Relazione Basi di Dati  
2019-2020

Andrea Mansi - 137857

Daniele Passabì - 137758

**Indice**

[Introduzione 3](#_Toc34842033)

[Requisiti 3](#_Toc34842034)

[Fasi del progetto 3](#_Toc34842035)

[1. Raccolta e analisi dei requisiti 4](#_Toc34842036)

[Glossario dei termini 4](#_Toc34842037)

[Analisi dei requisiti 5](#_Toc34842038)

[2. Progettazione concettuale 7](#_Toc34842039)

[3. Progettazione logica 8](#_Toc34842040)

[Ristrutturazione dello schema ER 10](#_Toc34842041)

[Analisi delle prestazioni 8](#_Toc34842042)

[Tabella dei volumi e delle operazioni 8](#_Toc34842043)

[Frequenza operazioni 9](#_Toc34842044)

[Schema relazionale 12](#_Toc34842045)

[4. Progettazione fisica e implementazione 13](#_Toc34842046)

[5. Popolazione e analisi dei dati in R 16](#_Toc34842047)

# Introduzione

<<introduzione al progetto da aggiungere>>

## Requisiti

Si vuole automatizzare il sistema di gestione degli animali di uno zoo.

* Ogni esemplare di animale ospitato è identificato dal suo genere (ad esempio, zebra) e da un codice unico all'interno del genere di appartenenza. Per ogni esemplare, si memorizzano la data di arrivo nello zoo, il nome proprio, il sesso, il paese di provenienza e la data di nascita.
* Lo zoo è diviso in aree. In ogni area c’è un insieme di case, ognuna destinata ad un determinato genere di animali. Ogni casa contiene un insieme di gabbie, ognuna contenente un solo esemplare. Ogni casa ha un addetto che pulisce ciascuna gabbia in un determinato giorno della settimana.
* Gli animali sono sottoposti periodicamente a controllo veterinario. In un controllo, un veterinario rileva il peso degli esemplari, diagnostica eventuali malattie e prescrive il tipo di dieta da seguire.

## Fasi del progetto

Segue la lista delle principali fasi del progetto:

* Raccolta e analisi dei requisiti;
* Progettazione concettuale;
* Progettazione logica;
* Progettazione fisica e implementazione;
* Popolazione e analisi dei dati in R.

# 1. Raccolta e analisi dei requisiti

## Glossario dei termini

La raccolta e analisi dei requisiti è stata preceduta dalla stesura di un glossario dei termini. Il suo scopo è quello di fornire, per ogni concetto rilevante: una breve descrizione del concetto, eventuali sinonimi, relazioni con altri concetti del glossario stesso.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Termine** | **Descrizione** | **Sinonimi** | **Collegamenti** | **Tipologia** |
| *zoo* | Dominio del database | --- | --- | dominio del DB |
| *esemplare* | È identificato dal suo genere e da un codice unico all'interno del genere di appartenenza. | animale | veterinario, gabbia, genere | entità |
| *area* | È formata da un insieme di abitazioni in essa collocate. | zona | abitazione | entità |
| *abitazione* | È destinata ad un determinato genere di animali; ogni abitazione contiene un insieme di gabbie; ogni abitazione ha un addetto che pulisce ciascuna gabbia. | casa | gabbia, addetto pulizie, area, genere | entità |
| *collocata* | Mette in relazione un'abitazione con una determinata area. |  | abitazione, area | relazione |
| *in* | Mette in relazione le gabbie con l'abitazione in cui si trovano. |  | gabbia, abitazione | relazione |
| *contenuto* | Mette in relazione un esemplare con la gabbia nel quale è contenuto. |  | esemplare, gabbia | relazione |
| *gabbia* | Contiene un solo animale. Viene pulita regolarmente da un addetto alle pulizie. |  | esemplare, abitazione | entità |
| *addetto pulizie* | Pulisce tutte le gabbie assegnate all'abitazione in base al proprio turno di pulizia. | dipendente | abitazione | entità |
| *pulire* | Mette in relazione un addetto delle pulizie con le abitazioni che deve pulire. |  | abitazione, addetto pulizie | relazione |
| *veterinario* | Controlla/visita periodicamente gli esemplari. |  | esemplare | entità |
| *visitare* | Controllo periodico degli animali, rilevamento del peso, diagnostica di eventuali malattie, prescrizione del tipo di dieta. | visita | veterinario, esemplare | relazione |
| *genere* | Identifica il genere di un esemplare con cui è in relazione. Identifica il genere di animali che possono essere contenuti nelle gabbie di una determinata abitazione. | tipologia, specie | abitazione, esemplare | entità |
| *assegnato* | Mette in relazione un'abitazione con un genere per identificare il tipo di animali che possono essere contenuti nelle sue gabbie. |  | abitazione, genere | relazione |
| *appartiene* | Mette in relazione un esemplare con un genere per identificarne l'appartenenza. |  | esemplare, genere | relazione |

## Analisi dei requisiti

Segue il risultato dell’analisi dei requisiti. I requisiti ottenuti sono stati organizzati in gruppi, ciascuno contenente le funzionalità che il database deve fornire per ogni singola entità chiave.

**Dominio: zoo**

Il sistema deve permettere la gestione di uno zoo, deve quindi consentire di mantenere le informazioni sugli esemplari ospitati, sulle aree, abitazioni e gabbie dello zoo, sugli addetti alle pulizie e sui veterinari.

