STIPENDIO,IBAN_DIPENDENTE,TOTALE_STIPENDIO)
VALUES ('1111111100000000',TO_DATE('01/02/2022','DD/MM/YYYY'),'IT89003002032808649383641 94',1200);

UPDATE REPARTO
SET CF_CAPOREPARTO ='1111111000000000'
WHERE NUMERO_REPARTO = 10;

INSERT INTO TURNI_DI_LAVORO
(GIORNATA_DI_LAVORO,ORA_INIZIO,ORA_FINE)
VALUES (TO_DATE('05/01/2022','DD/MM/YYYY'),TO_TIMESTAMP('14:00','HH24:MI'),TO_TIMESTAMP('20:00','HH24:MI'));

INSERT INTO E_PRESENTE
(GIORNATA_DI_LAVORO,ORA_INIZIO,CF_DIPENDENTE,E_PRESENTE,ORA_ENTRATA,ORA_USCITA)
VALUES (TO_DATE('06/01/2022','DD/MM/YYYY'),TO_TIMESTAMP('08:00','HH24:MI'),'1111111111000000','Y',TO_TIMESTAMP('0

INSERT INTO VETERINARIO
(CODICE_VETERINARIO, NOME, COGNOME, DATA_DI_NASCITA)
VALUES ('004892094849','LUCIA','CARPINO',TO_DATE('09/041988','DD/MM/YYYY'));

INSERT INTO PRENOTAZIONE
CODICE_PRENOTAZIONE,DATA_PRENOTAZIONE,ORA_PRENOTAZIONE,TIPO_PRENOTAZIONE)
VALUES ('000000',TO_DATE('04/01/2022','DD/MM/YYYY'),TO_DATE('18:00:00','HH24:MI:SS'),'P');

INSERT INTO VISITA_VETERINARIO
(CODICE_VETERINARIO,TARIFFA_VETERINARIO,PREZZO_VISITA,CODICE_ANIMALE,CODICE_PRENOTAZIONE)
VALUES ('004893992930',90,150, '112542139480','004893');

INSERT INTO VISITA_PETCARE
(CF_DIPENDENTE,PREZZO_VISITA,CODICE_ANIMALE, CODICE_PRENOTAZIONE)
VALUES ('1111111000000000',55,'112031056012','111011');

INSERT INTO PROMOZIONE
(CODICE_PROMOZIONE, DATA_INIZIO, DATA_FINE)
VALUES (8,TO_DATE('2022-02-01','YYYY-MM-DD'),TO_DATE('2022-02-05','YYYY-MM-DD'));

INSERT INTO PROMOZIONE_RIGUARDA
(CODICE_PROMOZIONE,CODICE_PRODOTTO,SCONTO_APPLICATO)
VALUES (3,'59022433',0.1);

INSERT INTO RESTITUISCE
(CODICE_ANIMALE,DATA_RESTITUZIONE,RIMBORSO,CF_CLIENTE)
VALUES ('111790614514',TO_DATE('1/08/2022','DD/MM/YYYY'),17.50,'SBRBNC80R41F839T');
```

2.4 Trigger

I Trigger salvaguardano l'integrità dei dati impedendo l'immisione di dati che non rispettano i vincoli da noi posti

2.4.1 RESTITUISCE_TR

```
/*
    TRIGGER #1
    TRIGGER CHE IMPEDISCE AD UN CLIENTE DI RESTITUIRE UN ANIMALE CHE NON HA MAI ADOTTATO
```

```
*/

CREATE OR REPLACE TRIGGER RESTITUISCE_TR
AFTER INSERT OR UPDATE ON RESTITUISCE
FOR EACH ROW

DECLARE

    COD_ANIMALE         ANIMALE.CODICE_ANIMALE%TYPE;
    DATA_AD             DATE;
    TROPPO_TEMPO        EXCEPTION;
    PRAGMA AUTONOMOUS_TRANSACTION;

BEGIN
    SELECT CODICE_ANIMALE,DATA_ADOZIONE
      INTO COD_ANIMALE,DATA_AD
       FROM ADOTTA
        WHERE CODICE_ANIMALE = :NEW.CODICE_ANIMALE AND CF_CLIENTE = :NEW.CF_CLIENTE;

        IF(ABS(FLOOR(MONTHS_BETWEEN(SYSDATE,DATA_AD))) > 1)
        THEN RAISE TROPPO_TEMPO;
        END IF;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR (-20009, 'NON PUOI RESTITUIRE UN ANIMALE CHE NON HAI MAI ADOTTATO');

  WHEN TROPPO_TEMPO THEN
    RAISE_APPLICATION_ERROR (-20019, 'NON PUOI RESTITUIRE UN ANIMALE SE È PASSATO PIU DI UN MESE');

END
COMMIT;
/
```

2.4.2 CONSEGNA_TR

```
/*
    TRIGGER #2
    TRIGGER CHE IMPENDISCE L'INSERIMENTO DI UN NUMERO DI CONSEGNE MAGGIORE A 2 NELLO STESSO GIORNO
*/
CREATE OR REPLACE TRIGGER CONSEGNA_TR
AFTER INSERT OR UPDATE ON CONSEGNA
FOR EACH ROW

DECLARE

    NUMERO_CONSEGNE     NUMBER;
    TROPPE_CONSEGNE     EXCEPTION;
    PRAGMA AUTONOMOUS_TRANSACTION;

BEGIN
    SELECT COUNT(*)
      INTO NUMERO_CONSEGNE
      FROM CONSEGNA
     WHERE DATA_CONSEGNA = :NEW.DATA_CONSEGNA;

     IF(NUMERO_CONSEGNE = 2) THEN

       RAISE TROPPE_CONSEGNE;

     END IF;

EXCEPTION
  WHEN TROPPE_CONSEGNE THEN
    RAISE_APPLICATION_ERROR (-20010, 'OGNI GIORNO IL NUMERO MASSIMO DI CONSEGNE È: 2. VERIFICA IL CORRETTO INSERIMENT
```

```
END
COMMIT;
/
```

2.4.3 DATA_ADOZIONE_TR

```
/*
    TRIGGER #3

    TRIGGER CHE IMPEDISCE L'IMMISSIONE DI UNA DATA D'ADOZIONE DIVERSA DA QUELLA ODIERNA
*/

CREATE OR REPLACE TRIGGER DATA_ADOZIONE_TR
BEFORE INSERT OR UPDATE ON ADOTTA
FOR EACH ROW

BEGIN
    /*
        TRUNC TRONCA LA DATA IN UN FORMATO SPECIFICO. SE PARAMETRO INERENTE AL FORMATO È OMESSO ALLORA LA FUNZIONE TR
        ESCLUSIVAMENTE LE INFORMAZIONI GG/MM/YYYY
    */
   IF (TRUNC (:NEW.DATA_ADOZIONE) < TRUNC(SYSDATE) OR TRUNC (:NEW.DATA_ADOZIONE) > TRUNC(SYSDATE))

    THEN
        :NEW.DATA_ADOZIONE := TRUNC(SYSDATE);

      DBMS_OUTPUT.PUT_LINE('LA DATA DI ADOZIONE INSERITA È ANTECEDENTE O SUCCESSIVA ALLA DATA ODIERNA. PER TANTO IL P
   END IF;

