# UNIVERSITÀ DEGLI STUDI DI UDINE

## Dipartimento di Scienze Matematiche, Informatiche e Fisiche

# Relazione Progetto Basi di Dati

### A cura di

* Roland GJOPALAJ 157277
* Giovanni PANTAROTTO 157707
* Cristian TOMASS 147813

## Indice degli argomenti

1. [Progettazione Concettuale](#progettazione-concettuale)
2. [Progettazione Logica](#progettazione-logica)
3. [Progettazione Fisica](#progettazione-fisica)
4. [Analisi con R](#analisi-con-r)
5. [Conclusioni](#conclusioni) # Progettazione Concettuale

## Progetto “Registro Automobilistico”

Si progetti uno schema entità/relazioni per la gestione di un registro automobilistico, facente parte del sistema informativo di un ufficio di motorizzazione, contenente le seguenti informazioni: - di ciascun veicolo interessa registrare la targa, la cilindrata, i cavalli fiscali, la velocità, il numero di posti e la data di immatricolazione; - i veicoli sono classificati in categorie (automobili, ciclomotori, camion, rimorchi, ecc.); - ciascun veicolo appartiene ad uno specifico modello; - tra i dati relativi ai veicoli, vi è la codifica del tipo di combustibile utilizzato; - di ciascun modello di veicolo è registrata la fabbrica di produzione e il numero delle versioni prodotte; - ciascun veicolo può avere uno o più proprietari, che si succedono nel corso della “vita” del veicolo; di ciascun proprietario interessa registrare cognome, nome e indirizzo di residenza.

Lo schema entità/relazioni dovrà essere completato con attributi “ragionevoli” per ciascuna entità, identificando le possibili chiavi e le relazioni necessarie per la gestione del sistema in esame. A partire dallo schema entità/relazioni, si costruisca il corrispondente schema relazionale.

### GLOSSARIO Termini

| Termine | Descrione | Sinonimi | Link |
| --- | --- | --- | --- |
| Veicolo | informazioni generali su un veicolo |  | Modello, Proprietario, Combustibile |
| Modello | Tipo di modello di veicolo |  | Veicolo, Fabbrica |
| Fabbrica | Azienda che procude un modello di veicolo |  | Modello |
| Proprietario | Chi ha posseduto e possiede un veicolo |  | Veicolo |
| Combustibile | Quale combustibile utilizza un veicolo |  | Veicolo |

### Frasi

**Frasi generiche** : > Si progetti uno schema entità/relazioni per la gestione di un registro automobilistico, facente parte del sistema informativo di un ufficio di motorizzazione, contenente le seguenti informazioni.

**Frasi relative al veicolo**: > Di ciascun veicolo interessa registrare la targa, la cilindrata, i cavalli fiscali, la velocità, il numero di posti e la data di immatricolazione.

**Frasi relative al modello**: > Ciascun veicolo appartiene ad uno specifico modello.

**Frasi relative alla fabbrica**: > Di ciascun modello di veicolo è registrata la fabbrica di produzione e il numero delle versioni prodotte.

**Frasi relative al proprietario**: > Ciascun veicolo può avere uno o piú proprietari, che si succedono nel corso della “vita” del veicolo; di ciascun proprietario interessa registrare cognome, nome e indirizzo di residenza.

**Frasi relative al combustibile**: > Di ciascun veicolo interessa registrare il tipo di combustibile utilizzato.

## Modello ER

La strategia che abbiamo utilizzato per costruire il modello ER è la strategia Mista (Mixed Strategy). Abbiamo optato per questa tecnica perché ci sembrava comodo avere uno scheletro iniziale che ci permettesse di avere una visione di base dello schema. Inizialmente abbiamo usato la strategia bottom-up per assemblare insieme tutte le entità. In seguito, abbiamo usato la tecnica top-down, in questo modo, tramite perfezionamenti, abbiamo sviluppato lo schema finale.

La prima entità che abbiamo esaminato è **Veicolo**. Gli attributi che abbiamo aggiunto a questa entità sono targa, cilindrata, cavalli fiscali, velocità, numero di posti e data immatricolazione. Questa entità viene identificata univocamente dall’attributo targa.

In seguito per quanto riguarda Veicolo abbiamo deciso di fare una **generalizzazione** **Totale** e **Disgunta**. In questo modo un veicolo puo` essere distinto tra Automobili, Ciclomotori, Camion e Rimorchio. La generalizzazione e’ disgunta perche’ veicolo puo’ ricoprire solo una delle quattro categorie.