**Esemplare**

* Ogni esemplare è caratterizzato da:
  + genere e codice unico all'interno del genere di appartenenza
  + data di arrivo nello zoo e data di nascita
  + nome proprio
  + sesso
  + paese di provenienza
* Il sistema deve consentire:
  + l’inserimento (e la rimozione) di un esemplare in qualsiasi momento
  + l’assegnazione univoca (e lo spostamento) di ogni esemplare ad una singola gabbia
  + la gestione periodica dei controlli veterinari effettuati a ciascun esemplare

**Area**

* Le aree sono formate da un insieme di abitazioni
* Le aree sono contraddistinte da un nome (univoco)
* Il sistema deve consentire di:
  + gestire (aggiungere, modificare e rimuovere) le aree dello zoo
  + cambiare (per ciascuna area) le abitazioni che le appartengono
  + tenere traccia del numero di abitazioni assegnate a ciascun’area

**Abitazione**

* Le abitazioni sono contraddistinte da un ID univoco.
* Ogni abitazione:
  + è destinata ad un determinato genere di animale
  + possiede un insieme di gabbie ad essa assegnate (ciascuna contenente un esemplare)
  + ha un addetto alla pulizia delle gabbie
* Il sistema deve permettere di:
  + gestire (aggiungere, modificare e rimuovere) le abitazioni dello zoo
  + tenere conto del numero di gabbie assegnate a ciascuna abitazione
  + cambiare (per ciascuna abitazione) il genere assegnato

**Gabbia**

* Ciascuna gabbia è identificata da un ID univoco.
* Ciascuna gabbia contiene un solo animale (il cui genere deve coincidere con quello dell’abitazione in cui si trova)
* Il sistema deve permettere di:
  + gestire (aggiungere, modificare e rimuovere) le gabbie dello zoo
  + cambiare (per ciascuna gabbia) l’esemplare in essa contenuto

**Genere**

* Il sistema deve permettere di:
  + aggiungere e rimuovere generi gestiti dallo zoo
  + assegnare a ciascuna abitazione il genere di animale che può ospitare
  + assegnare a ciascun esemplare il genere a cui appartiene

**Dipendente**

* Ogni dipendente è caratterizzato da:
  + CF (codice fiscale)
  + nome
  + cognome
  + stipendio
  + uno o più numeri di telefono
* I dipendenti sono suddivisi in due categorie:
  + Addetto pulizie
    - pulisce ogni gabbia presente nell’abitazione a cui il suo turno di pulizia fa riferimento
  + Veterinario
    - visita periodicamente gli esemplari
* Il sistema deve permettere di:
  + gestire (aggiungere, modificare e rimuovere) i dipendenti dello zoo
  + tenere traccia dei controlli veterinari effettuati
  + modificare il turno di pulizia e le abitazioni assegnate agli addetti alle pulizie