END
COMMIT;
/
```

2.4.4 POCA_ESPERIENZA_TR

```
/*
    TRIGGER #4
    TRIGGER CHE IMPEDISCE AI DIPENDENTI CON MENO DI DUE ANNI DI ESPERIENZA DI RICEVERE PIÙ DI 1300 EURO DI STIPENDIO
*/

CREATE OR REPLACE TRIGGER POCA_ESPERIENZA_TR
AFTER INSERT OR UPDATE ON STIPENDIO
FOR EACH ROW

DECLARE

    POCA_ESPERIENZA     EXCEPTION;
    DATA_ASS_DIP        DATE;
    PRAGMA AUTONOMOUS_TRANSACTION;

BEGIN
    SELECT DATA_ASSUNZIONE INTO DATA_ASS_DIP
        FROM DIPENDENTE WHERE CF = :NEW.CF_DIPENDENTE;


    IF
        TO_NUMBER(TO_CHAR(DATA_ASS_DIP,'YYYY')) > TO_NUMBER(TO_CHAR(SYSDATE,'YYYY')) - 2  AND :NEW.TOTALE_STIPENDIO>1
        THEN RAISE POCA_ESPERIENZA;

    END IF;

EXCEPTION
  WHEN POCA_ESPERIENZA THEN
    RAISE_APPLICATION_ERROR(-20011,'IL DIPENDETE NON HA RAGGIUNTO L ESPERIENZA MINIMA PER UN TALE STIPENDIO');

END
```

```
COMMIT;
/
```

2.4.5 NON_IDONEO_TR

```
/*
    TRIGGER #5
    TRIGGER CHE IMPEDISCE A UN CLIENTE NON IDONEO DI ADOTTARE UN ANIMALE
*/

CREATE OR REPLACE TRIGGER NON_IDONEO_TR
AFTER INSERT OR UPDATE ON ADOTTA
FOR EACH ROW

DECLARE
    NON_IDONEO            EXCEPTION;
    NUMERO_RESTITUZIONI   NUMBER;
    DATA_ADOZIONE_TR      DATE;
    ETA                   NUMBER;
    DN                    DATE;
    PRAGMA AUTONOMOUS_TRANSACTION;

BEGIN

    SELECT DATA_NASCITA INTO DN FROM CLIENTE WHERE CF = :NEW.CF_CLIENTE;

    ETA:= FLOOR(MONTHS_BETWEEN(SYSDATE, DN) / 12);

    IF (ETA < 18) THEN RAISE NON_IDONEO;
    END IF;

    SELECT COUNT (*) INTO NUMERO_RESTITUZIONI
     FROM RESTITUISCE WHERE CF_CLIENTE = :NEW.CF_CLIENTE
      GROUP BY CF_CLIENTE;

    IF (NUMERO_RESTITUZIONI > 1) THEN RAISE NON_IDONEO;
    END IF;

EXCEPTION
  WHEN NON_IDONEO THEN
    RAISE_APPLICATION_ERROR(-20012,'IL CLIENTE NON È IDONEO ALL ADOZIONE');

  WHEN NO_DATA_FOUND THEN
    NULL;
END
COMMIT;
/
```

2.4.6 REPARTO_TR

```
/*
    TRIGGER #6
    TRIGGER CHE IMPEDISCE DI INSERIRE GLI ANIMALI IN UN REPARTO SBAGLIATO
*/

CREATE OR REPLACE TRIGGER REPARTO_TR
BEFORE INSERT OR UPDATE ON ANIMALE
FOR EACH ROW

DECLARE
  REPARTO_SBAGLIATO     EXCEPTION;
BEGIN
  IF(:NEW.TIPO = 'CANE') THEN
  IF (:NEW.REPARTO != 1 AND :NEW.REPARTO != 2 AND :NEW.REPARTO != 3) THEN
    :NEW.REPARTO := 1;
```

```
  DBMS_OUTPUT.PUT_LINE ('REPARTO CAMBIATO IN :'||:NEW.REPARTO);
  END IF;
  ELSIF (:NEW.TIPO = 'GATTO' AND :NEW.REPARTO != 4) THEN
  :NEW.REPARTO := 4;
  DBMS_OUTPUT.PUT_LINE ('REPARTO CAMBIATO IN :'||:NEW.REPARTO);

  ELSIF (:NEW.TIPO = 'CONIGLIO' AND :NEW.REPARTO  != 5) THEN
  :NEW.REPARTO := 5;
  DBMS_OUTPUT.PUT_LINE ('REPARTO CAMBIATO IN :'||:NEW.REPARTO);

  ELSIF (:NEW.TIPO = 'RETTILE' AND :NEW.REPARTO  != 6) THEN
  :NEW.REPARTO := 6;
  DBMS_OUTPUT.PUT_LINE ('REPARTO CAMBIATO IN :'||:NEW.REPARTO);

  ELSIF (:NEW.TIPO = 'VOLATILE' AND :NEW.REPARTO  != 7) THEN
  :NEW.REPARTO := 7;
  DBMS_OUTPUT.PUT_LINE ('REPARTO CAMBIATO IN :'||:NEW.REPARTO);

  ELSIF (:NEW.TIPO = 'PESCE' AND :NEW.REPARTO  != 8) THEN
  :NEW.REPARTO := 8;
  DBMS_OUTPUT.PUT_LINE ('REPARTO CAMBIATO IN :'||:NEW.REPARTO);

  ELSIF (:NEW.TIPO = 'ANFIBIO' AND :NEW.REPARTO  != 12) THEN
  :NEW.REPARTO := 12;
  DBMS_OUTPUT.PUT_LINE ('REPARTO CAMBIATO IN :'||:NEW.REPARTO);


  END IF;
END
COMMIT;
/
```

2.4.7 PROMO_TR

```
/*
    TRIGGER #7
    TRIGGER CHE IMPEDISCE A UNA PROMOZIONE DI EFFETTUARE UNO SCONTO ECCESSIVO
*/

CREATE OR REPLACE TRIGGER PROMO_TR
AFTER INSERT OR UPDATE ON PROMOZIONE_RIGUARDA
FOR EACH ROW

DECLARE
    SCONTO_ECCESSIVO            EXCEPTION;
    COSTO_ACQUISTO_PRODOTTO     PRODOTTO.COSTO_ACQUISTO%TYPE;
    PREZZO_PRODOTTO             PRODOTTO.PREZZO%TYPE;
    PRAGMA AUTONOMOUS_TRANSACTION;

BEGIN
    SELECT COSTO_ACQUISTO INTO COSTO_ACQUISTO_PRODOTTO
    FROM PRODOTTO
    WHERE CODICE_PRODOTTO = :NEW.CODICE_PRODOTTO;

    SELECT PREZZO INTO PREZZO_PRODOTTO
    FROM PRODOTTO
    WHERE CODICE_PRODOTTO = :NEW.CODICE_PRODOTTO;

    IF (PREZZO_PRODOTTO - PREZZO_PRODOTTO*(:NEW.SCONTO_APPLICATO) - COSTO_ACQUISTO_PRODOTTO <= COSTO_ACQUISTO_PRODOTT

        RAISE SCONTO_ECCESSIVO;

