TortoiseHospital

PROGETTAZIONE E SVILUPPO DI UNA BASE DI DATI RELAZIONALE PER LA GESTIONE DEI RICOVERI DI TARTARUGHE MARINE NEI CENTRI DI RECUPERO NAZIONALI

Eduardo Gaudiosi

N86003812

Pagina lasciata volutamente bianca.

INDICE

1. DESCRIZIONE DEL PROGETTO

Il progetto si concentra sulla creazione di un sistema informativo per la gestione dei centri di recupero nazionali per tartarughe marine. Questo sistema comprenderà una base di dati relazionale e un'applicazione Java con un'interfaccia utente grafica (GUI) per semplificare la gestione delle tartarughe e dei dati dei centri. Sarà possibile registrare l'ingresso e la riammissione delle tartarughe, assegnando loro identificativi univoci e aggiornando le informazioni sulla loro salute. Il sistema offrirà inoltre la possibilità di visualizzare storici dettagliati delle tartarughe, nonché statistiche mensili e annuali per monitorare il numero di tartarughe accolte e il loro stato di salute. Inoltre, consentirà la gestione dei dati dei centri e del personale addetto.

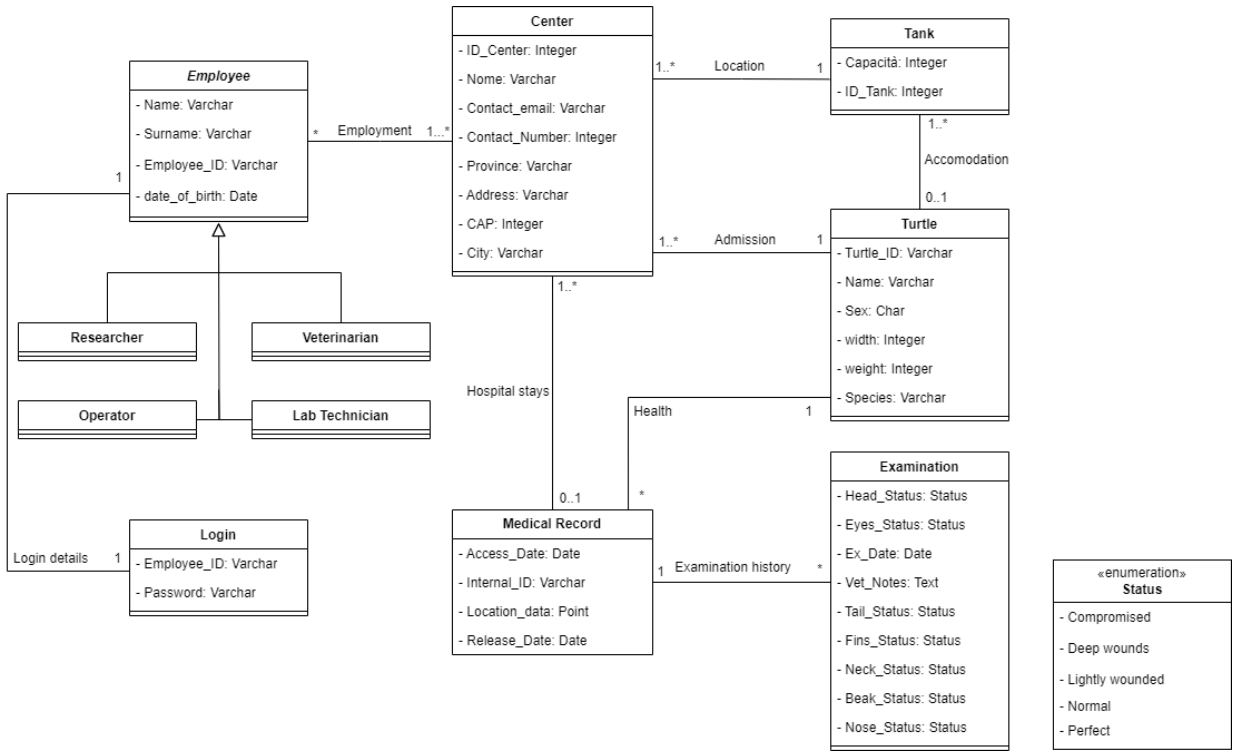
Questo progetto risponde alla necessità di migliorare la gestione e la tracciabilità delle tartarughe marine presso i centri di recupero, contribuendo alla loro conservazione e alla ricerca scientifica. L'utilizzo di una base di dati relazionale e un'interfaccia utente intuitiva renderà più efficienti le operazioni quotidiane nei centri, consentendo una registrazione accurata dei dati e una valutazione completa dello stato di salute delle tartarughe. Inoltre, le statistiche forniranno una panoramica utile per la pianificazione e l'analisi delle attività di recupero. Il progetto consiste nello sviluppo di un sistema informativo completo per la gestione del ricovero di tartarughe marine presso i centri di Recupero Nazionali. Il sistema comprenderà una base di dati relazionale per la memorizzazione dei dati fondamentali dei centri e delle tartarughe, nonché un'applicazione Java con interfaccia grafica utente (GUI), basata su JavaFX, per la gestione delle operazioni di registrazione, accesso, riammissione e monitoraggio delle tartarughe marine.

2. PROGETTAZIONE CONCETTUALE

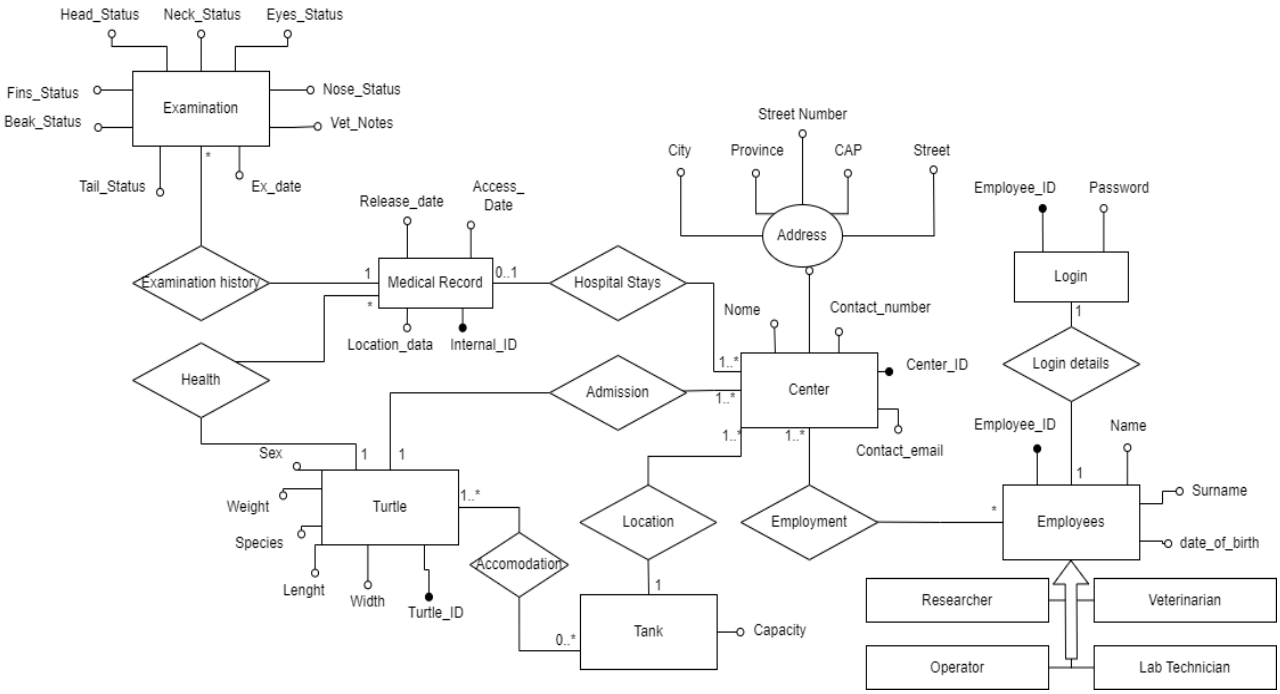
2.1. INTRODUZIONE

Nel capitolo, si comincerà la progettazione della base di dati al livello di astrazione più alto. Partendo dal risultato dell'analisi dei requisiti da soddisfarre, si arriverà ad uno schema concettuale completamente indipendente dalla struttura dei dati e dall'implementazione fisica. All'interno di tale schema concettuale, che sarà rappresentato tramite un class diagram UML e un diagramma Entità-Relazioni, saranno messe in evidenza le entità (concetti rilevanti) e le relazioni che intercorrono tra esse. Inoltre, saranno delineati eventuali vincoli da imporre.

2.2. CLASS DIAGRAM



2.3. DIAGRAMMA ENTITÀ-RELAZIONE



2.4. RISTRUTTURAZIONE DELLO SCHEMA CONCETTUALE

Per poter garantire che il diagramma delle classi possa essere tradotto facilmente in schemi relazionali e per ottimizzare l'efficienza dell'implementazione dello stesso, si procede a una completa ristrutturazione dello schema concettuale. Completata l'operazione, il class diagram sarà semplificato eliminando elementi che potrebbero complicare la traduzione in strutture dati relazionali e dunque fornire problemi durante l'implementazione del sistema.

2.4.1. ANALISI ATTRIBUTI STRUTTURATI

Allo stato attuale dello schema, esiste un singolo attributo strutturato, evidenziato nel diagramma Entità-Relazioni, di nome "ADDRESS", composto dagli attributi "City", "Province", "CAP", "Street" e "Street Number". Rispetto al raggruppare tutto all'interno di "ADDRESS", si è ritenuto più adeguato incorporare all'interno di "CENTER" tutti gli attributi sopra citati e fondendone due in particolare, ossia Street e Street Number, che saranno rappresentati un'unica stringa. Si è deciso contro la creazione di una weak entity che comprendesse gli attributi in questione poichè i centri da inserire dovrebbero essere nell'ordine delle decine o centinaia, e le interrogazioni sulla tabella "CENTER" sono state stimate in "una o due interrogazioni per ogni avvio del software di gestione", rendendo la creazione di una weak entity superflua.

2.4.2. ANALISI ATTRIBUTI MULTIPLI

Non sono presenti attributi multipli da eliminare.

2.4.3. ANALISI GERARCHIE DI SPECIALIZZAZIONE

Si procede all'eliminazione delle sottoclassi e alla loro compressione verso l'alto. È possibile creare dunque un attributo "ProfileType" all'interno dell'entità "EMPLOYEE" che specifichi il tipo di figura professionale.