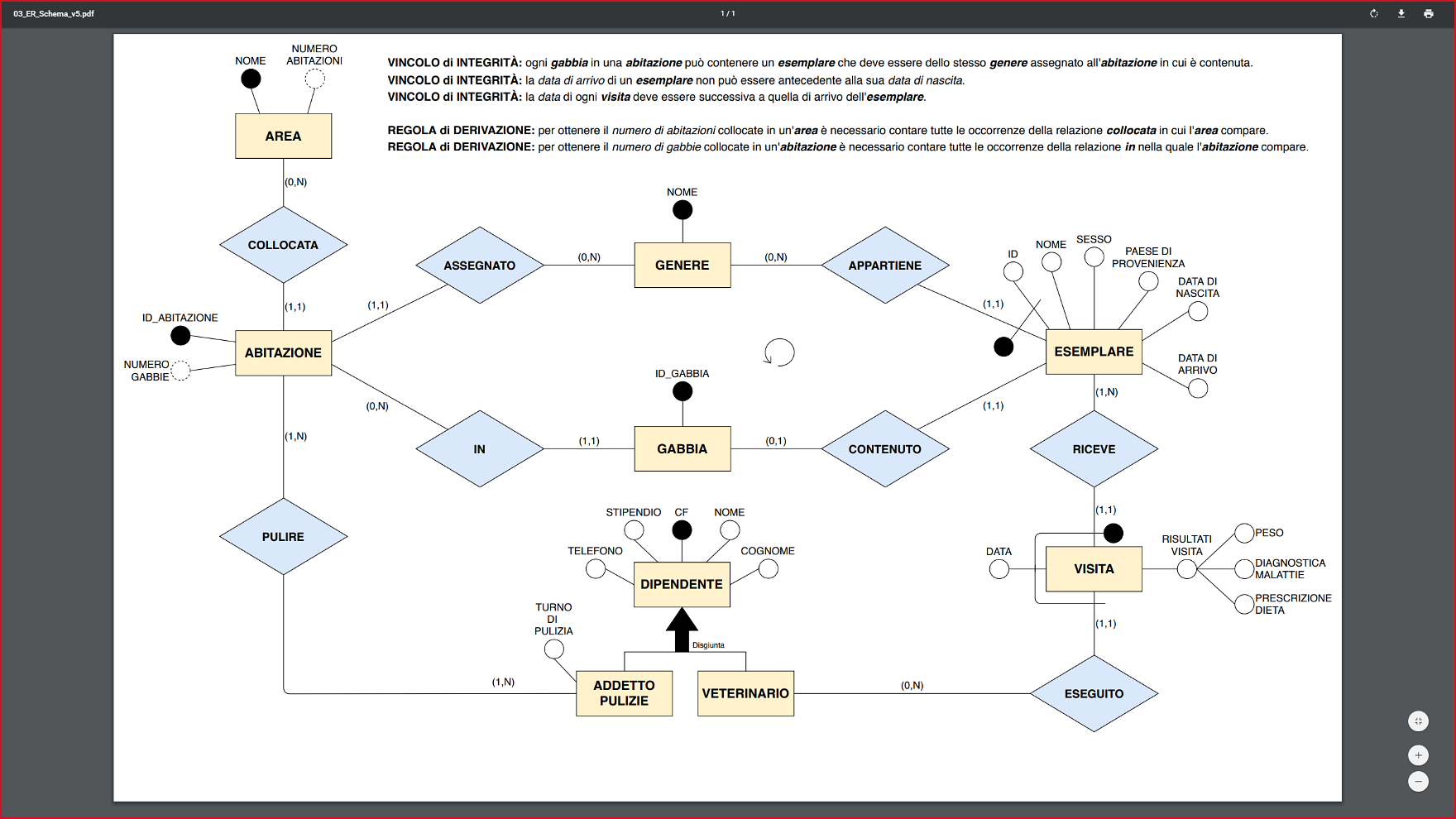
**Visita**

* Ogni visita è caratterizzata da:
  + data, esemplare visitato e veterinario che ha effettuato la visita
  + rilevamento del peso
  + diagnostica di eventuali malattie
  + prescrizione tipologia di dieta
* Il sistema deve permettere di:
  + inserire, modificare e rimuovere nuove visite una volta che vengono effettuate
  + mantenere uno storico delle visite effettuate

**Vincoli aggiuntivi**

* Il sistema deve garantire che in ogni momento sia rispettato il vincolo di integrità sui generi, ovvero: un esemplare di genere x può essere contenuto solo in una gabbia appartenente ad una abitazione di genere assegnato x (ovvero il genere di un esemplare deve essere lo stesso del genere assegnato all’abitazione in cui è contenuta la sua gabbia).
* Vincoli di integrità sulle date: il sistema deve controllare i seguenti due vincoli relativi alle date:
  1. La data di arrivo di un esemplare non può essere antecedente alla sua data di nascita.
  2. Un esemplare non può aver ricevuto una visita prima ancora che sia arrivato nello zoo.

# 2. Progettazione concettuale

Segue lo schema entità relazioni frutto della fase di progettazione concettuale.

<………scrivere qualcosa………..>

# 3. Progettazione logica

Seguono le fasi di progettazione logica.

## Analisi delle prestazioni

Come prima fase si è partiti effettuando un’analisi delle prestazioni basandosi su tabella dei volumi e operazioni. Il risultato dell’analisi è stato poi utilizzato come base di scelta per la ristrutturazione del modello entità relazioni.

### Tabella dei volumi e delle operazioni

Le seguenti tabelle sono frutto di una previsione d’uso del sistema ipotetica, in cui i volumi sono risultato di stime.

**Tabella dei volumi**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Concetto** | **Volume** |  | **Note** | |
| E | Area | 10 |  | n° medio di abitazioni per area | 10 |
| E | Abitazione | 100 |  | n° medio di gabbie per abitazione | 50 |
| E | Gabbia | 5000 |  | percentuale gabbie libere | 10% |
| E | Genere | 80 |  | media esemplari per genere | 56,25 |
| E | Esemplare | 4500 |  | media esemplari per abitazione (escluse quelle vuote) | 47,368 |
| E | Addetto pulizie | 100 |  | media gabbie assegnate a ciascun addetto | 50 |
| R | Collocata | 1000 |  | --- | --- |
| R | In | 5000 |  | si suppone che una gabbia vuota sia comunque assegnata | --- |
| R | Contenuto | 4500 |  | n° gabbie vuote | 500 |
| R | Assegnato | 95 |  | percentuale abitazioni senza genere assegnato (vuote) | 5% |
| R | Appartiene | 4500 |  | --- | --- |
| R | Pulire | 300 |  | n° medio abitazioni assegnate per ciascun addetto alle pulizie | 3 |
| R | Riceve | 270000 |  | --- | --- |
| R | Eseguito | 270000 |  | --- | --- |
| E | Visita | 270000 |  | n° medio visite per esemplare mensili | 0,5 |
| E | Veterinario | 20 |  | n° medio visite effettuate da veterinario dopo 10 anni | 13500 |
|  |  |  |  |  |  |
|  |  |  |  | durata (in anni) prevista utilizzo database | 20 |
|  |  |  |  | NB: si ipotizza che il numero di animali nello zoo rimanga pressoché costante, inoltre, il numero di visite è relativo a un periodo di 10 anni (metà vita del database). | |

**<< pulire 2 righe fango<<**

**Tabella delle operazioni**

Si suppone che in ogni visita vengano lette mediamente le informazioni relative alle due visite precedenti.

Esemplare \* n° medio visite mensili \* 0.25

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Operazione** | **Frequenza** | |
| **I** | Aggiunta nuovo esemplare | 5 | a sett. |
| **I** | Rimozione esemplare | 5 | a sett. |
| **I** | Ricerca collocazione di un esemplare | 4500 | al giorno |
| **I** | Lettura informazioni relative ad un esemplare | 900 | al giorno |
| **I** | Spostamento di un esemplare | 10 | a sett. |
| **I** | Aggiunta di una nuova visita | 563 | a sett. |
| **I** | Lettura informazioni di visite già effettuate | 1125 | a sett. |
| **B** | Lettura stipendio di ciascun dipendente dello zoo | 120 | al mese |
| **I** | Aggiunta, modifica e rimozione di gabbie | 30 | al mese |
| **I** | Aggiunta, modifica e rimozione di abitazioni | 5 | al mese |
| **I** | Aggiunta, modifica e rimozione dipendenti | 5 | al mese |

### Frequenza operazioni

Basandosi sulle tabelle dei volumi e delle operazioni sono state individuate le operazioni più e meno frequenti del database.