    END IF;

EXCEPTION
  WHEN SCONTO_ECCESSIVO THEN
```

```
    RAISE_APPLICATION_ERROR (-20014, 'LO SCONTO EFFETTUATO NON PERMETTE UN MARGINE DI PROFITTO DEL 30%');

END
COMMIT;
/
```

2.4.8 QT_DISPONIBILI_TR

```
/*
    TRIGGER #8
    TRIGGER CHE SCALA L'ATTRIBUTO QUANTITÀ DISPONIBILE OGNI VOLTA CHE UN ARTICOLO VIENE COMPRATO
*/

CREATE OR REPLACE TRIGGER QT_DISPONIBILI_TR
AFTER INSERT ON ACQUISTO_CONTIENE
FOR EACH ROW
DECLARE
    COD                 PRODOTTO.CODICE_PRODOTTO%TYPE;
    QT                  PRODOTTO.QUANTITA_DISPONIBILE%TYPE;
    QT_DISP             PRODOTTO.QUANTITA_DISPONIBILE%TYPE;
    ECCESSIVO           EXCEPTION;
    PRAGMA AUTONOMOUS_TRANSACTION;

BEGIN

  SELECT QUANTITA_DISPONIBILE INTO QT_DISP
  FROM PRODOTTO WHERE CODICE_PRODOTTO = :NEW.CODICE_PRODOTTO;

  IF (QT_DISP < :NEW.QUANTITA)  THEN
    RAISE ECCESSIVO;
  END IF;

  UPDATE PRODOTTO
  SET QUANTITA_DISPONIBILE = QUANTITA_DISPONIBILE - :NEW.QUANTITA
  WHERE CODICE_PRODOTTO = :NEW.CODICE_PRODOTTO;
  COMMIT;

  SELECT CODICE_PRODOTTO,QUANTITA_DISPONIBILE INTO COD,QT
    FROM PRODOTTO
    WHERE CODICE_PRODOTTO = :NEW.CODICE_PRODOTTO;

  DBMS_OUTPUT.PUT_LINE('QUANTITÀ DISPONIBILE PRODOTTO #'||COD||':  '||QT);

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR (-20015, 'PRODOTTO INESISTENTE');

  WHEN ECCESSIVO THEN
    RAISE_APPLICATION_ERROR (-20016, 'NON ABBIAMO ABBASTANZA PRODOTTO DA SODDISFARE LA VOSTRA RICHIESTA');

END
COMMIT;
/

CREATE OR REPLACE TRIGGER QT_DISPONIBILI_TR_II
AFTER UPDATE ON ACQUISTO_CONTIENE
FOR EACH ROW
DECLARE
    COD                 PRODOTTO.CODICE_PRODOTTO%TYPE;
    QT                  PRODOTTO.QUANTITA_DISPONIBILE%TYPE;
    QT_DISP             PRODOTTO.QUANTITA_DISPONIBILE%TYPE;
    ECCESSIVO           EXCEPTION;
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN

  SELECT QUANTITA_DISPONIBILE INTO QT_DISP
```

```
  FROM PRODOTTO WHERE CODICE_PRODOTTO = :NEW.CODICE_PRODOTTO;

  IF (QT_DISP + :OLD.QUANTITA < :NEW.QUANTITA)  THEN
    RAISE ECCESSIVO;
  END IF;


  UPDATE PRODOTTO
  SET QUANTITA_DISPONIBILE = QUANTITA_DISPONIBILE - :NEW.QUANTITA + :OLD.QUANTITA
  WHERE CODICE_PRODOTTO = :NEW.CODICE_PRODOTTO;
  COMMIT;

  SELECT CODICE_PRODOTTO,QUANTITA_DISPONIBILE INTO COD,QT
    FROM PRODOTTO
    WHERE CODICE_PRODOTTO = :NEW.CODICE_PRODOTTO;

  DBMS_OUTPUT.PUT_LINE('QUANTITÀ DISPONIBILE PRODOTTO #'||COD||':  '||QT);

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR (-20015, 'PRODOTTO INESISTENTE');

  WHEN ECCESSIVO THEN
    RAISE_APPLICATION_ERROR (-20016, 'NON ABBIAMO ABBASTANZA PRODOTTO DA SODDISFARE LA VOSTRA RICHIESTA');

END
COMMIT;
/
```

2.4.9 NON_DISPONIBILE_TR

```
/*
    TRIGGER #9
    TRIGGER CHE IMPEDISCE L'ADOZIONE DI UN ANIMALE GIA ADOTTATO DA QUALCUN ALTRO
*/


CREATE OR REPLACE TRIGGER NON_DISPONIBILE_TR
AFTER INSERT OR UPDATE ON ADOTTA
FOR EACH ROW
DECLARE
    CA              ANIMALE.CODICE_ANIMALE%TYPE;
    NON_DISP        EXCEPTION;
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN

    SELECT CODICE_ANIMALE INTO CA
    FROM (
    SELECT CF_CLIENTE,CODICE_ANIMALE FROM ADOTTA
    MINUS
    SELECT CF_CLIENTE, CODICE_ANIMALE FROM RESTITUISCE
    ) WHERE CODICE_ANIMALE = :NEW.CODICE_ANIMALE;

    IF (CA IS NOT NULL) THEN
       RAISE NON_DISP;

    END IF;


EXCEPTION
    WHEN NO_DATA_FOUND THEN
        NULL;

    WHEN NON_DISP THEN
        RAISE_APPLICATION_ERROR(-20017,'ANIMALE NON DISPONIBILE :( . . . GIA ADOTTATO DA QUALCUN ALTRO!');
```

```
END;
/
```

2.4.10 DEL_REPARTO_TR

```
/*
    TRIGGER #10
    TRIGGER CHE IMPEDISCE DI ELIMINARE UN REPARTO SE CI SONO ANCORA PRODOTTI DISPONIBILI,ANIMALI O DIPENDENTI CHE CI
*/


CREATE OR REPLACE TRIGGER DEL_REPARTO_TR
BEFORE DELETE ON REPARTO
FOR EACH ROW
DECLARE
    N_DIP                   NUMBER;
    N_AN                    NUMBER;
    REPARTO_POPOLATO        EXCEPTION;
    PRAGMA     AUTONOMOUS_TRANSACTION;
BEGIN

    SELECT COUNT(*) INTO N_DIP
    FROM DIPENDENTE WHERE NUMERO_REPARTO = :OLD.NUMERO_REPARTO;

    IF(N_DIP > 0)
        THEN RAISE REPARTO_POPOLATO;
    END IF;

    SELECT COUNT (*) INTO N_AN
    FROM
        (SELECT CODICE_ANIMALE,REPARTO FROM ANIMALE
        MINUS
        SELECT CODICE_ANIMALE,REPARTO FROM (
            SELECT AD.CODICE_ANIMALE,REPARTO FROM ADOTTA AD JOIN ANIMALE AN ON AD.CODICE_ANIMALE = AN.CODICE_ANIMALE
            MINUS
            SELECT R.CODICE_ANIMALE,REPARTO FROM RESTITUISCE R JOIN ANIMALE AN ON R.CODICE_ANIMALE = AN.CODICE_ANIMAL
    WHERE REPARTO = :OLD.NUMERO_REPARTO;