2.4.4. ANALISI CLASSI DI ASSOCIAZIONE

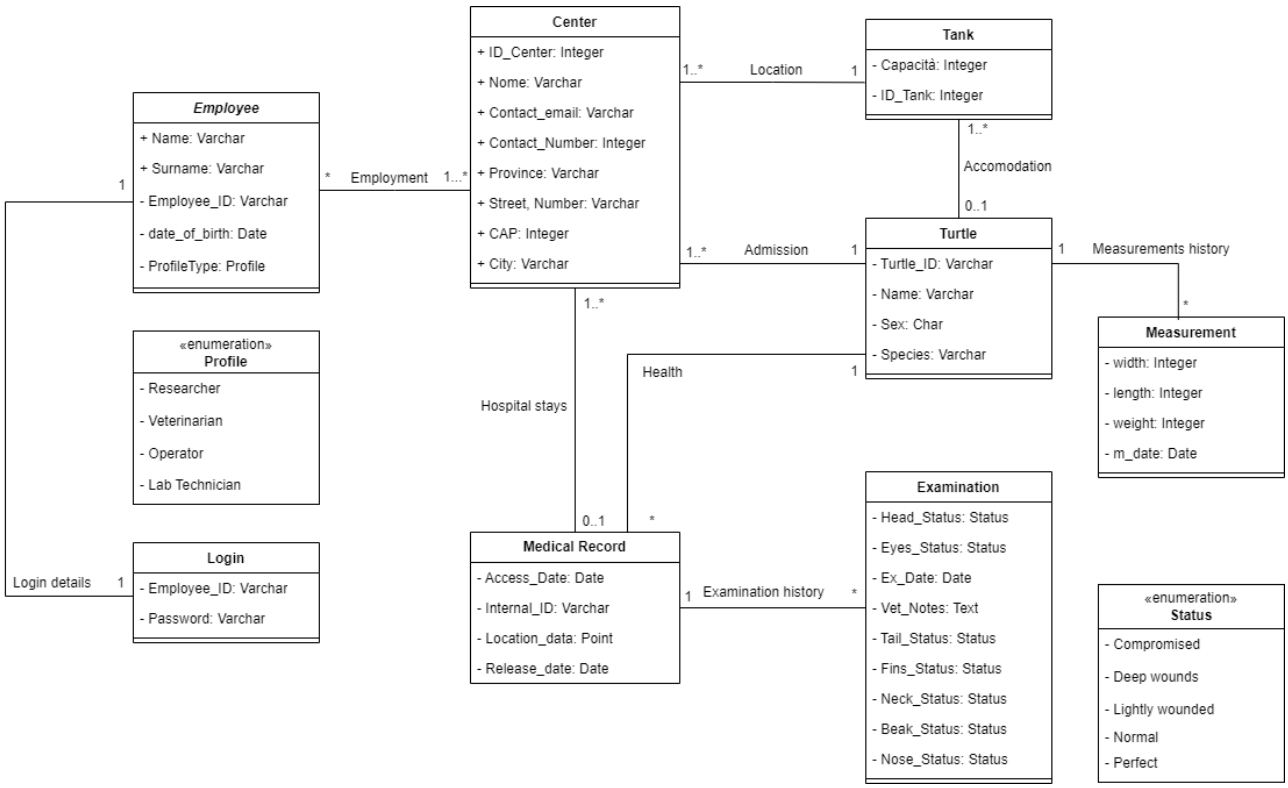
Non sono presenti classi di associazione da eliminare.

2.4.5. ALTRO

Al fine di poter avere delle interrogazioni più agevoli nella comunicazione con il database e di incorporare il caso "crescita di una tartaruga all'interno del centro", si procede a creare la tabella "MEASUREMENTS", che contiene le informazioni riguardo le tartarughe che potrebbero variare nel tempo, come width, length e weight, espressi in cm, cm e Kg. Per quanto riguarda l'associazione ridondante di nome "hospital stays", si sceglie di lasciarla al fine di semplificare la creazione di interrogazioni per le statistiche riguardanti gli specifici centri. Stessa scelta è stata fatta anche per l'associazione "Admission", al fine di rendere più agevole l'interrogazione "Tartarughe in centro X".

2.5. CLASS DIAGRAM AGGIORNATO

Allo stato delle modifiche effettuate durante la ristrutturazione del class diagram, il class diagram viene così aggiornato:



2.6. DIZIONARIO DELLE CLASSI

CLASSE	DESCRIZIONE CLASSE	ATTRIBUTI
Center	Struttura o luogo fisico adibito a Centro Nazionale di Recupero delle Tartarughe Marine	ID_Center (string): Identificativo del centro. Nome (string): Nome del centro. City (String): Città del Centro. CAP (String): CAP della zona del Centro. Address (String): Indirizzo del Centro. Si è preferito salvare entrambe le informazioni in un'unica stringa. Province (String): Provincia del Centro. Contact_Number (Integer): Numero di telefono (cellulare o fisso) per contattare il Centro. Contact_email (String): Email del centro. Dismissed (Boolean): Indica se il centro è ancora operativo.

Employee	Persona impiegata in uno dei Centri Nazionali di Recupero delle Tartarughe Marine	Name (string): nome dell'impiegato. Surname (string): cognome dell'impiegato. Employee_ID (string): Matricola dell'impiegato. ProfileType (Profile): Identifica il ruolo della persona all'interno del proprio centro. Date_of_birth (Date): Data di nascita dell'impiegato
Examination	Visita effettuata da un veterinario ad una tartaruga	Head_Status (Status): Stato della testa della tartaruga marina Eyes_Status (Status): Stato degli occhi della tartaruga marina Nose_Status (Status): Stato del naso della tartaruga marina Beak_Status (Status): Stato del becco della tartaruga marina Neck_Status (Status): Stato del collo della tartaruga marina Fins_Status (Status): Stato delle pinne della tartaruga marina Tail_Status (Status): Stato della coda della tartaruga marina Vet_Notes (Text): Note del veterinario. Sono usate per specificare meglio la diagnosi, eventuali tempi di recupero stimati, farmaci somministrati e qualsiasi cosa il veterinario possa ritenere utile avere in cartella clinica. Ex_Date (Date): Data della visita effettuata dal veterinario ad una tartaruga.
Login	Dati di login degli impiegati	Password (string): Hash della password dell'impiegato. Employee_ID (string): Matricola dell'impiegato.
Measurement	Misurazioni effettuate ad una tartaruga.	Width (Float): Larghezza massima del carapace della tartaruga (misura da prendere al centro del carapace) Length (Float): Lunghezza massima del carapace della tartaruga (misura da prendere al centro del carapace)

		Weight (Float): Peso della tartaruga M_date (Date): Data della misurazione effettuata
Medical Record	Singola cartella clinica appartenente ad una tartaruga	Internal_ID (String): Stringa che identifica la specifica cartella clinica di una tartaruga trattata in uno dei centri. Access_Date (Date): Data del ritrovamento della tartaruga. Location_Data (point): Latitudine e longitudine del punto di ritrovamento della tartaruga ferita. Release_Date (Date): Data del rilascio della tartaruga in oceano.
Tank	Vasche presenti all'interno del Centro di Recupero.	Capacità (Integer): Capacità massima della singola vasca ID_Tank (String): Identificativo della vasca.
Turtle	Tartaruga attualmente in cura o già curata da uno dei centri operativi.	Turtle_ID (String): Identificativo della tartaruga. Stampato anche sulla targhetta metallica che la tartaruga possiede. Name (String): Nome assegnato alla tartaruga. Species (String): Specie della tartaruga. Sex (Char): Sesso della tartaruga

2.7 DIZIONARIO DELLE ASSOCIAZIONI

NOME	DESCRIZIONE	CLASSI COINVOLTE
Accomodation	Lega una tartaruga alla vasca in cui è presente.	**Tank [1] accomodates:** indica le tartarughe presenti all'interno della vasca. **Turtle [0..Capacità]:** indica la vasca in cui sono presenti le tartarughe.
Admission	Esprime il rapporto che lega le tartarughe malate al centro in cui sono ospitate al fine di avere cure.	**Center [1] hosts:** indica le tartarughe ospitate all'interno del centro. **Turtle [0..*] accesses:** indica il centro in cui sono presenti una o più tartarughe.

Employment	Esprime il rapporto che lega un lavoratore al centro in cui lavora.	**Center [1] employes**: indica gli impiegati che lavorano all'interno del centro. **Employee [0..*] of:** indica il centro in cui lavorano uno o più impiegati.
Examination history	Lega una cartella medica alle visite fatte alla tartaruga	**Medical record [1]** is **constituted by:** indica le visite da cui è composta una cartella medica. **Examination [0..*] composes:** indica la cartella medica a cui vengono associate le visite.
Health	Lega una tartaruga alle proprie cartelle mediche.	**Turtle [1] has:** indica le cartelle mediche associate alla tartaruga. **Medical record [0..*] belongs to:** indica la tartaruga a cui è associata la cartella medica
Hospital Stays	Lega un centro alle cartelle mediche delle tartarughe che sono state presenti nel centro stesso. Usata a fini statistici.	**Center [1] archives:** indica tutte le cartelle mediche associate ad un singolo centro. **Medical record [0..*] is archived:** Indica il centro che ha archiviato la specifica cartella medica.
Location	Lega un centro alle vasche in esso presenti.	**Center [1] includes:** indica le vasche presenti nel centro. **Tank [0..*] located in:** indica il centro in cui sono presenti le vasche.
Login Details	Lega ogni impiegato con i propri dati di accesso	**Employee [1] has:** indica i dati di login associati all'impiegato.
Measurements history	Lega una tartaruga alle misurazioni che le sono state effettuate	**Turtle [1] has:** indica tutte le misurazioni associate ad una singola tartaruga.

2.8 DIZIONARIO DEI VINCOLI

NOME	DESCRIZIONE
appraisal_Update	Ogni "measurement" ha valore come singola misurazione e non potrà essere modificata in date posteriori.
exams_Consistency	Ogni "examination" può essere svolta solo ed esclusivamente in una data posteriore alla data di accesso della tartaruga.
exams_Update	Ogni "examination" ha valore di singola visita e non potrà essere dunque modificabile in date posteriori.
legit_Age	Ogni impiegato deve avere minimo 15 anni per poter essere registrato come tale (età minima per l'accesso al mondo del lavoro, legge 677/1977)
legit_CAP	Il CAP deve essere un valore numerico di 5 cifre.
legit_Capacity	La capacità di una vasca è sempre un numero positivo superiore a 0.
legit_Email	Le email devono essere necessariamente in formato **@**.**.
maxOneFirstVisit	Ogni cartella medica può avere una singola prima visita.
released_Turtle	Ogni cartella medica la cui "release_Date" risulti non NULL risulterà automaticamente chiusa, e dunque non modificabile.
tank_Consistency	Ogni "Tank" non può contenere più di "tank.capacity" tartarughe.
unique_Login	Ogni employee può avere solamente una combinazione di dati di accesso per il login.