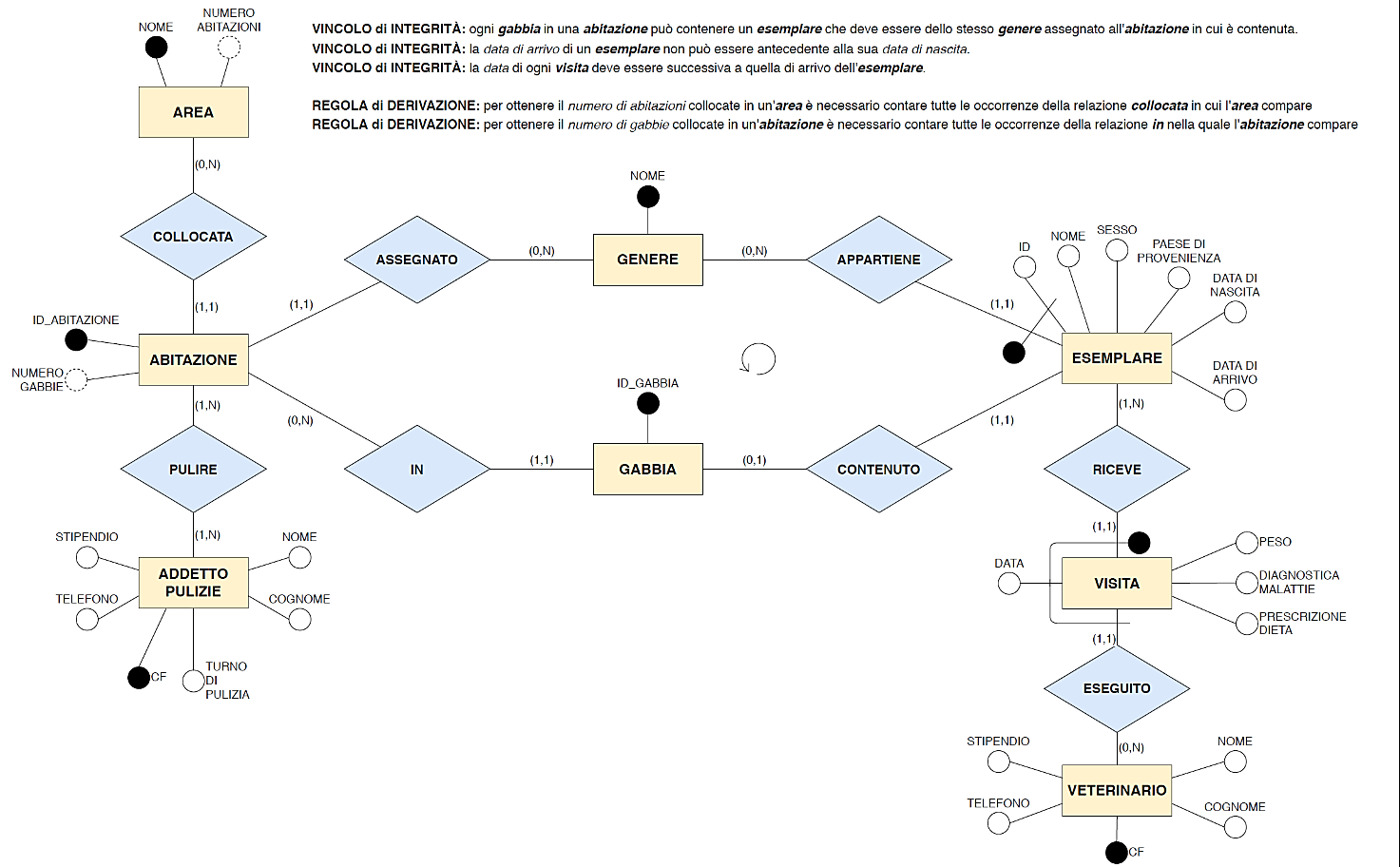
**Operazioni di scrittura:**

* **Operazioni più frequenti (a cadenza giornaliera)**
  + Aggiunta/rimozione di un esemplare ad una gabbia
  + Aggiunta/rimozione esemplari
  + Aggiunta nuove visite veterinarie (scrittura nello “storico”)
* **Operazioni meno frequenti**
  + Modifiche delle assegnazioni degli addetti alle pulizie (si suppone siano modifiche fatte di rado)
  + Aggiunta/rimozione dipendenti
  + Modifica della struttura gerarchica area/abitazione/gabbia (modificati molto raramente)