    IF(N_AN > 0)
        THEN RAISE REPARTO_POPOLATO;
    END IF;

    FOR i IN (SELECT QUANTITA_DISPONIBILE FROM PRODOTTO WHERE REPARTO = :OLD.NUMERO_REPARTO) LOOP

        IF(i.QUANTITA_DISPONIBILE > 0) THEN
            RAISE REPARTO_POPOLATO;
        END IF;

    END LOOP;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        NULL;

    WHEN REPARTO_POPOLATO THEN
        RAISE_APPLICATION_ERROR(-20018,'CI SONO ANCORA DIPENDENTI,ANIMALI O PRODOTTI RIMASTI NEL REPARTO!');

END;
/
```

2.4.11 DEL_STIPENDIO_TR

```
/*
    TRIGGER #11
    TRIGGER CHE IMPEDISCE DI ELIMINARE UNO STIPENDIO SE IL DIPENDENTE FA PARTE DELL'AZIENDA O SE FA PARTE DELL'ANNO C
*/
```

```
CREATE OR REPLACE TRIGGER DEL_STIPENDIO_TR
BEFORE DELETE ON STIPENDIO
FOR EACH ROW
DECLARE
  CF_DIP              DIPENDENTE.CF%TYPE;
  IN_ANNO             EXCEPTION;
  DIPENDENTE_ATTIVO   EXCEPTION;
  ANNO_CORRENTE       NUMBER;
  ANNO_STIPENDIO      NUMBER;
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN

    SELECT EXTRACT(YEAR FROM :OLD.DATA_STIPENDIO)  INTO ANNO_STIPENDIO
    FROM STIPENDIO
    WHERE DATA_STIPENDIO = :OLD.DATA_STIPENDIO AND IBAN_DIPENDENTE = :OLD.IBAN_DIPENDENTE;

    SELECT EXTRACT(YEAR FROM SYSDATE) INTO ANNO_CORRENTE
    FROM DUAL;

    IF (ANNO_STIPENDIO = ANNO_CORRENTE) THEN
        RAISE IN_ANNO;
    END IF;

    SELECT CF INTO CF_DIP
      FROM DIPENDENTE
        WHERE CF = :OLD.CF_DIPENDENTE;

    IF (CF_DIP IS NOT NULL)
      THEN RAISE DIPENDENTE_ATTIVO;
    END IF;


EXCEPTION
  WHEN NO_DATA_FOUND THEN
    NULL;

  WHEN DIPENDENTE_ATTIVO THEN
    RAISE_APPLICATION_ERROR(-20020, 'IL DIPENDENTE FA ANCORA PARTE DELLA AZIENDA E PERTANTO NON È POSSIBILE ELIMINARE

  WHEN IN_ANNO THEN
    RAISE_APPLICATION_ERROR(-20021, 'LO STIPENDIO APPARTIENE ALL ANNO CORRENTE E NON PUO ESSERE CANCELLATO');

END;
/
```

2.4.12 CHECK_ETA_TR

```
/*
    TRIGGER #12
    TRIGGER CHE IMPEDISCE ASSUMERE UN LAVORATORE CON UNA ETA NON COMPRESA TRA 18 E 60
*/

CREATE OR REPLACE TRIGGER CHECK_ETA_TR
BEFORE INSERT OR UPDATE ON DIPENDENTE
FOR EACH ROW
DECLARE
  ETA                 NUMBER;
  ETA_NO              EXCEPTION;
BEGIN

  ETA := ABS(MONTHS_BETWEEN(SYSDATE,:NEW.DATA_NASCITA))/12;

  IF(ETA < 18 OR ETA > 60) THEN
  RAISE ETA_NO;
  END IF;
```

```
EXCEPTION
  WHEN ETA_NO THEN
    RAISE_APPLICATION_ERROR(-20022, 'IL DIPENDENTE NON HA LA GIUSTA ETÀ PER ESSERE ASSUNTO');
END;
/

CREATE OR REPLACE VIEW ANIMALI_DISPONIBILI AS
    SELECT CODICE_ANIMALE,SESSO,TIPO,RAZZA,REPARTO FROM ANIMALE
    MINUS
    SELECT CODICE_ANIMALE,SESSO,TIPO,RAZZA,REPARTO FROM (
        SELECT AD.CODICE_ANIMALE,SESSO,TIPO,RAZZA,REPARTO FROM ADOTTA AD JOIN ANIMALE AN ON AD.CODICE_ANIMALE = AN.CO
        MINUS
        SELECT R.CODICE_ANIMALE,SESSO,TIPO,RAZZA,REPARTO FROM RESTITUISCE R JOIN ANIMALE AN ON R.CODICE_ANIMALE = AN.
        );


CREATE OR REPLACE VIEW COSTO_CONSEGNE AS
  SELECT SUM(COSTO_ACQUISTO * QUANTITA) AS TOTALE,NUMERO_CONSEGNA,MAX(DATA_CONSEGNA) AS DATA_CONSEGNA FROM
  (SELECT CO.NUMERO_CONSEGNA,CO.DATA_CONSEGNA,PR.CODICE_PRODOTTO,COSTO_ACQUISTO,QUANTITA FROM CONSEGNA CO JOIN CONSEG
   )
  IQ WHERE NUMERO_CONSEGNA = IQ.NUMERO_CONSEGNA
  GROUP BY NUMERO_CONSEGNA;


  CREATE OR REPLACE VIEW ISCRIZIONI_CORSI AS
  SELECT CF_CLIENTE,CDA.CODICE_CORSO,NOME_CORSO,NUMERO_TESSERINO_ADDESTRATORE,INIZIO_CORSO,FINE_CORSO
    FROM SI_ISCRIVE SI JOIN CORSO_DI_ADDESTRAMENTO CDA ON SI.CODICE_CORSO = CDA.CODICE_CORSO
            WHERE TRUNC(INIZIO_CORSO) < TRUNC(SYSDATE) AND TRUNC(FINE_CORSO) > TRUNC(SYSDATE);

CREATE OR REPLACE VIEW PRENOTAZIONI_PETCARE AS
SELECT * FROM PRENOTAZIONE
    WHERE TIPO_PRENOTAZIONE = 'P'
    AND TRUNC(DATA_PRENOTAZIONE) > TRUNC(SYSDATE);

CREATE OR REPLACE VIEW PRENOTAZIONI_VET AS
SELECT * FROM PRENOTAZIONE
    WHERE TIPO_PRENOTAZIONE = 'V'
    AND TRUNC(DATA_PRENOTAZIONE) > TRUNC(SYSDATE);
```

2.5 VISTE

Le viste tornano utili al creatore del DB per mostrare agli utenti la porzione di dati per cui vogliamo garantire l'accesso

2.5.1 ANIMALI_DISPONIBILI