3 PROGETTAZIONE LOGICA

Si tratta ora la fase successiva della progettazione della base di dati: la progettazione logica, ad un livello di astrazione più basso rispetto al precedente. Si andrà dunque a tradurre lo schema concettuale ristrutturato in uno schema logico secondo il tipo di struttura dei dati prescelto (relazionale puro). Le chiavi primarie saranno indicate con una singola sottolineatura, le chiavi esterne con una sottolineatura doppia.

Nella tabella "Tank" si procede all'aggiunta di una chiave surrogata per poter identificare univocamente una vasca. Tale valore farà da chiave esterna nella tabella "turtle".

Nella tabella "Examination" si procede all'aggiunta di un boolean "first_visit" e "avghealth" (quest'ultimo sarà compilato in automatico dal backend Java) per poter più agevolmente creare delle statistiche mensili ed annuali.

CLASSE	SCHEMA LOGICO
Center	(**ID_Center**, Name, Contact_email, Contact_Number, Province, Address, CAP, City)
Employee	(Name, Surname, **Employee_ID**, ProfileType, date_of_birth, Center_ID) Center_ID → Center.ID_Center
Employment **(Tabella Ponte)**	(Employee_ID, Center_ID) Center_ID → Center.ID_Center Employee_ID → Employee.Employee_ID
Examination	(Internal_ID, Head_Status, Eyes_Status, Ex_Date, Vet_Notes, Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status, first_visit) Internal_ID → Medical_Record.Internal_ID
Login	(Employee_ID, Password) Employee_ID → Employee.Employee_ID
Measurements	(Turtle_ID, width, length, weight, m_date) Turtle_ID → Turtle.Turtle_ID
Medical_Record	(**Internal_ID**, Access_Date, Location_Data, Release_Date, Center_ID, Turtle_ID) Center_ID → Center.ID_Center Turtle_ID → Turtle.Turtle_ID
Tank	(ID_Tank, Capacity, Center_ID, **Tank_Surrogate**) Center_ID → Center.ID_Center
Turtle	(**Turtle_ID**, Name, Sex, Species, Center_ID, Tank_ID) Center_ID → Center.ID_Center Tank_ID → Tank.Tank_Surrogate

4 PROGETTAZIONE FISICA

Si tratta ora l'implementazione fisica del database. Per l'implementazione è stato scelto il DBMS PostgreSQL. Poichè PostgreSQL non implementa le ASSERTIONS, si useranno dei trigger per la creazione di vincoli. Inoltre, l'utilizzo di PostgreSQL obbliga alla creazione di coppie trigger/function: queste ultime verranno lanciate dal trigger a loro associato; pertanto, nella documentazione verranno riportate insieme alla creazione del trigger stesso.

Sia all'interno dei trigger che delle tabelle si è preferito evitare sia l'utilizzo di SEQUENCES che di SERIAL poiché si è riscontrato, nel caso di INSERT andati male, un avanzo sia della sequenza che del serial, lasciando di fatto dei numeri non utilizzati all'interno del database. Si è preferito pertanto lasciare a dei trigger la gestione della generazione di ID.

4.1 DEFINIZIONE TABELLA CENTER

Nota: Si è proceduto a dichiarare l'attributo CAP come CHAR(5) e non come INT4 per prevenire la cancellazione degli 0 precedenti ad una cifra diversa dallo 0 in CAP come quelli del Lazio (ex. 001XX).

```
CREATE TABLE center (
    -- Identificatore univoco del centro (generato dal trigger
new_CenterID)
    ID_Center CHAR(10) PRIMARY KEY,

    -- Nome del centro (massimo 50 caratteri, obbligatorio)
    name VARCHAR(50) NOT NULL,

    -- Indirizzo email di contatto (massimo 100 caratteri, obbligatorio)
- Legato al vincolo legit_Emails
    contact_email VARCHAR(100) NOT NULL,

    -- Numero di contatto del centro (massimo 20 caratteri)
    contact_number VARCHAR(20),

    -- Provincia del centro (massimo 2 caratteri, obbligatorio)
    province VARCHAR(2) NOT NULL,

    -- Indirizzo fisico del centro (massimo 30 caratteri, obbligatorio)
    address VARCHAR(30) NOT NULL,

    -- Codice di avviamento postale (massimo 5 caratteri, obbligatorio) -
Legato al vincolo legit_CAP
    CAP CHAR(5) NOT NULL,

    -- Città del centro (massimo 15 caratteri, obbligatoria)
    city VARCHAR(15) NOT NULL,

    dismissed BOOLEAN DEFAULT FALSE,

/* Legit CAP - Un CAP valido deve contenere 5 caratteri, tutti numerici
(Si presuppone un CAP italiano) */
    CONSTRAINT legit_CAP
        CHECK (CAP ~* '^\d{5}$'),
/* Legit Emails - Una mail valida può contenere 100 caratteri in totale,
e deve rispettare il formato "**@**.**", il tutto senza contenere
caratteri speciali. */

    CONSTRAINT legit_Email
    CHECK (contact_email ~* '^[a-zA-Z0-9_+&*-]+(?:\.[a-zA-Z0-9_+&*-
]+)*@(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]{2,}$')

);
```

4.1.1 FUNZIONE E TRIGGER SU TABELLA CENTER

Il trigger e la funzione di sotto implementati si occupano della creazione di un nuovo ID per ogni centro che viene aggiunto, in modo del tutto automatico. Risulta così molto più semplice eseguire la query INSERT per l'aggiunta di un nuovo centro. Esempio di id generati dal trigger: CTR0000001

IMPORTANTE: All'interno della funzione new_CenterId(), nel file SQL, è presente un "FOR" commentato. Il FOR in questione copre il caso in cui un Centro possa essere cancellato. Questo database è stato creato volutamente ignorando il caso "Cancellazione di un centro" poiché, anche nel caso della dismissione di un centro, l'amministratore dell'applicazione avrebbe tutto l'interesse a non perdere i dati sanitari delle tartarughe raccolti durante il periodo in cui il centro è stato operativo. Conservando i dati ma non il centro, si avrebbe una violazione del vincolo di integrità referenziale, assegnando cartelle mediche ad un centro non esistente, o peggio, ad un centro aperto successivamente a cui è stato riassegnato l'identificativo del centro non più operativo. Pertanto, un centro a cui sono assegnate cartelle mediche non sarà cancellabile, e per definirne l'operatività si userà un boolean all'interno della tabella stessa.

```
CREATE FUNCTION new_CenterId()
RETURNS TRIGGER AS $$
DECLARE
    counter INT;
    ctr_id VARCHAR;
BEGIN
    -- Caso base: se non esiste alcun ID per il centro, assegna
CTR0000001
    IF((SELECT ID_Center FROM center ORDER BY ID_Center LIMIT 1) IS NULL)
THEN
        ctr_id := 'CTR0000001';
    ELSE
        -- Caso generico: recupera il centro con id più grande
numericamente, cast a INT e assegna il valore trovato ad un counter
        counter := (substring((SELECT ID_Center FROM center ORDER BY
ID_Center DESC LIMIT 1) FROM 4 FOR 7))::INT;

        -- Se l'esecuzione della funzione arriva a questo punto, è
ragionevole pensare che esista un ID assegnato ad ogni centro
        -- fino a CTR{counter}. Di conseguenza, si incrementa di uno il
counter e si restituisce l'ID unico così trovato.
        ctr_id := 'CTR' || lpad((counter+1)::VARCHAR, 7, '0');

    END IF;
        NEW.ID_center := ctr_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER new_Center
BEFORE INSERT ON Center
FOR EACH ROW
EXECUTE FUNCTION new_CenterId();
```

4.2 DEFINIZIONE TABELLA EMPLOYEE

Ogni riga di Employee definisce in modo completo il profilo di un impiegato di uno o più centri. All'assunzione, la matricola viene assegnata in modo automatico in base al ruolo. La matricola viene poi automaticamente aggiornata all'aggiornarsi del ruolo, nell'eventuale caso di promozioni, grazie agli "ON UPDATE CASCADE".

```
-- Definizione del dominio 'Profile', utilizzato dalla tabella 'Employee'
per identificare il ruolo di ciascun impiegato
CREATE DOMAIN Profile AS CHAR(1)
    CHECK(VALUE='V' OR VALUE='R' OR VALUE='O' OR VALUE='L');

CREATE TABLE employee (
    -- Identificatore unico del dipendente (Generato dal trigger
employeeInsert)
    employee_ID CHAR(10) PRIMARY KEY,

    -- Nome del dipendente (massimo 50 caratteri, obbligatorio)
    name VARCHAR(50) NOT NULL,

    -- Cognome del dipendente (massimo 50 caratteri, obbligatorio)
    surname VARCHAR(50) NOT NULL,

    -- Data di nascita del dipendente (obbligatoria)
    date_of_birth DATE NOT NULL,

    -- Profilo del lavoratore nel centro (Vincolato al dominio 'Profile',
obbligatorio)
    ProfileType Profile NOT NULL,

    -- Vincolo per garantire che l'età del dipendente sia almeno 15 anni
    CONSTRAINT legit_Age
        CHECK (date_of_birth <= CURRENT_DATE - INTERVAL '15 years')
);
```

4.1.2 FUNZIONI E TRIGGER TABELLA EMPLOYEE

I trigger e le funzioni di sotto implementati si occupano della creazione di un nuovo ID per ogni impiegato che viene aggiunto, in modo del tutto automatico. Il trigger viene lanciato anche per gli UPDATE, nell'eventualità di promozioni o cambi di ruolo del personale. Si è preferito separare il loop dalla funzione con il CASE per favorire la leggibilità del codice.