**Operazioni di lettura:**

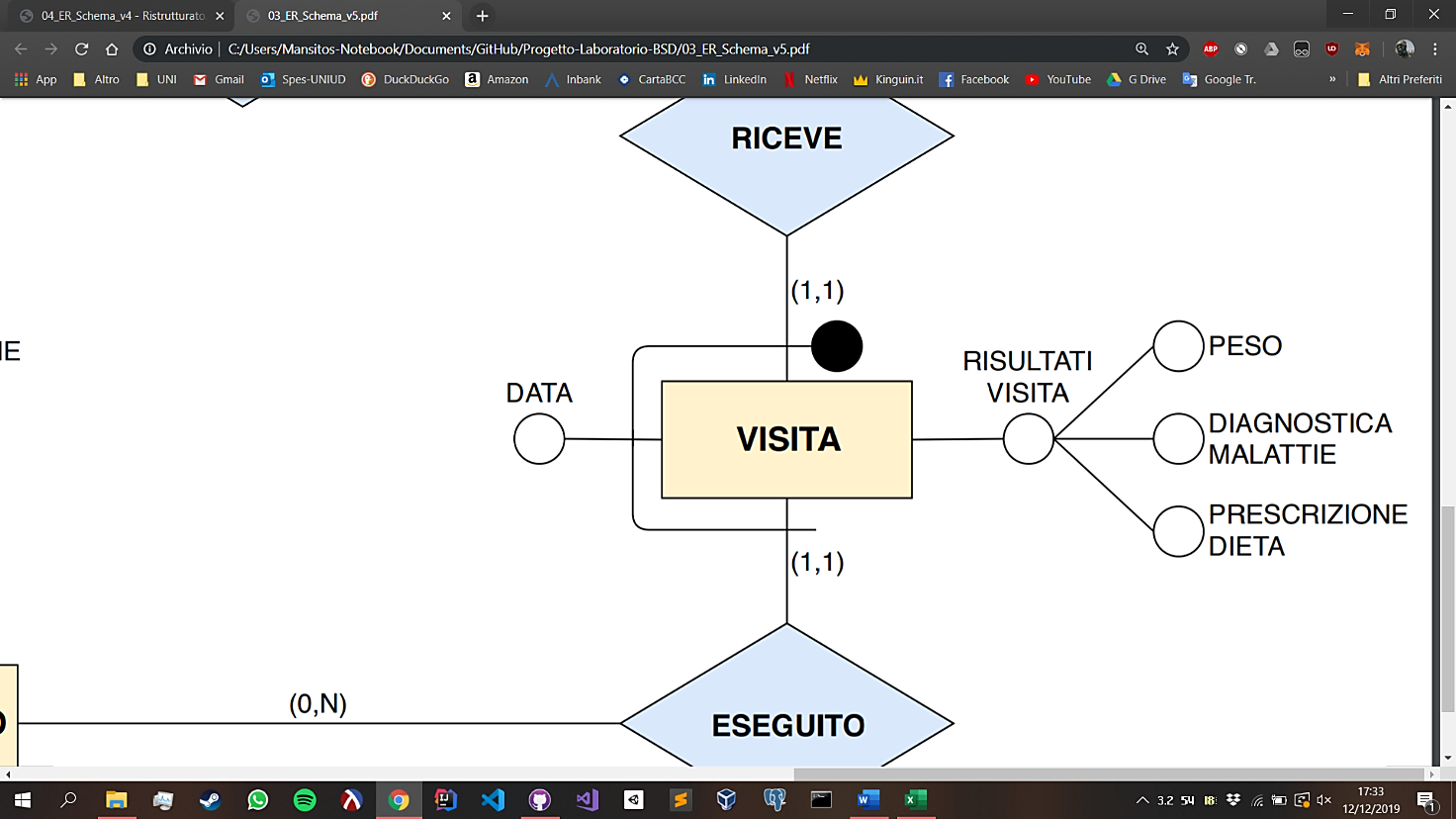
* **Operazioni più frequenti (a cadenza giornaliera)**
  + Informazioni sugli esemplari (molto frequenti)
    - Attributi
    - Collocazione: le entità gabbia, abitazione, area verranno coinvolte nelle operazioni di lettura
  + Informazioni sulle visite (si ipotizza che prima di effettuare una visita, un veterinario consulti quelle precedenti)
* **Operazioni meno frequenti**
  + Informazioni sugli attributi dei dipendenti

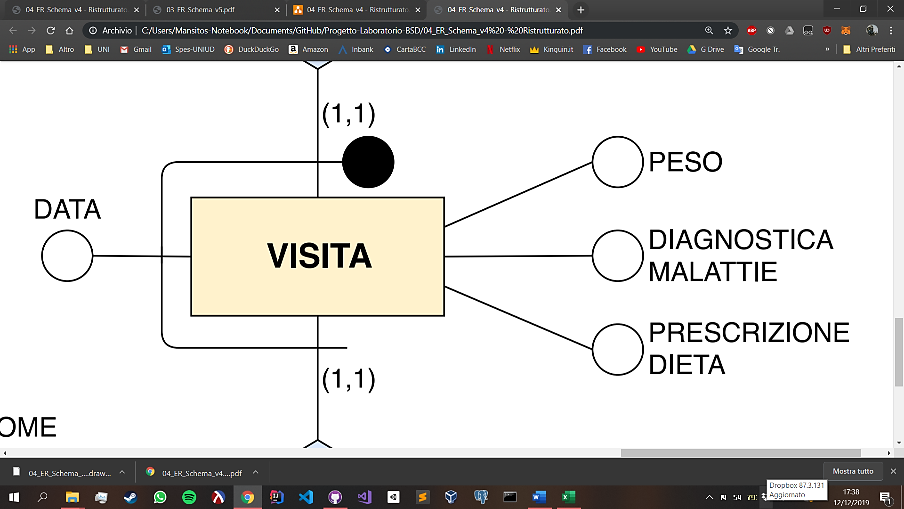
## Ristrutturazione dello schema ER

Segue lo schema ER ristrutturato e le relative modifiche effettuate.