```
CREATE OR REPLACE VIEW ANIMALI_DISPONIBILI AS
    SELECT CODICE_ANIMALE,SESSO,TIPO,RAZZA,REPARTO FROM ANIMALE
    MINUS
    SELECT CODICE_ANIMALE,SESSO,TIPO,RAZZA,REPARTO FROM (
        SELECT AD.CODICE_ANIMALE,SESSO,TIPO,RAZZA,REPARTO FROM ADOTTA AD JOIN ANIMALE AN ON AD.CODICE_ANIMALE = AN.CO
        MINUS
        SELECT R.CODICE_ANIMALE,SESSO,TIPO,RAZZA,REPARTO FROM RESTITUISCE R JOIN ANIMALE AN ON R.CODICE_ANIMALE = AN.
        );
```

2.5.2 COSTO_CONSEGNE

```
CREATE OR REPLACE VIEW COSTO_CONSEGNE AS
  SELECT SUM(COSTO_ACQUISTO * QUANTITA) AS TOTALE,NUMERO_CONSEGNA,MAX(DATA_CONSEGNA) AS DATA_CONSEGNA FROM
```

```
  (SELECT CO.NUMERO_CONSEGNA,CO.DATA_CONSEGNA,PR.CODICE_PRODOTTO,COSTO_ACQUISTO,QUANTITA FROM CONSEGNA CO JOIN CONSEG
   )
  IQ WHERE NUMERO_CONSEGNA = IQ.NUMERO_CONSEGNA
  GROUP BY NUMERO_CONSEGNA;
```

2.5.3 ISCRIZIONI_CORSI

```
  CREATE OR REPLACE VIEW ISCRIZIONI_CORSI AS
  SELECT CF_CLIENTE,CDA.CODICE_CORSO,NOME_CORSO,NUMERO_TESSERINO_ADDESTRATORE,INIZIO_CORSO,FINE_CORSO
    FROM SI_ISCRIVE SI JOIN CORSO_DI_ADDESTRAMENTO CDA ON SI.CODICE_CORSO = CDA.CODICE_CORSO
            WHERE TRUNC(INIZIO_CORSO) < TRUNC(SYSDATE) AND TRUNC(FINE_CORSO) > TRUNC(SYSDATE);
```

2.5.4 PRENOTAZIONI_PETCARE

```
CREATE OR REPLACE VIEW PRENOTAZIONI_PETCARE AS
SELECT * FROM PRENOTAZIONE
    WHERE TIPO_PRENOTAZIONE = 'P'
    AND TRUNC(DATA_PRENOTAZIONE) > TRUNC(SYSDATE);
```

2.5.5 PRENOTAZIONI_VET

```
CREATE OR REPLACE VIEW PRENOTAZIONI_VET AS
SELECT * FROM PRENOTAZIONE
    WHERE TIPO_PRENOTAZIONE = 'V'
    AND TRUNC(DATA_PRENOTAZIONE) > TRUNC(SYSDATE);
```

2.6 Procedure

Le procedure servono ad automatizzare il nostro database

2.6.1 LICENZIAMENTO

```
-- =============
-- LICENZIAMENTO
-- =============


CREATE OR REPLACE PROCEDURE LICENZIAMENTO
(CF_DIP IN CHAR)
IS
  N_REPARTO                     REPARTO.NUMERO_REPARTO%TYPE;
  CF_NUOVO_CAPOREPARTO          CHAR(16);
BEGIN

    BEGIN
    SELECT NUMERO_REPARTO INTO N_REPARTO
    FROM REPARTO
    WHERE CF_CAPOREPARTO = CF_DIP;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DELETE FROM DIPENDENTE WHERE CF = CF_DIP;
            COMMIT;
    END;

    BEGIN
```

```
    IF (N_REPARTO IS NOT NULL) THEN
    SELECT CF INTO CF_NUOVO_CAPOREPARTO FROM (
        SELECT CF FROM DIPENDENTE
        MINUS
        SELECT CF_CAPOREPARTO FROM REPARTO
    )
    WHERE ROWNUM = 1;
    END IF;

    COMMIT;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            SELECT MAX(CF) INTO CF_NUOVO_CAPOREPARTO FROM DIPENDENTE WHERE DATA_NASCITA = (SELECT MIN(DATA_NASCITA) F
        WHERE CF != CF_DIP);
        COMMIT;
    END;


    UPDATE REPARTO
    SET CF_CAPOREPARTO = CF_NUOVO_CAPOREPARTO
    WHERE NUMERO_REPARTO = N_REPARTO;
    COMMIT;

    UPDATE VISITA_PETCARE
    SET CF_DIPENDENTE = CF_NUOVO_CAPOREPARTO
    WHERE CF_DIPENDENTE = CF_DIP;
    COMMIT;

    DELETE FROM DIPENDENTE WHERE CF = CF_DIP;
    COMMIT;

    DBMS_OUTPUT.PUT_LINE ('IL DIPENDENTE LICENZIATO È: '||CF_DIP);

END;
/
```

2.6.2 BILANCIO

```
-- ===============
-- BILANCIO TOTALE
-- ===============



CREATE OR REPLACE PROCEDURE BILANCIO IS -- Creo delle variabili in cui andrò a memorizzare le varie uscite/entrate

STIPENDI            NUMBER(10,0);
TOTALE_CONSEGNE     NUMBER(10,0);
COSTO_VET           NUMBER(10,0);
COSTO_ADD           NUMBER(10,0);
RIMBORSI            NUMBER(10,0);
ACQUISTO_ANIMALI    NUMBER(10,0);
VENDITA_ANIMALI     NUMBER(10,0);
VENDITA_PRODOTTI    NUMBER(10,0);
VISITE_PC           NUMBER(10,0);
VISITE_VET          NUMBER(10,0);
CORSI               NUMBER(10,0);
ENTRATE             NUMBER(10,0);
USCITE              NUMBER(10,0);
CF_PIU_PAGATO       CHAR(16);
ANNO_CORRENTE       NUMBER(4,0);

BEGIN

  SELECT EXTRACT(YEAR FROM SYSDATE) INTO ANNO_CORRENTE
```

```
  FROM DUAL;

 SELECT  SUM(TOTALE_STIPENDIO) INTO STIPENDI
  FROM STIPENDIO
  WHERE EXTRACT(YEAR FROM DATA_STIPENDIO) IN (ANNO_CORRENTE);


DBMS_OUTPUT.PUT_LINE ('COSTI STIPENDI DIPENDENTI:'||STIPENDI);

 SELECT SUM(TOTALE) INTO TOTALE_CONSEGNE
 FROM COSTO_CONSEGNE
 WHERE EXTRACT(YEAR FROM DATA_CONSEGNA) IN (ANNO_CORRENTE);


DBMS_OUTPUT.PUT_LINE ('COSTI CONSEGNE:'||TOTALE_CONSEGNE);

 SELECT  SUM(TARIFFA_VETERINARIO),SUM(PREZZO_VISITA) INTO COSTO_VET,VISITE_VET
  FROM VISITA_VETERINARIO VV
  JOIN PRENOTAZIONE P ON P.CODICE_PRENOTAZIONE = VV.CODICE_PRENOTAZIONE
  WHERE EXTRACT(YEAR FROM DATA_PRENOTAZIONE) IN (ANNO_CORRENTE);

DBMS_OUTPUT.PUT_LINE ('COSTI VETERINARI:'||COSTO_VET);