Formato di uno degli id generati dal trigger: LBT0000001

```
CREATE FUNCTION loopEmployee(IN profileT CHAR)
RETURNS VARCHAR AS $$
DECLARE
    counter INT;
    final_ID VARCHAR;
    curs2 CURSOR FOR SELECT employee_ID FROM employee WHERE employee_ID
LIKE profileT || '%' ORDER BY employee_ID;
BEGIN
    -- Inizializzazione del contatore
    counter := 0;

    -- Scorrimento degli impiegati con identificatori dello stesso ruolo
    FOR recordvar IN curs2 LOOP
        counter := counter + 1;

        -- Se l'identificatore non corrisponde al contatore, vuol dire
che l'impiegato possessore di quell'id è stato cancellato.
        -- Si procede dunque alla riassegnazione dell'identificatore
all'impiegato che si sta inserendo in database
        IF(substring(recordvar.employee_ID FROM 4 FOR 7)::INT <> counter)
THEN
            final_ID := substring(recordvar.employee_ID FROM 1 FOR 3) ||
lpad(counter::VARCHAR, 7, '0');
            RETURN final_ID;
        END IF;
    END LOOP;

    -- Se nessun identificatore è stato generato all'interno del loop,
crea il primo identificatore disponibile
    final_ID := substring((SELECT employee_ID FROM employee WHERE
employee_ID LIKE profileT || '%' LIMIT 1) FROM 1 FOR 3) ||
lpad((counter+1)::VARCHAR, 7, '0');
    RETURN final_ID;
END;
$$ LANGUAGE plpgsql;

CREATE FUNCTION newEmployeeId()
RETURNS TRIGGER AS $$
DECLARE
    emp_id VARCHAR;
BEGIN
    -- Verifica se non ci sono impiegati con l'identificatore specificato
```

```
    IF((SELECT employee_ID FROM employee WHERE employee_ID LIKE
new.profiletype || '%' ORDER BY employee_ID DESC LIMIT 1) IS NULL) THEN
        -- Se non ci sono impiegati con l'identificatore specificato,
crea il primo identificatore disponibile
        CASE new.profiletype
            WHEN 'V' THEN emp_id := 'VET' || lpad('1', 7, '0');
            WHEN 'R' THEN emp_id := 'RES' || lpad('1', 7, '0');
            WHEN 'O' THEN emp_id := 'OPR' || lpad('1', 7, '0');
            WHEN 'L' THEN emp_id := 'LBT' || lpad('1', 7, '0');
        END CASE;

        -- Assegna il nuovo identificatore al nuovo impiegato
        NEW.employee_id := emp_id;
        RETURN NEW;
    END IF;

    -- Se ci sono impiegati con l'identificatore specificato, utilizza la
funzione loopEmployee per generare il nuovo identificatore
    emp_ID := loopEmployee(NEW.profiletype);

    -- Concatenazione della stringa a seconda del ProfileType al numero
identificativo, con aggiunta di padding laddove necessario
    NEW.employee_ID := emp_id;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire la funzione automaticamente prima
dell'inserimento in 'employee'
CREATE TRIGGER new_Employee
BEFORE INSERT ON employee
FOR EACH ROW
EXECUTE FUNCTION newEmployeeId();
```

4.2 DEFINIZIONE TABELLA EMPLOYMENT

Si procede a creare la tabella ponte employment. Si è dato per scontato all'interno del sistema che ogni impiegato possa lavorare in uno o più centri, o eventualmente in nessuno. Un impiegato non associato a nessun centro è un impiegato andato oramai in pensione oppure un ex impiegato con cui si è interrotto il rapporto di lavoro, di cui si preferisce tenere i dati in database per motivi giuridici o statistici.

```
CREATE TABLE employment (
    -- Identificativo del centro (massimo 10 caratteri, obbligatorio)
    center_ID CHAR(10) NOT NULL,

    -- Identificativo del dipendente (massimo 10 caratteri, obbligatorio)
    employee_ID CHAR(10) NOT NULL,

    -- Vincoli di chiave esterna per garantire l'integrità referenziale
con le tabella 'employee' e
    -- 'center', con specifica dell'opzione ON DELETE CASCADE per
eliminare automaticamente le
    -- relazioni tra le tabelle nel caso di eliminazione di un centro o
di un impiegato dal database
    FOREIGN KEY (center_ID) REFERENCES center(ID_Center) ON DELETE
CASCADE,
    FOREIGN KEY (employee_ID) REFERENCES employee(employee_ID) ON DELETE
CASCADE
);
```

4.3 DEFINIZIONE TABELLA LOGIN

Si procede, tramite comando, ad installare l'estensione di PostgreSQL di nome pgcrypto, al fine di non avere in storage permanente le password in chiaro. Nella seguente tabella, si è scelto di usare BLOWFISH per garantire la sicurezza delle password contro un eventuale attacco bruteforce. Sono forniti, a fine pagina in un commento, degli esempi di INSERT e di SELECT come template per l'uso dell'estensione in questione.

```
CREATE EXTENSION pgcrypto;

CREATE TABLE login (
    -- Identificativo del dipendente (massimo 10 caratteri, obbligatorio)
    employee_ID CHAR(10) NOT NULL,

    -- Hash della password associata al dipendente (60 caratteri, fisso,
da ricavare usando la funzione crypt)
    password VARCHAR(60) NOT NULL,

    -- Vincolo per garantire l'unicità dell'identificativo del dipendente
    UNIQUE(employee_ID),

    -- Vincolo di chiave esterna per garantire l'integrità referenziale
con la tabella 'employees'
    FOREIGN KEY (employee_ID) REFERENCES employee (employee_ID) ON DELETE
CASCADE ON UPDATE CASCADE

);




/*

TEMPLATE DI SELECT ED INSERT PER L'USO DI PGCRYPTO


INSERT INTO login(employee_ID, password)
VALUES ('VET0000001', crypt('placeholder', gen_salt('bf'))) ;


VALIDAZIONE DI UNA PASSWORD PER IL LOGIN:

SELECT * FROM login
WHERE employee_id='VET0000001'
AND password=crypt('password da validare', password);
*/
```

4.4 DEFINIZIONE TABELLA TANK

Si ritiene necessaria la creazione di una surrogate key “surrogate_tank” per facilitare la creazione di eventuali statistiche a partire dall’identificativo di una vasca. La creazione di una surrogate key rende necessario un vincolo UNIQUE sulle combinazioni di tank_ID e center_ID. La creazione sia del tank_ID che del surrogate_tank sarà gestita da trigger in modo da facilitare le operazioni di INSERT.

```
CREATE TABLE tank (
    -- Identificativo unico della vasca (massimo 2 byte, obbligatorio)
    tank_ID INT2 NOT NULL,

    -- Identificativo del centro associato alla vasca (massimo 10
caratteri, obbligatorio)
    center_ID CHAR(10) NOT NULL,

    -- Capacità massima di tartarughe nella vasca (massimo 2 byte,
obbligatoria)
    capacity INT2 NOT NULL,

    -- Identificativo unico della vasca (utilizzato come chiave primaria)
    surrogate_tank INT4 PRIMARY KEY,

    -- Vincolo per garantire l'unicità della combinazione di tank_ID e
center_ID
    UNIQUE (tank_ID, center_ID),

    -- Vincolo per garantire che la capacità della vasca sia maggiore o
uguale a 0
    CONSTRAINT legit_Capacity
        CHECK (capacity >= 0),

    -- Vincolo di chiave esterna per garantire l'integrità referenziale
con la tabella 'center' e
    -- specifica l'opzione ON DELETE CASCADE per eliminare
automaticamente le vasche associate
    -- a un centro quando il centro viene eliminato
    FOREIGN KEY (center_ID) REFERENCES center(ID_Center) ON DELETE
CASCADE
);
```

4.4.1 FUNZIONI E TRIGGER TABELLA TANK

Il trigger e la funzione di sotto implementati si occupano della creazione di due nuovi ID per ogni vasca che viene aggiunta, in modo del tutto automatico. Il Risulta così molto più semplice eseguire la query INSERT per l'aggiunta di una nuova vasca per tartarughe.

Generazione tank_ID:

```
CREATE FUNCTION generateTankId()
RETURNS TRIGGER AS $$
DECLARE
    counter INT;
    cursorTankID CURSOR FOR SELECT tank_ID FROM tank WHERE center_ID =
NEW.center_ID ORDER BY tank_ID;
BEGIN
    -- Recupero dell'identificatore di una vasca associata al centro
specificato dall'utente
    SELECT tank_ID INTO counter FROM tank WHERE center_ID = NEW.center_ID
LIMIT 1;

    -- Se non ci sono vasche associate al centro (centro nuovo), imposta
il nuovo identificatore a 1
    IF(counter IS NULL) THEN
        NEW.tank_ID := 1;
        RETURN NEW;
    END IF;

    -- Sovrascrittura del risultato non NULL dell'interrogazione
precedente
    counter := 0;

    -- Scorrimento delle vasche esistenti associate al centro
    FOR row_record IN cursorTankID LOOP
        counter := counter + 1;
        -- Se l'identificatore corrente non coincide con il contatore,
imposta il nuovo identificatore
        IF(row_record.tank_ID <> counter) THEN
            NEW.tank_ID := counter;
            RETURN NEW;
        END IF;
    END LOOP;

    -- Incrementa il contatore e imposta il nuovo identificatore
    counter := counter + 1;
    NEW.tank_ID := counter;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire la funzione automaticamente prima
dell'inserimento in 'tank'
```

```
CREATE TRIGGER tankIdGeneration
BEFORE INSERT ON Tank
FOR EACH ROW
EXECUTE FUNCTION generateTankId();
```

Generazione surrogate_Tank:

```
CREATE FUNCTION tankUIDGeneration()
RETURNS TRIGGER AS $$
DECLARE
    counter INT;
    cursorUniqueTankID CURSOR FOR SELECT surrogate_tank FROM tank ORDER
BY surrogate_tank;
BEGIN
    SELECT surrogate_tank INTO counter FROM tank LIMIT 1;

    IF(counter IS NULL) THEN
        NEW.surrogate_tank := 1;
        RETURN NEW;
    END IF;

    -- Azzeramento della variabile counter, se non NULL
    counter := 0;

    -- Scorrimento delle vasche esistenti
    FOR rows IN cursorUniqueTankID LOOP
        counter := counter + 1;
        -- Se l'id non coincide con il contatore, imposta come id il
numero del contatore
        IF(rows.surrogate_tank <> counter) THEN
            NEW.surrogate_tank := counter;
            RETURN NEW;
        END IF;
    END LOOP;

    NEW.surrogate_tank := (counter + 1);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire la funzione automaticamente prima
dell'inserimento in 'tank'
CREATE TRIGGER tankUniqueIdGeneration
BEFORE INSERT ON Tank
FOR EACH ROW
EXECUTE FUNCTION tankUIDGeneration();
```