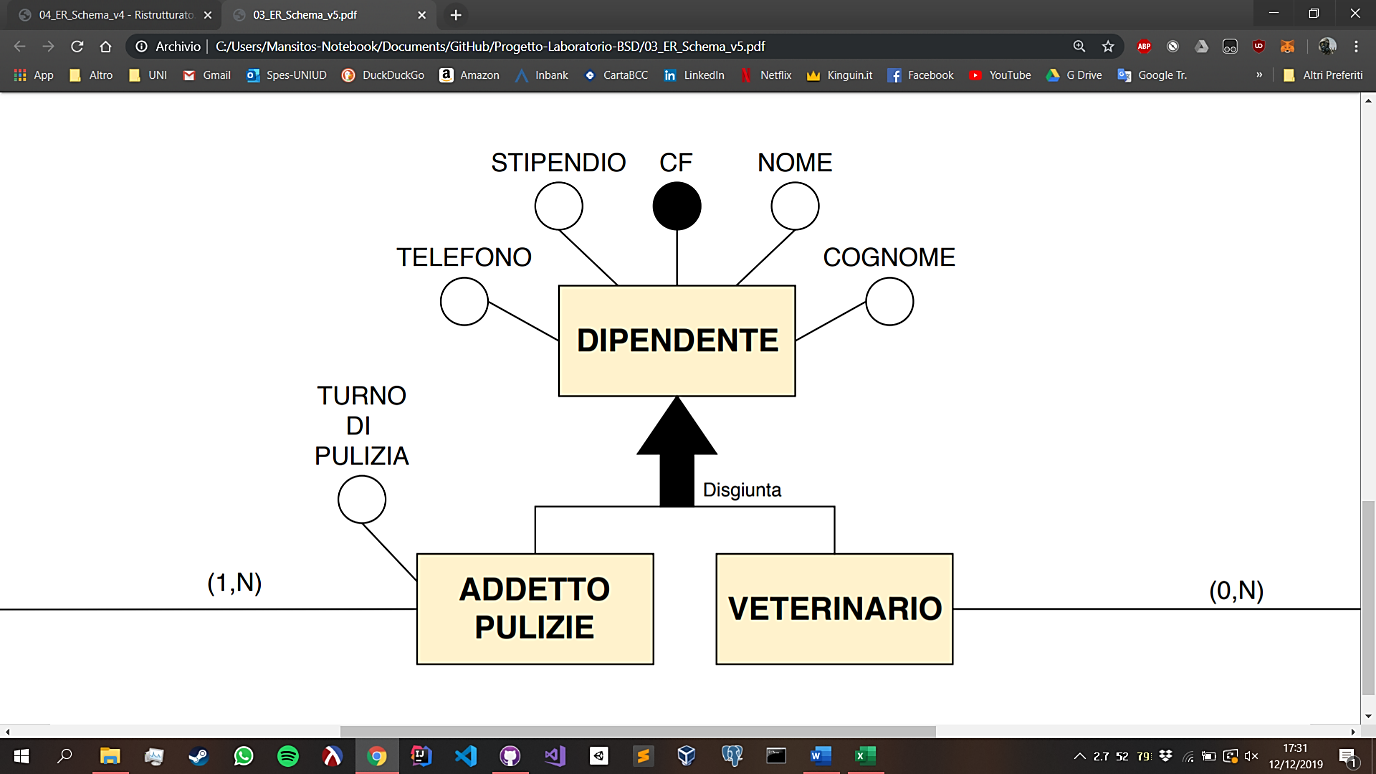
### Rimozione attributi multipli

**Attributo multiplo risultati visita (entità visita)**





### Rimozione generalizzazioni

**Generalizzazione addetto pulizie – veterinario, totale disgiunta**

In fase di analisi delle generalizzazioni è stato valutato di mantenere entrambi i figli e rimuovere l’entità genitore. I motivi che hanno portato a questa scelta sono:

* La generalizzazione è totale
* Non risulta mai necessario rappresentare dipendenti che non siano addetti alle pulizie o veterinari
* Si suppone siano più frequenti le operazioni di lettura riguardanti gli attributi specifici (di addetto alle pulizie e veterinario) che quelli generici
* Mantenere l’entità dipendente avrebbe comportato la presenza di un attributo il cui scopo sarebbe stato quello di indicare se un determinato dipendente fosse un addetto alle pulizie o un veterinario
  + Inoltre, l’attributo turno di pulizia sarebbe stato sempre presente per ogni dipendente, anche per i veterinari (a cui non serve)
* Mantenere l’entità dipendente avrebbe reso più difficoltoso il controllo e la gestione delle inconsistenze (ad es. un addetto alle pulizie che visita un esemplare

Analisi delle ridondanze  
**Attributo genere (ridondanza su ESEMPLARE-ABITAZIONE)**

Possibile inconsistenza: un’abitazione con genere assegnato “x” ha gabbie con esemplari di genere “y”.