 SELECT  SUM(TARIFFA_ADDESTRATORE) INTO COSTO_ADD -- Calcolo costi addestratore
  FROM CORSO_DI_ADDESTRAMENTO
  WHERE EXTRACT(YEAR FROM INIZIO_CORSO) IN (ANNO_CORRENTE);
DBMS_OUTPUT.PUT_LINE ('COSTI ADDESTRATORI:'||COSTO_ADD);

 SELECT  SUM(RIMBORSO) INTO RIMBORSI -- Calcolo costi rimborso animali
  FROM RESTITUISCE
  WHERE EXTRACT(YEAR FROM DATA_RESTITUZIONE) IN (ANNO_CORRENTE);

DBMS_OUTPUT.PUT_LINE ('COSTI INERENTI AI RIMBORSI:'||RIMBORSI);

SELECT  SUM(COSTO_ACQUISIZIONE) INTO ACQUISTO_ANIMALI -- Calcolo costi acquisto animali
  FROM ANIMALE
  WHERE EXTRACT(YEAR FROM DATA_ACQUISIZIONE) IN (ANNO_CORRENTE);


DBMS_OUTPUT.PUT_LINE ('COSTI ACQUISTO ANIMALI:'||ACQUISTO_ANIMALI);


SELECT  SUM(PREZZO_VISITA) INTO VISITE_PC -- Calcolo guadagni dalle visite petcare
  FROM VISITA_PETCARE VP
  JOIN PRENOTAZIONE P ON P.CODICE_PRENOTAZIONE = VP.CODICE_PRENOTAZIONE
  WHERE EXTRACT(YEAR FROM DATA_PRENOTAZIONE) IN (ANNO_CORRENTE);

DBMS_OUTPUT.PUT_LINE ('ENTRATE VISITE PETCARE:'||VISITE_PC);

DBMS_OUTPUT.PUT_LINE ('ENTRATE VISITE VETERINARIO:'||VISITE_VET);

SELECT  SUM(PREZZO_ADOZIONE) INTO VENDITA_ANIMALI -- Calcolo guadagni dalla vendita di animali
  FROM ADOTTA
  WHERE EXTRACT(YEAR FROM DATA_ADOZIONE) IN (ANNO_CORRENTE);


DBMS_OUTPUT.PUT_LINE ('ENTRATE VENDITA ANIMALI:'||VENDITA_ANIMALI);

SELECT  SUM(COSTO_ISCRIZIONE) INTO CORSI -- Calcolo guadagni dall'iscrizione ai corsi
  FROM SI_ISCRIVE SI JOIN CORSO_DI_ADDESTRAMENTO CDA ON
  CDA.CODICE_CORSO = SI.CODICE_CORSO
  WHERE EXTRACT(YEAR FROM INIZIO_CORSO) IN (ANNO_CORRENTE);

DBMS_OUTPUT.PUT_LINE ('ENTRATE CORSI DI ADDESTRAMENTO:'||CORSI);
```

```
SELECT SUM(PREZZO*QUANTITA) INTO VENDITA_PRODOTTI -- Calcolo guadagni dalla vendita dei prodotti
  FROM ACQUISTO_CONTIENE AC JOIN PRODOTTO PR ON PR.CODICE_PRODOTTO = AC.CODICE_PRODOTTO
  JOIN ACQUISTO A ON A.NUMERO_SCONTRINO = AC.NUMERO_SCONTRINO
    WHERE EXTRACT(YEAR FROM DATA_ACQUISTO) IN (2022);

DBMS_OUTPUT.PUT_LINE ('ENTRATE VENDITA PRODOTTI:'||VENDITA_PRODOTTI);

ENTRATE := VENDITA_ANIMALI + VENDITA_PRODOTTI + VISITE_PC + VISITE_VET + CORSI;
USCITE := ACQUISTO_ANIMALI + STIPENDI + TOTALE_CONSEGNE + COSTO_ADD + COSTO_VET + RIMBORSI; -- Calcolo le entrate e l

DBMS_OUTPUT.PUT_LINE ('ENTRATE TOTALI:' ||ENTRATE); -- Le stampo
DBMS_OUTPUT.PUT_LINE ('USCITE TOTALI:' ||USCITE);

IF(USCITE < ENTRATE) THEN DBMS_OUTPUT.PUT_LINE ('SEI IN POSITIVO!'); -- Determino l'andamento

ELSE
DBMS_OUTPUT.PUT_LINE ('SEI IN NEGATIVO!');
SELECT CF_DIPENDENTE INTO CF_PIU_PAGATO
FROM STIPENDIO
WHERE TOTALE_STIPENDIO = (SELECT MAX(TOTALE_STIPENDIO) FROM STIPENDIO)
AND ROWNUM = 1;

LICENZIAMENTO(CF_PIU_PAGATO);

END IF;
END;
/
```

2.6.3 CALCOLO_PUNTI

```
-- ==================================
-- CALCOLO PUNTI TESSERA DEL CLIENTE X
-- ==================================


CREATE OR REPLACE PROCEDURE CALCOLO_PUNTI(CF_CL IN CHAR)
IS
  PUNTI           NUMBER := 0;
  ANNO_CORRENTE   NUMBER;
BEGIN

  SELECT EXTRACT(YEAR FROM SYSDATE) INTO ANNO_CORRENTE
  FROM DUAL;

  SELECT * INTO PUNTI
  FROM
    (SELECT SUM(PREZZO*QUANTITA) FROM
    ACQUISTO_CONTIENE AC JOIN PRODOTTO PR ON PR.CODICE_PRODOTTO = AC.CODICE_PRODOTTO
    JOIN ACQUISTO_TESSERA AT ON AT.NUMERO_SCONTRINO = AC.NUMERO_SCONTRINO
    WHERE CF_CLIENTE = 'DNSMTT00P23F839I' AND EXTRACT(YEAR FROM DATA_ACQUISTO) IN (ANNO_CORRENTE));


  PUNTI:= FLOOR(PUNTI);
  DBMS_OUTPUT.PUT_LINE ('PUNTI : '||PUNTI);

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    PUNTI:= 0;

END;
/
```

2.6.4 DIPENDENTE_POCO_PRODUTTIVO

```
-- ============================
-- DIPENDENTE POCO PRODUTTIVO
-- ============================



CREATE OR REPLACE PROCEDURE DIPENDENTE_POCO_PRODUTTIVO
IS
  CF_DPP             CHAR(16);
  NUMERO_RICHIAMI   NUMBER;
BEGIN
  SELECT CF INTO CF_DPP FROM (
    SELECT
        SUM(EXTRACT(HOUR FROM (ORA_USCITA - ORA_ENTRATA))) AS ORE_DI_LAVORO,
        MAX(CF_DIPENDENTE) AS CF
    FROM E_PRESENTE
        GROUP BY CF_DIPENDENTE
        ORDER BY ORE_DI_LAVORO ASC)
  WHERE ROWNUM = 1;

  DBMS_OUTPUT.PUT_LINE ('DIPENDENTE CON MENO ORE: '||CF_DPP);

  SELECT COUNT(*) INTO NUMERO_RICHIAMI
    FROM RICHIAMO
      WHERE CF_DIPENDENTE = CF_DPP
        GROUP BY CF_DIPENDENTE;