4.5 DEFINIZIONE TABELLA TURTLE

La tabella rappresenta tutte le tartarughe con targhetta. In generale, una tartaruga che abbia avuto almeno una cartella medica in un centro sarà in questa tabella con il numero identificativo ad essa assegnato in automatico. Una tartaruga che possiede sia un tank_UID che un center_ID è a data attuale presente in una vasca all'interno di un centro. Al contrario, una tartaruga che riporta entrambi gli attributi come NULL è stata liberata in mare. Si è preferito non creare un vincolo del tipo "Ogni tartaruga deve avere avuto almeno una cartella medica per essere registrata nel sistema" per poter lasciare la libertà ai ricercatori di assegnare targhette metalliche anche a tartarughe che non necessitano di assistenza medica, magari direttamente in spiaggia. Si è seguito lo stesso ragionamento anche per le misurazioni.

```
-- Creazione del dominio 'sex' per rappresentare il sesso delle
tartarughe (Maschio o Femmina)
CREATE DOMAIN sex AS CHAR(1)
    CHECK(VALUE='M' OR VALUE='F');

CREATE TABLE turtle (
    -- Identificativo unico della vasca associata alla tartaruga
    tank_UID INT,

    -- Identificativo del centro associato alla tartaruga
    center_ID CHAR(10),

    -- Identificativo unico della tartaruga (15 caratteri, generato dal
trigger new_TurtleID)
    turtle_ID CHAR(15) PRIMARY KEY,

    -- Sesso della tartaruga (Vincolato al dominio 'sex', obbligatorio)
    turtle_sex sex NOT NULL,

    -- Nome della tartaruga (massimo 10 caratteri, obbligatorio)
    name VARCHAR(20) NOT NULL,

    -- Specie della tartaruga (massimo 30 caratteri, obbligatoria)
    species VARCHAR(30) NOT NULL,

    -- Vincoli di chiave esterna per garantire l'integrità referenziale
con le tabelle 'tank' e 'center'
    FOREIGN KEY (tank_UID) REFERENCES tank(surrogate_tank),
    FOREIGN KEY (center_ID) REFERENCES center(ID_Center)
);
```

4.5.1 FUNZIONI E TRIGGER TABELLA TURTLE

4.5.1.1 TANK_CONSISTENCY

Il trigger e la funzione di sotto implementati si occupano del controllo della capacità di una vasca prima dell'inserimento di una nuova tartaruga. Se la capacità è stata superata, fornirà un errore, invitando a cambiare vasca in cui si sta per inserire la tartaruga in questione

```
CREATE FUNCTION tank_Check()
RETURNS TRIGGER AS $$
DECLARE
    -- Dichiarazione di una variabile per memorizzare il conteggio delle
tartarughe nella vasca
    turtleCount INT;
BEGIN
    -- Count del numero di tartarughe nella vasca corrente
    turtleCount := (SELECT COUNT(*) FROM turtle WHERE tank_UID =
NEW.tank_UID);

    -- Verifica se il numero di tartarughe supera la capacità massima
della vasca
    IF turtleCount > (SELECT capacity FROM tank WHERE surrogate_tank =
NEW.tank_UID) THEN
        -- Se la capacità è superata, genera un'eccezione
        RAISE EXCEPTION 'Vasca già piena, selezionane una diversa';
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire automaticamente la funzione prima
dell'inserimento di una nuova tartaruga
CREATE TRIGGER tank_Consistency
BEFORE INSERT ON turtle
FOR EACH ROW
EXECUTE FUNCTION tank_Check();
```

4.5.1.2 GENERAZIONE ID NUOVA TARTARUGA

Il trigger e la funzione di sotto implementati si occupano della creazione di un nuovo ID per ogni tartaruga che viene aggiunta, in modo del tutto automatico. Risulta così molto più semplice eseguire la query INSERT per l'aggiunta di una nuova tartaruga. Il numero generato sarà anche quello da stampare su targhetta metallica che verrà poi usata per identificare univocamente la tartaruga.

```
CREATE FUNCTION new_TurtleId()
RETURNS TRIGGER AS $$
DECLARE
    trt_number INT;
    trt_id VARCHAR;
BEGIN
    trt_number := substring((SELECT turtle_ID FROM turtle ORDER BY
turtle_ID DESC LIMIT 1) FROM 4 FOR 12)::INT;
    IF(trt_number IS NULL) THEN
        trt_id := 'TID' || lpad('1', 12, '0');
    ELSE
        trt_id := 'TID' || lpad((trt_number+1)::VARCHAR, 12, '0');
    END IF;
    NEW.turtle_id := trt_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire automaticamente la funzione prima
dell'inserimento di una nuova tartaruga
CREATE TRIGGER new_Turtle
BEFORE INSERT ON turtle
FOR EACH ROW
EXECUTE FUNCTION new_TurtleId();
```

4.5.1.3 SELEZIONE CENTRO AUTOMATICA

```
CREATE FUNCTION automaticCenterId()
RETURNS TRIGGER AS $$
BEGIN
    -- Assegnazione del centro_ID in base alla vasca associata alla
tartaruga
    NEW.center_ID := (SELECT center_ID FROM tank WHERE NEW.tank_UID =
surrogate_tank);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire la funzione automaticamente prima
dell'inserimento in 'turtle'
CREATE TRIGGER autoCenterId
BEFORE INSERT ON Turtle
FOR EACH ROW
EXECUTE FUNCTION automaticCenterId();
```

4.6 DEFINIZIONE TABELLA MEDICAL_RECORD

La seguente tabella rappresenta una cartella medica associata ad una tartaruga. Ogni cartella medica viene aperta all'entrata di una tartaruga nel centro di recupero e viene chiusa automaticamente all'update di "release_date" da NULL a un valore. Una volta chiusa, una cartella medica non deve più essere modificabile e la tartaruga viene considerata fuori dal centro, ergo rilasciata in mare.

```
CREATE TABLE medical_record (
    -- Identificativo interno del record medico (massimo 20 caratteri,
chiave primaria)
    Internal_ID CHAR(20) PRIMARY KEY,

    -- Data di accesso al record medico (obbligatoria)
    access_date DATE NOT NULL,

    -- Dati di localizzazione (tipo 'point', obbligatorio)
    location_data POINT NOT NULL,

    -- Data di rilascio del record medico (opzionale)
    release_date DATE,

    -- Identificativo del centro (massimo 10 caratteri, obbligatorio)
    Center_ID CHAR(10) NOT NULL,

    -- Identificativo della tartaruga associata al record medico (massimo
15 caratteri, obbligatorio)
    Turtle_ID CHAR(15) NOT NULL,

    -- Vincoli di chiave esterna per garantire l'integrità referenziale
con le tabelle 'center' e 'turtle'
    FOREIGN KEY(Center_ID) REFERENCES center(ID_Center),
    FOREIGN KEY(Turtle_ID) REFERENCES turtle(Turtle_ID) ON DELETE CASCADE
);
```

4.6.1 FUNZIONI E TRIGGER TABELLA MEDICAL_RECORD

4.6.1.1 RELEASED_TURTLE

Il seguente trigger impedisce la modifica in toto di una cartella medica chiusa. Una cartella medica è considerata chiusa nel momento in cui all'istanza di "medical_record" viene aggiunta la data "release_date", che risulterà non più NULL, suo valore di default.

```
CREATE FUNCTION noRecordModification()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se la data di rilascio del record medico è già impostata
    IF OLD.release_date IS NOT NULL THEN
       -- Se è già chiuso, genera un'eccezione che impedisce la modifica
       RAISE EXCEPTION 'Non puoi modificare una cartella medica chiusa.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire la funzione prima
dell'aggiornamento di un record medico
CREATE TRIGGER released_Turtle
BEFORE UPDATE ON medical_record
FOR EACH ROW
EXECUTE FUNCTION noRecordModification();
```

4.6.1.2 AUTORELEASE TURTLE

Il seguente trigger, alla modifica di release_date di una cartella medica, e quindi al rilascio di una tartaruga, esegue l'update sulla tabella TURTLE per conservare la coerenza del database.

```
CREATE FUNCTION turtle_Release()
RETURNS TRIGGER AS $$
BEGIN
    -- Aggiorna le informazioni sulla tartaruga impostando 'center_ID' e
'tank_UID' a 'NULL'
    UPDATE turtle
    SET center_ID = 'NULL', tank_UID = 'NULL'
    WHERE turtle_ID = OLD.Turtle_ID;
END
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire automaticamente la funzione prima
dell'aggiornamento della data di rilascio
CREATE TRIGGER autoRelease
AFTER UPDATE OF release_date ON medical_record
FOR EACH ROW
EXECUTE FUNCTION turtle_Release();
```

4.6.1.3 GENERAZIONE ID INTERNO

Il seguente trigger genera un Id interno per la nuova cartella medica.

```
CREATE FUNCTION internalIdGen()
RETURNS TRIGGER AS $$
DECLARE
    -- Dichiarazione di variabili per memorizzare l'ultimo identificatore
e il nuovo identificatore
    lastId VARCHAR(20);
    finalId VARCHAR(20);
    finalNumber INT;
BEGIN
    -- Ottiene l'ultimo identificatore interno associato al centro
corrente
    SELECT Internal_ID INTO lastId FROM medical_record
    WHERE Internal_ID LIKE concat(NEW.Center_ID,'-','%')
    ORDER BY Internal_ID DESC LIMIT 1;

    -- Verifica se non ci sono identificatori precedenti per il centro
corrente
    IF lastId IS NULL
        THEN finalId := NEW.Center_ID || '-' || lpad(1::VARCHAR, 9, '0');
    ELSE
        -- Estrae il numero finale dall'ultimo identificatore e
incrementa di 1
        finalNumber := CAST(substring(lastId FROM 12 for 9) AS INT);
        finalNumber := finalNumber + 1;

        -- Crea il nuovo identificatore concatenando il numero
incrementato
        finalId := NEW.Center_ID || '-' || lpad(finalNumber::VARCHAR, 9,
'0');
    END IF;

    -- Assegna il nuovo identificatore interno al record medico in
inserimento
    NEW.Internal_ID := finalId;

    -- Restituisce la nuova riga
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire automaticamente la funzione prima
dell'inserimento di un nuovo record medico
CREATE TRIGGER newInternalID
BEFORE INSERT ON medical_record
FOR EACH ROW
EXECUTE FUNCTION internalIdGen();
```