Considerazioni:

* L’attributo genere non può essere rimosso dall’entità ESEMPLARE in quanto chiave (chiave multipla genere-id)
* La ricerca di quale genere sia assegnato a una determinata abitazione risulta più veloce nel caso in cui l’attributo genere venga mantenuto (nell’entità ABITAZIONE)
* Non c’è necessità di modificare l’attributo genere nell’entità ESEMPLARE (se non in seguito a un inserimento errato)
* Nel caso in cui si modifichi l’attributo genere nell’entità ABITAZIONE sarà necessario gestire correttamente la riassegnazione degli esemplari (tutti) che non soddisfano più il vincolo di eguaglianza di genere
* Si suppone che il numero di istanze presenti nel database dell’entità ESEMPLARE siano molto superiori al numero di istanze di ABITAZIONE, di conseguenza, la ridondanza dell’attributo non causa un eccessivo spreco di memoria secondaria

Conclusioni: è stato deciso di mantenere la ridondanza sull’attributo “genere”. Sarà necessario quindi:

* Gestire correttamente il cambiamento dell’attributo genere nell’entità ABITAZIONE: cambiare il genere assegnato ad una abitazione crea inconsistenza con gli esemplari ospitati nelle sue gabbie.
* Ad ogni assegnamento di un esemplare a una gabbia bisogna effettuare un controllo sul vincolo di integrità: “ogni gabbia assegnata a una abitazione deve contenere lo stesso genere presente nell’attributo GENERE dell’abitazione”

**Attributi numero abitazioni e numero gabbie**

L’attributo numero di abitazioni (attributo derivato) è ridondante in quanto è possibile calcolarlo contando quante volte una determinata area è in relazione “collocata” con delle abitazioni.

Si considera comunque opportuno mantenere l’attributo derivato (che dovrà essere aggiornato ad ogni aggiunta/rimozione di un’abitazione ad un’area) per abbattere il tempo di recupero di quest’informazione.

Le considerazioni sovrastanti risultano analoghe per l’attributo “numero gabbie”.

**Relazione assegnato**

La relazione “assegnato” è ridondante in quanto risulta possibile ricavare l’informazione relativa al genere di un’abitazione controllando il genere degli animali contenuti nelle sue gabbie.

Si considera opportuno mantenere la relazione assegnato in quanto l’operazione di lettura di quale genere sia assegnato ad una determinata abitazione risulta effettuato ogniqualvolta si voglia inserire/spostare un esemplare in una gabbia (operazione eseguita frequentemente).

<< aggiungere numero letture/scritture (accessi) con e senza ridondanze >>

## Schema relazionale

# 4. Progettazione fisica e implementazione

## Progettazione fisica

indici

## Implementazione

Il database è stato implementato utilizzando il DBMS ad oggetti PostgreSQL. Segue la descrizione delle varie fasi con relativi commenti.

**Il codice SQL è fornito in allegato alla relazione.**

### Definizione delle tabelle

Durante la definizione delle tabelle sono state fatte alcune scelte relative ai tipi dei dati. Più nello specifico è stato scelto di rappresentare:

* i nomi con variabili di tipo ***varchar(32)***
* gli id con il tipo di dato ***oid*** già fornito da PostgreSQL
* il sesso degli esemplari con un ***char***che può assumere solo valore ‘M’ o ‘F’
  + con relativo check per controllarne la correttezza:



* gli attributi *turno di pulizia*, *diagnostica* e *dieta* con ***varchar(1024)***.
* le date utilizzando il tipo di dato ***date*** già fornito da PostgreSQL
* i CF dei dipendenti con ***char(16)***(non viene effettuato un controllo sulla sua correttezza)

Segue un esempio di creazione di tabella: entità ESEMPLARE.

create table Esemplare(

    id                  *oid*,

    genere              varchar(32),

    nome                varchar(32) not null,

    sesso               varchar(1) *check*(sesso IN ( 'F' , 'M' )) not null,

    paese\_provenienza   varchar(32) not null,

    data\_nascita        date,

    data\_arrivo         date not null,

    gabbia              *oid* unique not null,

*constraint* pk\_Esemplare primary key(id,genere),

*constraint* fk\_genere\_Esemplare\_Genere *foreign key* (genere) *references* Genere(nome)

*on delete* restrict

        on update cascade,

*constraint* fk\_gabbia\_Esemplare\_Gabbia *foreign key* (gabbia) *references* Gabbia(id)

*on delete* restrict

        on update cascade

);

**Note:** è stato scelto di vietare la cancellazione di un genere affinché nel db siano presenti esemplari che ne fanno parte; analogamente non è possibile eliminare una gabbia se contiene un animale. Scelte simili sono state fatte per aree, abitazioni, visite etc.

### Definizione dei trigger

Segue una lista dei vincoli di integrità e operazioni da eseguire tramite l’ausilio di trigger.

|  |  |
| --- | --- |
| **ID** | **Controllo da eseguire** |
| **1** | All'aggiunta **(INSERT/UPDATE)** di un esemplare ad una gabbia bisogna controllare che l'abitazione in cui essa sia contenuta abbia il genere assegnato corretto (ovvero uguale). |
| **2** | Alla modifica (spostamento) **(UPDATE)** di una gabbia in una abitazione, bisogna controllare che il genere dell'animale in essa contenuto combaci con quello assegnato alla nuova abitazione di destinazione; similmente al controllo n°1  **NB:** non serve eseguire il check sull'inserimento perché è necessario prima aggiungere una gabbia e poi assegnargli un animale, di conseguenza, non è possibile assegnare una gabbia errata alla sua aggiunta in quanto sono sempre vuote durante la creazione. |
| **3** | All'aggiunta **(INSERT/UPDATE)** di un esemplare bisogna controllare che data arrivo >= data nascita. |
| **4** | All'aggiunta **(INSERT/UPDATE)** di una visita bisogna controllare che data visita > data arrivo esemplare. |
| **5** | Alla modifica di un genere assegnato **(UPDATE)** ad un'abitazione, bisogna controllare che non vengano violati i vincoli di genere (in riferimento al n°1)  **NB:** non serve il check sull'insert perché non è possibile inserire una abitazione già con delle gabbie (non c'è rischio che queste violino il vincolo di genere perché vengono aggiunte e controllate successivamente) |
| **6** | Alla modifica della data di arrivo di un esemplare bisogna controllare che questa sia coerente con le date delle visite: una visita non può essere stato effettuata prima che un esemplare sia arrivato nello zoo. |