  IF (NUMERO_RICHIAMI > 1) THEN
    DBMS_OUTPUT.PUT_LINE ('IL DIPENDENTE ' ||CF_DPP|| ' HA SUPERATO IL NUMERO MASSIMO DI RICHIAMI E PERTANTO VERRÀ LI
    LICENZIAMENTO(CF_DPP);

  ELSE INSERT INTO RICHIAMO(DATA_RICHIAMO,CF_DIPENDENTE) VALUES (TRUNC(SYSDATE,'MONTH'),CF_DPP);
  COMMIT;
  END IF;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    INSERT INTO RICHIAMO(DATA_RICHIAMO,CF_DIPENDENTE) VALUES (TRUNC(SYSDATE,'MONTH'),CF_DPP);
    COMMIT;

END;
/
```

2.6.5 LOTTERIA

```
-- ====================
-- LOTTERIA FORTUNATA!
-- ====================



CREATE OR REPLACE PROCEDURE LOTTERIA
IS
  CF_VINCITORE              CHAR(16);
  NOME_VINCITORE            CLIENTE.NOME%TYPE;
  COGNOME_VINCITORE         CLIENTE.COGNOME%TYPE;
  MESE_ATTUALE              VARCHAR2(10);
  ULTIMO_MESE               VARCHAR2(10);
BEGIN

  BEGIN
  SELECT MESE_VINCITA INTO ULTIMO_MESE
```

```
    FROM VINCITORE_LOTTERIA
        WHERE ROWNUM = 1;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      ULTIMO_MESE:= TO_DATE('01/01/1900','DD/MM/YYYY');
  END;

  SELECT TO_CHAR(SYSDATE,'MONTH') INTO MESE_ATTUALE
   FROM DUAL;

  IF(MESE_ATTUALE = ULTIMO_MESE AND ULTIMO_MESE IS NOT NULL) THEN
     DBMS_OUTPUT.PUT_LINE ('VINCITA GIA EFFETTUATA QUESTO MESE. TORNARE IL MESE PROSSIMO');

  ELSE

  SELECT CF_CLIENTE INTO CF_VINCITORE FROM
  (SELECT CF_CLIENTE
  FROM (SELECT CF_CLIENTE, COUNT(*) AS ACQUISTI_EFFETTUATI
    FROM ACQUISTO_TESSERA
      WHERE DATA_ACQUISTO >= SYSDATE - 30
        GROUP BY CF_CLIENTE)
        WHERE ACQUISTI_EFFETTUATI > 3
        ORDER BY DBMS_RANDOM.RANDOM
  )
            WHERE ROWNUM = 1;


  SELECT NOME,COGNOME INTO NOME_VINCITORE,COGNOME_VINCITORE
   FROM CLIENTE
    WHERE CF = CF_VINCITORE;

  INSERT INTO VINCITORE_LOTTERIA(NOME,COGNOME,CF_VINCITORE,MESE_VINCITA)
  VALUES(NOME_VINCITORE,COGNOME_VINCITORE,CF_VINCITORE,TO_CHAR(SYSDATE,'MONTH'));
  COMMIT;

  END IF;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20017,'NESSUN CLIENTE HA EFFETTUATO PIU DI 3 ACQUISTI QUESTO MESE :(');

END;
/
```

2.6.6 PROMOZIONE_AUTOMATICA

```
-- ================================================================================================
-- PROMOZIONE AUTOMATICA DEL 10% PER IL PRODOTTO PIU VENDUTO DEL MESE (DURATA PROMO: 1 SETTIMANA)
-- ================================================================================================


CREATE OR REPLACE PROCEDURE PROMOZIONE_AUTOMATICA IS

COD_PROD_PIU_VENDUTO          PRODOTTO.CODICE_PRODOTTO%TYPE;
COD_ULTIMA_PROMOZIONE         PROMOZIONE.CODICE_PROMOZIONE%TYPE;

BEGIN

  BEGIN
  SELECT CODICE_PROMOZIONE INTO COD_ULTIMA_PROMOZIONE
   FROM (SELECT CODICE_PROMOZIONE
    FROM PROMOZIONE
      ORDER BY CODICE_PROMOZIONE DESC) WHERE ROWNUM = 1;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      COD_ULTIMA_PROMOZIONE:= -1;
```

```
  END;


  SELECT CODICE INTO COD_PROD_PIU_VENDUTO
   FROM (SELECT SUM(QUANTITA) AS TOTALE,MAX(CODICE_PRODOTTO) AS CODICE FROM ACQUISTO_CONTIENE
    GROUP BY CODICE_PRODOTTO
    ORDER BY TOTALE DESC) WHERE ROWNUM = 1;


  INSERT INTO PROMOZIONE(CODICE_PROMOZIONE,DATA_INIZIO,DATA_FINE)
  VALUES(COD_ULTIMA_PROMOZIONE+1,SYSDATE+1,SYSDATE+8); -- PROMOZIONE ATTIVA DAL GIORNO DOPO PER TUTTA LA SETTIMANA
  COMMIT;

  INSERT INTO PROMOZIONE_RIGUARDA(CODICE_PROMOZIONE,CODICE_PRODOTTO,SCONTO_APPLICATO)
  VALUES(COD_ULTIMA_PROMOZIONE+1,COD_PROD_PIU_VENDUTO,0.10);
  COMMIT;
END;
/
```

2.6.7 STIPENDI

```
-- ====================================
-- STIPENDI -_-_-_- PROCEDURA SCHEDULER
-- ====================================

CREATE OR REPLACE PROCEDURE STIPENDI IS

    ULTIMO_STIPENDIO    NUMBER;
    CF_DIP              CHAR(16);
    IBAN                STIPENDIO.IBAN_DIPENDENTE%TYPE;
BEGIN

    FOR i IN (SELECT CF FROM DIPENDENTE) LOOP

    BEGIN

    SELECT CF INTO CF_DIP
    FROM DIPENDENTE
    WHERE CF = (i.CF);

    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20021,'IL DIPENDETE NON È STATO TROVATO NEL DATABASE');
    END;


    BEGIN

    SELECT MAX(TOTALE_STIPENDIO),MAX(IBAN_DIPENDENTE)  INTO ULTIMO_STIPENDIO,IBAN
    FROM STIPENDIO
    WHERE CF_DIPENDENTE = (i.CF)
    GROUP BY CF_DIPENDENTE;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            ULTIMO_STIPENDIO := 900;
    END;

    INSERT INTO STIPENDIO (CF_DIPENDENTE,DATA_STIPENDIO,IBAN_DIPENDENTE,TOTALE_STIPENDIO) VALUES (i.CF,SYSDATE,IBAN,U
    COMMIT;
    END LOOP;
END;
/
```

2.7 Scheduler

Lo scheduler effettua un JOB (nel nostro caso un blocco di codice PL/SQL) con una cadenza scelta dal programmatore

```
BEGIN DBMS_SCHEDULER.CREATE_JOB (
    JOB_NAME => 'STIPENDI_AUTOMATIZED',
    JOB_TYPE => 'PLSQL_BLOCK',
    JOB_ACTION =>  '
                        BEGIN
                            STIPENDI;
                        END;
                    ',
    START_DATE => TO_DATE('01-JAN-2022','DD-MM-YYYY'),
    REPEAT_INTERVAL => 'FREQ = MONTHLY',
    ENABLED => TRUE,
    COMMENTS => 'ASSEGNAZIONE AUTOMATICA STIPENDI');
END;