4.6.2 DEFINIZIONE TABELLA EXAMINATION

La tabella delle visite alle tartarughe raggruppa lo sviluppo della salute delle tartarughe ferite e con parti del corpo compromesse. All'inserimento delle condizioni di salute della tartaruga all'interno del frontend JavaFX, il backend creerà anche una stima della condizione della tartaruga nel suo intero. In generale, la stima della condizione sarà definita dalla ferita più grave riportata. (Se una tartaruga ha gli occhi compromessi (C) e il resto del corpo normale (N) la stima della condizione sarà uguale a "Compromessa"). La condizione media della tartaruga sarà utile per le interrogazioni che serviranno a definire le statistiche per ogni centro; per questo motivo, viene definito anche il boolean "first_visit".

```
-- Creazione del dominio 'medcondition' per rappresentare le condizioni
mediche
CREATE DOMAIN medcondition AS CHAR(1)
    CHECK(VALUE='C' OR VALUE='D' OR VALUE='L' OR VALUE='N' OR VALUE='P');

CREATE TABLE Examination(
    -- Identificativo interno della cartella medica (massimo 20 caratteri,
obbligatorio)
    Internal_ID CHAR(20) NOT NULL,

    -- Condizioni delle parti della tartaruga (condizione medica,
obbligatopiaria)
    Head_Status medcondition NOT NULL,
    Eyes_Status medcondition NOT NULL,
    Tail_Status medcondition NOT NULL,
    Fins_Status medcondition NOT NULL,
    Neck_Status medcondition NOT NULL,
    Beak_Status medcondition NOT NULL,
    Nose_Status medcondition NOT NULL,

    -- Salute media complessiva (condizione medica, obbligatoria)
    AvgHealth medcondition NOT NULL,

    -- Data dell'esame (obbligatoria, valore predefinito: data corrente)
    Ex_Date DATE NOT NULL DEFAULT CURRENT_DATE,

    -- Note del veterinario (testo libero)
    Vet_Notes TEXT,

    -- Flag per indicare se è la prima visita (obbligatorio, valore
predefinito: FALSE)
    first_visit BOOLEAN NOT NULL DEFAULT FALSE,

    -- Vincolo di chiave esterna per garantire l'integrità referenziale
con la tabella 'medical_record'
    FOREIGN KEY(Internal_ID) REFERENCES medical_record(Internal_ID) ON
DELETE CASCADE
);
```

4.6.3 DEFINIZIONE TRIGGER TABELLA EXAMINATION

4.6.3.1 EXAMS_CONSISTENCY

Il seguente trigger ha la funzione di controllare la coerenza delle date degli esami inseriti. Si è voluta lasciare la libertà ai veterinari di eseguire visite e di inserirle in un successivo momento all'interno del database, ma solo in parte. È possibile eseguire questo tipo di operazione solo finchè la tartaruga risulta ancora presente all'interno del centro. Nel momento in cui la release_date viene aggiornata sulla cartella medica, e di conseguenza la tartaruga risulta liberata in mare, non potranno più essere aggiunti esami, né la cartella risulterà modificabile.

```
CREATE FUNCTION check_Examination()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se la data dell'esame è precedente alla data di accesso
al record medico
    IF (NEW.Ex_Date < (SELECT Access_Date FROM medical_record WHERE
internal_id = NEW.Internal_ID))
    -- Oppure se la data di rilascio in "medicalrecord" è già impostata
       OR ((SELECT Release_Date FROM medical_record WHERE internal_id =
NEW.Internal_ID) IS NOT NULL)
    THEN
    -- Se una delle condizioni non è soddisfatta, genera un'eccezione
       RAISE EXCEPTION 'Errore: Data non consistente con il database';
    END IF;
END;
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire la funzione prima
dell'inserimento di un nuovo esame
CREATE TRIGGER exams_Consistency
BEFORE INSERT ON Examination
FOR EACH ROW
EXECUTE FUNCTION check_Examination();
```

4.6.3.2 EXAMS_UPDATE

Il seguente trigger ha la funzione di impedire la modifica di una visita una volta passata la giornata in cui è stata effettuata. Questo perché ogni visita è considerata completa una volta inserita in database e per costringere alla creazione di una nuova visita nel caso di sviluppi con la tartaruga, in modo tale che eventuali update non vadano ad intaccare le statistiche delle tartarughe singole, da cui poter visualizzare l'andamento di salute generale della tartaruga in questione.

```
CREATE FUNCTION updateNoModify()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se la data corrente è successiva alla data di rilascio
del record medico associato
    IF CURRENT_DATE > (SELECT Release_Date FROM medical_record WHERE
internal_id = OLD.Internal_ID)
       -- Se la condizione è soddisfatta, genera un'eccezione
       THEN RAISE EXCEPTION 'Non puoi modificare una visita una volta
inserita.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire la funzione prima
dell'aggiornamento di un esame medico
CREATE TRIGGER exams_Update
BEFORE UPDATE ON Examination
FOR EACH ROW
EXECUTE FUNCTION updateNoModify();
```

4.6.3.3 MAXONEFIRSTIVIST

Il seguente trigger ha la funzione di controllare l'inserimento e l'update di prime visite. Ne può infatti esistere solo una associata ad ogni cartella medica.

```
CREATE FUNCTION oneFirstVisit()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se esiste già un esame medico con 'first_visit' impostato
a TRUE per la stessa cartella medica
    IF ((NEW.first_visit = TRUE) AND (SELECT Internal_ID FROM Examination
WHERE first_visit=TRUE AND Internal_ID = NEW.Internal_ID LIMIT 1) IS NOT
NULL)
       -- Se la condizione è soddisfatta, genera un'eccezione
       THEN RAISE EXCEPTION 'La prima visita è unica per ogni cartella
medica.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire la funzione prima
dell'inserimento di un nuovo esame medico
CREATE TRIGGER maxOneFirstVisit
BEFORE INSERT OR UPDATE OF first_visit ON Examination
FOR EACH ROW
EXECUTE FUNCTION oneFirstVisit();
```

4.7 DEFINIZIONE TABELLA MEASUREMENT

La seguente tabella definisce una misurazione di una tartaruga. La logica è similare a quella delle visite. Considerato però che le misurazioni di una tartaruga potrebbero avvenire in qualsiasi momento e non necessariamente ad opera di un veterinario, è risultato più logico associarle alla tartaruga stessa e non alla visita. Si è preferito non creare un vincolo del tipo "Una misurazione può essere effettuata solo in date in cui la tartaruga fosse presente in un centro" per poter lasciare la libertà ai ricercatori di assegnare targhette metalliche e misurare anche tartarughe che non necessitano di assistenza medica, magari direttamente in spiaggia.

```
CREATE TABLE Measurement(
    -- Identificativo della tartaruga associata alla misurazione (massimo
15 caratteri, obbligatorio)
    turtle_ID CHAR(15) NOT NULL,

    -- Larghezza della tartaruga (valore a virgola mobile, obbligatorio)
    width float4 NOT NULL,

    -- Lunghezza della tartaruga (valore a virgola mobile, obbligatorio)
    length float4 NOT NULL,

    -- Peso della tartaruga (valore a virgola mobile, obbligatorio)
    weight float4 NOT NULL,

    -- Data della misurazione (obbligatoria, valore predefinito: data
corrente)
    m_Date DATE NOT NULL DEFAULT CURRENT_DATE,

    -- Vincolo di chiave esterna per garantire l'integrità referenziale
con la tabella 'turtle'
    FOREIGN KEY(Turtle_ID) REFERENCES turtle(Turtle_ID) ON DELETE CASCADE
);
```

4.7.1 TRIGGER TABELLA MEASUREMENT

Il seguente trigger impone una logica molto simile a quella già vista per gli esami. Una volta creata una misurazione, risulta impossibile modificarla.

```
CREATE FUNCTION check_Measurements()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se la data della misurazione da aggiornare è diversa
dalla data corrente
    IF OLD.m_Date <> CURRENT_DATE THEN
       -- Se la condizione è soddisfatta, genera un'eccezione
       RAISE EXCEPTION 'Non puoi modificare una misurazione in una data
diversa. Crearne una nuova';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Creazione di un trigger per eseguire la funzione prima
dell'aggiornamento di una misurazione
CREATE TRIGGER appraisal_Update
BEFORE UPDATE ON Measurement
FOR EACH ROW
EXECUTE FUNCTION check_Measurements();
```

5 DEFINIZIONE FUNZIONI A FINE STATISTICO

5.1 STATISTICHE MENSILI

```
CREATE FUNCTION MonthlyStats(
    IN month_to_get INT,
    IN year_to_get INT,
    OUT total_turtles INT,
    OUT compromised_t INT,
    OUT deepwounded_t INT,
    OUT lightwounded_t INT,
    OUT normal_t INT,
    OUT perfect_t INT
) AS $$
BEGIN
    total_turtles := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(MONTH FROM Ex_Date) =
month_to_get
                    AND EXTRACT(YEAR FROM Ex_Date) = year_to_get);

    compromised_t := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(MONTH FROM Ex_Date) =
month_to_get
                    AND EXTRACT(YEAR FROM Ex_Date) = year_to_get AND
AvgHealth = 'C');

    deepwounded_t := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(MONTH FROM Ex_Date) =
month_to_get
                    AND EXTRACT(YEAR FROM Ex_Date) = year_to_get AND
AvgHealth = 'D');

    lightwounded_t := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(MONTH FROM Ex_Date) =
month_to_get
                    AND EXTRACT(YEAR FROM Ex_Date) = year_to_get AND
AvgHealth = 'L');

    normal_t := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(MONTH FROM Ex_Date) =
month_to_get
                    AND EXTRACT(YEAR FROM Ex_Date) = year_to_get AND
AvgHealth = 'N');

    perfect_t := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(MONTH FROM Ex_Date) =
month_to_get
                    AND EXTRACT(YEAR FROM Ex_Date) = year_to_get AND
AvgHealth = 'P');
END;
$$ language plpgsql;
```