Segue il codice della creazione dei trigger necessari per eseguire i controlli appena elencati.

create trigger aggiunta\_modifica\_esemplare *– per il controllo n°1,3,6*

before insert or update of data\_arrivo,data\_nascita,genere,gabbia on Esemplare

for each row

execute procedure aggiunta\_modifica\_esemplare();

create trigger modifica\_gabbia *– per il controllo n°2*

before update of abitazione on Gabbia

for each row

execute procedure modifica\_gabbia();

create trigger aggiunta\_modifica\_visita *– per il controllo n°4*

before insert or update of data on Visita

for each row

execute procedure aggiunta\_modifica\_visita();

create trigger modifica\_genere\_abitazione *– per il controllo n°5*

before update of genere on Abitazione

for each row

execute procedure modifica\_genere\_abitazione();

Verrà ora illustrato solo il funzionamento del primo trigger (e relativa funzione) in quanto in più complesso.

Il trigger **aggiunta\_modifica\_esemplare** causa l’esecuzione di una procedura che si assicura che i controlli n°1, 3 e 6 della tabella vengano passati con successo. Più nello specifico la procedura è chiamata ogni qual volta che:

* inseriamo un nuovo esemplare: quindi bisognerà controllare che non violi le tre condizioni
* aggiorniamo i campi ***data\_arrivo, data\_nascita, genere*** e ***gabbia*** (in quanto gli altri non rischiano di violare alcun vincolo).

La funzione che esegue i tre controlli è la seguente:

create or replace function aggiunta\_modifica\_esemplare()

returns trigger as $$ begin

*-- questa prima fase esegue il primo controllo della tab. --*

*In questa fase controllo che il genere dell’esemplare che sto aggiungendo/spostando sia lo stesso dell’abitazione di destinazione (che ottengo andando a guardare dov’è collocata la gabbia)*

    perform \*

    from(select    A.genere *- ottengo il genere assegnato all'abitazione contenente la gabbia.*

         from      Abitazione A

         where     A.id IN(select    G.abitazione *- ottengo l'id dell'abit. della gabbia in cui*

                             from    Gabbia G *sto inserendo l'esemplare*

                            where    G.id = new.gabbia)) genere\_ok

    where new.genere = genere\_ok.genere;

*-- questa seconda fase esegue il terzo controllo della tab. (nel caso il n°1 abbia avuto successo) –*

*Un semplice controllo sulle date, nel caso la condizione risulti VERA (e quindi le date non sono congrue)*

*viene lanciata un’eccezione (diversa in base al tipo di operazione).*

    if found then

        if(new.data\_arrivo <= new.data\_nascita) then *- dopo aver controllato il vincolo 1, controlliamo il*

*vincolo 3 (la coerenza delle date)*

            if(TG\_OP = 'UPDATE') then

               raise exception 'Operazione di UPDATE non consentita! La modifica delle date ha portato a

delle incongruenze! Vincolo da rispettare: la data di nascita deve essere

  antecedente o uguale alla data di arrivo';

            elseif(TG\_OP = 'INSERT') then

               raise exception 'Operazione di INSERT non consentita! L'esemplare possiede delle incongruenze

sulle date! Vincolo da rispettare: la data di nascita deve essere antecedente

o uguale alla data di arrivo';

            end if;

        end if; *- end if del controllo sulle date*

*-- questa terza fase esegue il sesto controllo della tab. (nel caso il n°3 abbia avuto successo)*

*Il perform va alla ricerca di visite la cui data di esecuzione risulterebbe antecedente alla nuova*

*data d’arrivo dell’esemplare, se ne vengono trovate viene lanciata un’eccezione*

        perform \*

        from    Visita V

        where   V.esemplare\_id = new.id and V.esemplare\_gen = new.genere and V.data < new.data\_arrivo;

        if found then

            raise exception 'Operazione di UPDATE non consentita! La modifica della data di arrivo ha causato

un'incongruenza: ci sono visite effettuate prima della nuova data di arrivo

dell'esemplare ma non si può aver visistato un esemplare prima che questo sia

arrivato nello zoo.';

        end if; *- end if del controllo sulle visite*

        return new; *- tutti I controlli sono andati a buon fine, ritorna NEW*

    end if; *- se il primo controllo non è andato a buon fine: eccezione sul genere*

    if(TG\_OP = 'UPDATE') then

        if(new.genere = old.genere) then

            raise exception 'Operazione di UPDATE non consentita! La gabbia in cui si vuole spostare

l’esemplare è contenuta in un abitazione il cui genere assegnato differisce

  da quello dell''esemplare';

        elseif(new.gabbia = old.gabbia) then

            raise exception 'Operazione di UPDATE non consentita! Il nuovo genere assegnato all''esemplare

non concide con quello assegnato all''abitazione in cui è contenuta la sua

gabbia';

        end if;

        raise exception 'Operazione di UPDATE non consentita! ';

    elseif(TG\_OP = 'INSERT') then

       raise exception 'Operazione di INSERT non consentita! La gabbia in cui si vuole inserire l'esemplare è contenuta in un abitazione il cui genere assegnato differisce da quello dell''esemplare';

    end if;

end; $$ language plpgsql;

### Funzioni SQL

# 5. Popolazione e analisi dei dati in R