-- CANCELLAZIONE JOB DI STIPENDI_AUTOMATIZED
BEGIN
    DBMS_SCHEDULER.DROP_JOB ('STIPENDI_AUTOMATIZED');
END;
```

2.8 Data Control Language

Il data control language riguarda la gestione degli utenti e i permessi che vogliamo assegnare ad essi. Nel nostro Database DB_PETSHOP ha tutti i permessi possibili sul DB aziendale.

2.8.1 Permessi DB_PETSHOP

beginverbatim GRANT ALL PRIVILEGES TO DB_PETSHOP endverbatim

2.8.2 Permessi DIPENDENTE

```
    GRANT CONNECT, CREATE SESSION TO C##DIPENDENTE;
```

```
GRANT SELECT ON ALLEVAMENTO TO C##DIPENDENTE;
GRANT SELECT ON PROVIENE_DA TO C##DIPENDENTE;
GRANT SELECT ON REPARTO TO C##DIPENDENTE;
GRANT SELECT ON FORNITORE TO C##DIPENDENTE;
GRANT SELECT ON CONSEGNA TO C##DIPENDENTE;
GRANT SELECT ON PRODOTTO TO C##DIPENDENTE;
GRANT SELECT ON ACQUISTO TO C##DIPENDENTE;
GRANT SELECT ON ACQUISTO_CONTIENE TO C##DIPENDENTE;
GRANT SELECT ON VINCITORE_LOTTERIA TO C##DIPENDENTE;
GRANT SELECT ON CONSEGNA_CONTIENE TO C##DIPENDENTE;
GRANT SELECT ON CORSO_DI_ADDESTRAMENTO TO C##DIPENDENTE;
GRANT SELECT ON SI_ISCRIVE TO C##DIPENDENTE;
GRANT SELECT ON ADDESTRATORE TO C##DIPENDENTE;
GRANT SELECT ON DIPENDENTE TO C##DIPENDENTE;
GRANT SELECT ON TURNI_DI_LAVORO TO C##DIPENDENTE;
GRANT SELECT ON E_PRESENTE TO C##DIPENDENTE;
GRANT SELECT ON VETERINARIO TO C##DIPENDENTE;
GRANT SELECT ON VISITA_VETERINARIO TO C##DIPENDENTE;
GRANT SELECT ON VISITA_PETCARE TO C##DIPENDENTE;
GRANT SELECT ON PROMOZIONE TO C##DIPENDENTE;
GRANT SELECT ON PROMOZIONE_RIGUARDA TO C##DIPENDENTE;
GRANT SELECT ON PRENOTAZIONE TO C##DIPENDENTE;

GRANT SELECT, INSERT, UPDATE, DELETE ON CLIENTE TO C##DIPENDENTE;
GRANT SELECT, INSERT, UPDATE, DELETE ON ANIMALE TO C##DIPENDENTE;
GRANT SELECT, INSERT, UPDATE, DELETE ON ADOTTA TO C##DIPENDENTE;
GRANT SELECT, INSERT, UPDATE, DELETE ON RESTITUISCE TO C##DIPENDENTE;
GRANT SELECT, INSERT, UPDATE, DELETE ON ACQUISTO TO C##DIPENDENTE;
GRANT SELECT, INSERT, UPDATE, DELETE ON ACQUISTO_CONTIENE TO C##DIPENDEN
GRANT SELECT, INSERT, UPDATE, DELETE ON VISITA_PETCARE TO C##DIPENDENTE;
GRANT SELECT, INSERT, UPDATE, DELETE ON PRENOTAZIONE TO C##DIPENDENTE;

GRANT SELECT ON ANIMALI_DISPONIBILI TO C##DIPENDENTE;
GRANT SELECT ON PRENOTAZIONI_PETCARE TO C##DIPENDENTE;

GRANT EXECUTE ON CALCOLO_PUNTI TO C##DIPENDENTE

GRANT CREATE VIEW TO C##DIPENDENTE;
GRANT CREATE PROCEDURE TO C##DIPENDENTE;
```

2.8.3 Permessi CLIENTE

```
GRANT CONNECT, CREATE SESSION TO C##CLIENTE;
GRANT SELECT ON ALLEVAMENTO TO C##CLIENTE;
GRANT SELECT ON PROVIENE_DA TO C##CLIENTE;
GRANT SELECT ON REPARTO TO C##CLIENTE;
GRANT SELECT ON PRODOTTO TO C##CLIENTE;
GRANT SELECT ON CORSO_DI_ADDESTRAMENTO TO C##CLIENTE;
GRANT SELECT ON SI_ISCRIVE TO C##CLIENTE;
GRANT SELECT ON ADDESTRATORE TO C##CLIENTE;
GRANT SELECT ON VETERINARIO TO C##CLIENTE;
GRANT SELECT ON VISITA_VETERINARIO TO C##CLIENTE;
GRANT SELECT ON VISITA_PETCARE TO C##CLIENTE;
GRANT SELECT ON PROMOZIONE TO C##CLIENTE;
GRANT SELECT ON PROMOZIONE_RIGUARDA TO C##CLIENTE;
GRANT SELECT ON PRENOTAZIONE TO C##CLIENTE;

GRANT SELECT ON ANIMALI_DISPONIBILI TO C##CLIENTE;



GRANT CREATE VIEW TO C##CLIENTE;
GRANT CREATE PROCEDURE TO C##CLIENTE;
```

2.8.4 Permessi VETERINARIO

beginverbatim GRANT CONNECT, CREATE SESSION TO CVET; GRANT SE-
LECT ON ALLEVAMENTO TO CVET; GRANT SELECT ON PROVIENE$_DATOCVET$; G
GRANT SELECT, INSERT, UPDATE, DELETE ON VISITA$_VETERINARIOTOCVE$
GRANT SELECT ON PRENOTAZIONI$_VETTOCVET$;
GRANT CREATE VIEW TO CVET;
GRANT CREATE PROCEDURE TO CVET;
endverbatim

2.8.5 Permessi ADDESTRATORE

```
GRANT CONNECT, CREATE SESSION TO C##ADDESTRATORE;
GRANT SELECT ON REPARTO TO C##ADDESTRATORE;
```

```
GRANT SELECT ON CORSO_DI_ADDESTRAMENTO TO C##ADDESTRATORE;
GRANT SELECT ON SI_ISCRIVE TO C##ADDESTRATORE;
GRANT SELECT ON ADDESTRATORE TO C##ADDESTRATORE;
GRANT SELECT ON VETERINARIO TO C##ADDESTRATORE;

GRANT SELECT, INSERT, UPDATE, DELETE ON SI_ISCRIVE TO C##ADDESTRATORE;

GRANT SELECT ON ISCRIZIONI_CORSI TO C##ADDESTRATORE;

GRANT CREATE VIEW TO C##ADDESTRATORE;

GRANT CREATE PROCEDURE TO C##ADDESTRATORE;
```