5.2 STATISTICHE ANNUALI

```
CREATE FUNCTION YearlyStats(
    IN year_to_get INT,
    OUT total_turtles INT,
    OUT compromised_t INT,
    OUT deepwounded_t INT,
    OUT lightwounded_t INT,
    OUT normal_t INT,
    OUT perfect_t INT
) AS $$
BEGIN
    total_turtles := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(YEAR FROM Ex_Date) =
year_to_get);

    compromised_t := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(YEAR FROM Ex_Date) =
year_to_get
                    AND AvgHealth = 'C');

    deepwounded_t := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(YEAR FROM Ex_Date) =
year_to_get
                    AND AvgHealth = 'D');

    lightwounded_t := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(YEAR FROM Ex_Date) =
year_to_get
                    AND AvgHealth = 'L');

    normal_t := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(YEAR FROM Ex_Date) =
year_to_get
                    AND AvgHealth = 'N');

    perfect_t := (SELECT COUNT(*) FROM examination WHERE
                    first_visit = TRUE AND EXTRACT(YEAR FROM Ex_Date) =
year_to_get
                    AND AvgHealth = 'P');
END;
$$ language plpgsql;
```

6 POPOLAMENTO DATABASE CON DATI DI ESEMPIO

Per il popolamento del database verranno creati 12 centri di recupero, diversi impiegati con password associate, diverse vasche per ogni centro. Al fine di testare meglio la creazione di esami, verranno create 6 tartarughe principali a cui verranno associati cartelle mediche e diversi esami, e saranno create altre tartarughe con sia il CENTER_ID che il TANK_ID come NULL, quindi già rilasciate in mare oppure a cui è stata associata la targhetta direttamente in spiaggia.

```
-- Popolamento tabella center - 12 centri di recupero per tartarughe
marine
INSERT INTO center(name, contact_email, contact_number, province, address,
CAP, city) VALUES
('SeaTurtleRescue Center', 'info@seaturtlerescue.com', '1234567890', 'GE',
'Via Mare 1', '16033', 'Genova'),
('OceanCare Center', 'contact@oceancarecenter.org', '2345678901', 'NA',
'Via Oceano 2', '80100', 'Napoli'),
('TurtleHaven Center', 'info@turtlehavencenter.net', '3456789012', 'CA',
'Viale Costiero 3', '09123', 'Cagliari'),
('AquaRehab Center', 'aquarehab@email.com', '4567890123', 'PA',
'Lungomare 4', '90100', 'Palermo'),
('MarineRescue Center', 'rescue@marinerescue.org', '5678901234', 'RC',
'Corso Marittimo 5', '88100', 'Reggio Calabria'),
('BlueWave Center', 'bluewave.contact@gmail.com', '6789012345', 'VE',
'Viale Costiero 6', '30100', 'Venezia'),
('TurtleSafe Center', 'info@turtlesafe.com', '7890123456', 'FI', 'Via
Conchiglia 7', '50100', 'Firenze'),
('CoastalCare Center', 'coastalcare@center.net', '8901234567', 'SP',
'Lungomare 8', '16100', 'La Spezia'),
('SeaLife Haven', 'sealifehaven@gmail.com', '9012345678', 'SA', 'Viale
Oceanico 9', '34100', 'Salerno'),
('TurtleRescue Center', 'rescue@turtlerescue.org', '0123456789', 'CT',
'Via Marina 10', '95100', 'Catania'),
('PacificRehab Center', 'pacificrehab@center.com', '2345678901', 'GE',
'Corso Pacifico 11', '16100', 'Portofino'),
('SunsetShores Center', 'sunsetshores@email.com', '3456789012', 'CA',
'Viale Tramonto 12', '07100', 'Cagliari');

COMMIT;

-- Popolamento tabella employee - 30 impiegati distribuiti tra operatori,
-- tecnici di laboratorio, veterinari e ricercatori.
INSERT INTO employee(name, surname, date_of_birth, ProfileType) VALUES
('Mario', 'Rossi', '1985-03-15', 'V'),
('Anna', 'Bianchi', '1990-06-22', 'R'),
('Luca', 'Verdi', '1988-11-10', 'O'),
('Giulia', 'Neri', '1995-04-30', 'L'),
('Marco', 'Gialli', '1987-08-05', 'V'),
('Elena', 'Marroni', '1992-01-18', 'R'),
('Alessio', 'Blu', '1989-07-12', 'O'),
('Francesca', 'Rosa', '1997-09-25', 'L'),
('Paolo', 'Arancioni', '1986-12-03', 'V'),
```

```
('Sara', 'Turchesi', '1993-05-28', 'R'),
('Davide', 'Grigi', '1984-02-08', 'O'),
('Martina', 'Ciano', '1998-08-14', 'L'),
('Simone', 'Indaco', '1985-10-20', 'V'),
('Laura', 'Celesti', '1991-04-16', 'R'),
('Giovanni', 'Ruggine', '1988-07-07', 'O'),
('Valentina', 'Rosa', '1996-11-23', 'L'),
('Roberto', 'Turchese', '1987-06-01', 'V'),
('Elisa', 'Gialloro', '1994-03-12', 'R'),
('Andrea', 'Arancione', '1989-09-08', 'O'),
('Camilla', 'Azzurro', '1990-12-28', 'L'),
('Nicola', 'Verde', '1986-02-17', 'V'),
('Cristina', 'Grigio', '1992-07-19', 'R'),
('Stefano', 'Magenta', '1984-05-09', 'O'),
('Alessandra', 'Bluette', '1993-10-31', 'L'),
('Federico', 'Giallino', '1988-04-26', 'V'),
('Giorgia', 'Turchina', '1995-01-14', 'R'),
('Enrico', 'Aranciato', '1987-08-03', 'O'),
('Beatrice', 'Rosa', '1991-03-22', 'L'),
('Matteo', 'Azzurro', '1986-11-07', 'V'),
('Chiara', 'Viola', '1994-06-18', 'R');

COMMIT;

-- Popolamento tabella employment - Costruzione delle relazioni di
impiego
INSERT INTO employment(center_id, employee_id) VALUES
('CTR0000001', 'VET0000001'),
('CTR0000002', 'RES0000001'),
('CTR0000003', 'OPR0000001'),
('CTR0000004', 'LBT0000001'),
('CTR0000005', 'VET0000002'),
('CTR0000006', 'RES0000002'),
('CTR0000007', 'OPR0000002'),
('CTR0000008', 'LBT0000002'),
('CTR0000009', 'VET0000003'),
('CTR0000010', 'RES0000003'),
('CTR0000011', 'OPR0000003'),
('CTR0000012', 'LBT0000003'),
('CTR0000001', 'VET0000004'),
('CTR0000002', 'RES0000004'),
('CTR0000003', 'OPR0000004'),
('CTR0000004', 'LBT0000004'),
('CTR0000005', 'VET0000005'),
('CTR0000006', 'RES0000005'),
('CTR0000007', 'OPR0000005'),
('CTR0000008', 'LBT0000005'),
('CTR0000009', 'VET0000006'),
('CTR0000010', 'RES0000006'),
('CTR0000011', 'OPR0000006'),
```

```
('CTR0000012', 'LBT0000006'),
('CTR0000001', 'VET0000007'),
('CTR0000002', 'RES0000007'),
('CTR0000003', 'OPR0000007'),
('CTR0000004', 'LBT0000007'),
('CTR0000005', 'VET0000008'),
('CTR0000006', 'RES0000008'),
('CTR0000012', 'VET0000001'),
('CTR0000005', 'RES0000002'),
('CTR0000009', 'OPR0000003'),
('CTR0000004', 'LBT0000004'),
('CTR0000008', 'VET0000005'),
('CTR0000011', 'RES0000006'),
('CTR0000006', 'OPR0000007'),
('CTR0000002', 'LBT0000001'),
('CTR0000007', 'VET0000008'),
('CTR0000001', 'RES0000004'),
('CTR0000010', 'OPR0000002'),
('CTR0000003', 'LBT0000006'),
('CTR0000004', 'VET0000003'),
('CTR0000008', 'RES0000007'),
('CTR0000005', 'OPR0000001'),
('CTR0000009', 'LBT0000002'),
('CTR0000006', 'VET0000002'),
('CTR0000012', 'RES0000005');

COMMIT;

-- Popolazione tabella login con password di esempio
INSERT INTO login(employee_ID, hash_password) VALUES
('VET0000001', crypt('vet_password1', gen_salt('bf'))),
('RES0000001', crypt('res_password1', gen_salt('bf'))),
('OPR0000001', crypt('opr_password1', gen_salt('bf'))),
('LBT0000001', crypt('lbt_password1', gen_salt('bf'))),
('VET0000002', crypt('vet_password2', gen_salt('bf'))),
('RES0000002', crypt('res_password2', gen_salt('bf'))),
('OPR0000002', crypt('opr_password2', gen_salt('bf'))),
('LBT0000002', crypt('lbt_password2', gen_salt('bf'))),
('VET0000003', crypt('vet_password3', gen_salt('bf'))),
('RES0000003', crypt('res_password3', gen_salt('bf'))),
('OPR0000003', crypt('opr_password3', gen_salt('bf'))),
('LBT0000003', crypt('lbt_password3', gen_salt('bf'))),
('VET0000004', crypt('vet_password4', gen_salt('bf'))),
('RES0000004', crypt('res_password4', gen_salt('bf'))),
('OPR0000004', crypt('opr_password4', gen_salt('bf'))),
('LBT0000004', crypt('lbt_password4', gen_salt('bf'))),
('VET0000005', crypt('vet_password5', gen_salt('bf'))),
('RES0000005', crypt('res_password5', gen_salt('bf'))),
('OPR0000005', crypt('opr_password5', gen_salt('bf'))),
('LBT0000005', crypt('lbt_password5', gen_salt('bf'))),
```

```
('VET0000006', crypt('vet_password6', gen_salt('bf'))),
('RES0000006', crypt('res_password6', gen_salt('bf'))),
('OPR0000006', crypt('opr_password6', gen_salt('bf'))),
('LBT0000006', crypt('lbt_password6', gen_salt('bf'))),
('VET0000007', crypt('vet_password7', gen_salt('bf'))),
('RES0000007', crypt('res_password7', gen_salt('bf'))),
('OPR0000007', crypt('opr_password7', gen_salt('bf'))),
('LBT0000007', crypt('lbt_password7', gen_salt('bf'))),
('VET0000008', crypt('vet_password8', gen_salt('bf'))),
('RES0000008', crypt('res_password8', gen_salt('bf')));

COMMIT;

-- Popolamento tabella Tank
INSERT INTO tank(center_ID, capacity) VALUES
('CTR0000001', 5),
('CTR0000001', 8),
('CTR0000001', 12),
('CTR0000002', 3),
('CTR0000002', 10),
('CTR0000003', 7),
('CTR0000004', 15),
('CTR0000004', 4),
('CTR0000004', 9),
('CTR0000005', 6),
('CTR0000005', 11),
('CTR0000005', 2),
('CTR0000006', 8),
('CTR0000007', 14),
('CTR0000007', 1),
('CTR0000008', 10),
('CTR0000009', 3),
('CTR0000009', 5),
('CTR0000010', 12),
('CTR0000011', 9),
('CTR0000012', 7),
('CTR0000012', 13),
('CTR0000012', 6);

COMMIT;

-- Tartaruga 1
INSERT INTO turtle(tank_UID, turtle_sex, name, species) VALUES
(1, 'M', 'Terry', 'Caretta comune');

-- Tartaruga 2
INSERT INTO turtle(tank_UID, turtle_sex, name, species) VALUES
(2, 'F', 'Shelly', 'Testudo graeca');

-- Tartaruga 3
```

```
INSERT INTO turtle(tank_UID, turtle_sex, name, species) VALUES
(3, 'M', 'Leonardo', 'Chrysemys picta');

-- Tartaruga 4
INSERT INTO turtle(tank_UID, turtle_sex, name, species) VALUES
(4, 'F', 'Aqua', 'Trachemys scripta');

-- Tartaruga 5
INSERT INTO turtle(tank_UID, turtle_sex, name, species) VALUES
(1, 'M', 'Michelangelo', 'Emys orbicularis');

-- Tartaruga 6
INSERT INTO turtle(tank_UID, turtle_sex, name, species) VALUES
(2, 'F', 'Tina', 'Kinosternon scorpioides');

COMMIT;

-- Tartaruga 1
INSERT INTO medical_record(access_date, location_data, Center_ID,
Turtle_ID) VALUES
('2022-01-15', POINT(41.9028, 12.4964), 'CTR0000001', 'TID000000000001'),
('2022-03-22', POINT(41.9028, 12.4964), 'CTR0000002', 'TID000000000001');

-- Tartaruga 2
INSERT INTO medical_record(access_date, location_data, Center_ID,
Turtle_ID) VALUES
('2022-02-10', POINT(41.9028, 12.4964), 'CTR0000003', 'TID000000000002'),
('2022-04-18', POINT(41.9028, 12.4964), 'CTR0000004', 'TID000000000002');

-- Tartaruga 3
INSERT INTO medical_record(access_date, location_data, Center_ID,
Turtle_ID) VALUES
('2022-01-25', POINT(41.9028, 12.4964), 'CTR0000005', 'TID000000000003'),
('2022-03-30', POINT(41.9028, 12.4964), 'CTR0000006', 'TID000000000003');

-- Tartaruga 4
INSERT INTO medical_record(access_date, location_data, Center_ID,
Turtle_ID) VALUES
('2022-02-18', POINT(41.9028, 12.4964), 'CTR0000007', 'TID000000000004'),
('2022-04-10', POINT(41.9028, 12.4964), 'CTR0000008', 'TID000000000004');

-- Tartaruga 5
INSERT INTO medical_record(access_date, location_data, Center_ID,
Turtle_ID) VALUES
('2022-01-20', POINT(41.9028, 12.4964), 'CTR0000009', 'TID000000000005'),
('2022-03-15', POINT(41.9028, 12.4964), 'CTR0000010', 'TID000000000005');

-- Tartaruga 6
INSERT INTO medical_record(access_date, location_data, Center_ID,
Turtle_ID) VALUES
```

```
('2022-02-05', POINT(41.9028, 12.4964), 'CTR0000011', 'TID0000000000006'),
('2022-04-05', POINT(41.9028, 12.4964), 'CTR0000012', 'TID0000000000006');



-- Esame 1
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000001-000000001', 'C', 'D', 'L', 'N', 'P', 'C', 'N', 'C', '2022-
01-15', 'Esame regolare', TRUE);

-- Esame 2
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000002-000000001', 'N', 'C', 'D', 'L', 'P', 'D', 'N', 'D', '2022-
03-22', 'Note sulle pinne danneggiate', TRUE);

-- Esame 3
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000003-000000001', 'L', 'L', 'P', 'L', 'P', 'N', 'L', 'L', '2022-
03-20', 'Leggeri problemi alla testa', TRUE);

-- Esame 4
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000004-000000001', 'D', 'C', 'N', 'C', 'L', 'C', 'D', 'C', '2022-
04-19', 'Problemi con gli occhi', TRUE);

-- Esame 5
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000005-000000001', 'P', 'P', 'N', 'N', 'N', 'N', 'P', 'N', '2022-
01-31', 'Nessun problema evidente', TRUE);

-- Esame 6
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000006-000000001', 'N', 'L', 'D', 'P', 'D', 'D', 'L', 'D', '2022-
03-30', 'Problemi al collo', TRUE);

-- Esame 7
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
```

```
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000007-000000001', 'N', 'N', 'N', 'N', 'P', 'N', 'N', 'N', '2022-
02-18', 'Esame regolare', TRUE);

-- Esame 8
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000008-000000001', 'N', 'C', 'L', 'D', 'P', 'N', 'C', 'C', '2022-
04-11', 'Pinne danneggiate', TRUE);

-- Esame 9 -- Due visite nello stesso giorno
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000009-000000001', 'N', 'L', 'N', 'N', 'P', 'P', 'P', 'L', '2022-
01-20', 'Leggeri problemi alla testa', TRUE);
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000009-000000001', 'N', 'L', 'N', 'N', 'P', 'P', 'P', 'L', '2022-
01-20', 'Ancora leggeri problemi alla testa', FALSE);

-- Esame 10
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000010-000000001', 'D', 'P', 'N', 'D', 'L', 'D', 'D', 'D', '2022-
03-15', 'Problemi con le pinne', TRUE);

-- Esame 11
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000011-000000001', 'P', 'C', 'N', 'L', 'D', 'L', 'P', 'C', '2022-
02-05', 'Problemi agli occhi', TRUE);

-- Esame 12
INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES
('CTR0000012-000000001', 'N', 'L', 'L', 'P', 'C', 'D', 'L', 'C', '2022-
04-06', 'Problemi al collo', TRUE);



INSERT INTO Measurement(turtle_ID, width, length, weight, m_Date) VALUES
('TID000000000001', 20.1, 25.5, 5.3, '2022-01-15'),
('TID000000000001', 21.2, 26.3, 5.8, '2022-04-25'),
('TID000000000001', 21.5, 27.0, 6.1, '2022-08-10'),
```

```
('TID000000000002', 18.8, 22.3, 4.7, '2022-02-20'),
('TID000000000002', 19.0, 22.7, 4.9, '2022-06-05'),
('TID000000000002', 19.5, 23.2, 5.2, '2022-09-15'),
('TID000000000003', 22.3, 28.5, 6.7, '2022-03-10'),
('TID000000000003', 22.8, 29.0, 7.0, '2022-07-20'),
('TID000000000003', 23.1, 29.5, 7.3, '2022-11-30'),
('TID000000000004', 16.5, 20.0, 4.0, '2022-01-25'),
('TID000000000004', 17.0, 20.5, 4.2, '2022-06-05'),
('TID000000000004', 17.5, 21.0, 4.5, '2022-09-15'),
('TID000000000005', 19.8, 23.5, 5.8, '2022-02-10'),
('TID000000000005', 20.0, 24.0, 6.0, '2022-07-20'),
('TID000000000005', 20.5, 24.5, 6.3, '2022-11-30'),
('TID000000000006', 14.2, 18.0, 3.5, '2022-02-25'),
('TID000000000006', 14.5, 18.5, 3.7, '2022-07-05'),
('TID000000000006', 15.0, 19.0, 4.0, '2022-10-15');

-- Creazione di ulteriori tartarughe senza tank_UID
INSERT INTO turtle(turtle_sex, name, species) VALUES
('M', 'Leonardo', 'Chelonia mydas'),
('F', 'Michelangelo', 'Eretmochelys imbricata'),
('M', 'Donatello', 'Dermochelys coriacea'),
('F', 'Raphael', 'Caretta caretta'),
('M', 'Splinter', 'Natator depressus'),
('F', 'April', 'Chelonia mydas'),
('M', 'Shredder', 'Eretmochelys imbricata'),
('F', 'Mona Lisa', 'Caretta caretta'),
('M', 'Bebop', 'Dermochelys coriacea'),
('F', 'Rocksteady', 'Natator depressus'),
('M', 'Casey', 'Chelonia mydas'),
('F', 'Karai', 'Eretmochelys imbricata'),
('M', 'Baxter', 'Dermochelys coriacea'),
('F', 'Irma', 'Natator depressus'),
('M', 'Krang', 'Caretta caretta');
```

7 TEMPLATE PER INSERT IN OGNI TABELLA

```
-- Template dell'INSERT nella tabella Center:
-- INSERT INTO center(name, contact_email, contact_number, province,
address, CAP, city) VALUES ();

-- Template dell'INSERT nella tabella Employee:
-- INSERT INTO employee(name, surname, date_of_birth, ProfileType) VALUES
();

-- Template dell'INSERT nella tabella Employment:
-- INSERT INTO employment(center_ID, employee_ID) VALUES();

/* Template INSERT e SELECT per la tabella login
INSERT INTO login(employee_ID, hash_password)
VALUES ('VET0000001', crypt('placeholder', gen_salt('bf'))) ;
```

```
Validazione di una password inserita:

SELECT * FROM login
WHERE employee_id='VET0000001'
AND password=crypt('password inserita', hash_password);
*/

-- Template dell'INSERT nella tabella Tank:
-- INSERT INTO TANK(center_ID, capacity) VALUES();

-- Template di un INSERT nella tabella Turtle:
--INSERT INTO turtle(tank_UID, turtle_sex, name, species) VALUES()

-- Template di un INSERT nella tabella Medical_record
-- INSERT INTO medical_record(access_date, location_data, Center_ID,
Turtle_ID) VALUES(..., POINT(...), ..., ...)

-- Template di un INSERT nella tabella Examination
-- INSERT INTO Examination (Internal_ID, Head_Status, Eyes_Status,
Tail_Status, Fins_Status, Neck_Status, Beak_Status, Nose_Status,
AvgHealth, Ex_Date, Vet_Notes, first_visit) VALUES ();

-- Template INSERT Measurement
-- INSERT INTO Measurement(turtle_ID, width, length, weight, m_Date)
VALUES()
```