La generalizzazione è stata gestita nel modo sotto riportato aggiungendo degli attributi per ogni entità figlia.

**Veicolo** (Entità genitore): \* **Automobile** (Entità figlia)  
\* tipologia  
\* **Camion** (Entità figlia)  
\* numero assi  
\* **Ciclomotore** (Entità figlia)  
\* bauletto  
\* **Rimorchio** (Entità figlia)  
\* tipologia  
\* carico

L’entità successiva che abbiamo analizzato è **Modello**. Gli attributi che abbiamo aggiunto a questa entità sono *idModello*, *nomeModello* e *numeroVersioni*. Questa entità viene identificata univocamente dall’attributo id\_modello.

La prossima entità che abbiamo visto è **Fabbrica**. A livello conettuale abbiamo gestito fabbrica come il marchio di un modello di veicolo, per esempio: Audi, BMW, Fiat, etc. Gli attributi che abbiamo aggiunto a questa entità sono *idFabbrica*, *nome* e *numeroVeicoloProdotti*. Questa entità viene identificata univocamente dall’attributo idFabbrica. Noi abbiamo scelto “numeroVeicoliProdotti” come l’attributo ridondante che in seguito verifecheremo se si può tenere o meno.

S

L’entità successiva è **Combustibile**. Gli attributi che abbiamo aggiunto a questa entità sono codiceCombustibile e tipoCombustibile. Questa entità viene identificata univocamente dall’attributo **codiceCombustibile** e descritta dall’attributo **TipoCombustibile**. Tra codiciCombustibile abbiamo messo un caso particolare **TR** che sarebbe il codiceCombustibile per i rimorchi che vengono trainati. In questo modo risolviamo il problema dei rimorchi che usano combustibile.

Infine l’ultima entità è **Proprietario**. Gli attributi che abbiamo aggiunto a questa entità sono *idProprietario* e *indirizzo*. Questa entità viene identificata univocamente dall’attributo idProprietario. Per comodità abbiamo gestito indirizzo come una unica stringa ma si poteva pensare di utilizzare un attributo composto che rappresentasse la via, il numero civico e la città.

Come aggiunta abbiamo deciso di aggiungere una generalizzazione su proprietario che ci permetta di identificare se è **Privato**, quindi una persona fisica o se appartiene ad una **Società**, quindi un veicolo aziendale. La generalizzazione e’ disgunta perche’ proprietario puo’ ricoprire solo una delle due categorie.

In queste generalizzazione abbiamo pensato di ricavare gli attributi di Privato tramite Proprietario e aggiungere gli attributi di Societa partita iva e nome. Proprietario (Entità genitore): \* **Privato** (Entità figlia)  
\* CF: not NULL  
\* nome not NULL \* cognome not NULL \* data di nascita  
\* **Societa** (Entità figlia)  
\* partita iva

Tra **Modello** e **Veicolo** è presente una relazione uno a molti “**Appartiene**” → Tra Veicolo (1:1) e Modello (1:N)

* Un veicolo può appartenere solo ad un modello.
* Più veicoli possono essere dello stesso modello.
* Per ogni modello deve esserci almeno un veicolo che appartiene a quel modello per essere presente nel database

Tra **Modello** e **Fabbrica** è presente una relazione uno a molti “**Prodotto**” → Tra Fabbrica (1:1) e Modello (1:N)

* Una fabbrica produce solo un modello.
* Più fabbriche possono produrre lo stesso modello.
* Per ogni modello deve esserci almeno una fabbrica che produce quel modello per essere presente nel database

Tra **Veicolo** e **Combustibile** è presente una relazione uno a molti “**Utilizza**” → Tra Veicolo (1:1) e Combustibile (1:N)

* Un veicolo utilizza solo un tipo di combustibile.
* Più veicoli possono utilizzare lo stesso tipo di combustibile.
* Un tipo di combustibile deve essere utilizzato da almeno un veicolo per poter essere presente nel database

Tra **Veicolo** e **Proprietario** sono presenti due relazioni: 1) Uno a molti “**Proprietario Corrente**” → Tra Veicolo (1:1) e Proprietario (0:N) - Un veicolo appartiene solo un proprietario corrente.  
- Un proprietario può non avere un veicolo al momento oppure avere più di uno - Data è l’attributo che identifica la data dell’acquisto del veicolo al proprietario corrente 1) Molti a molti “**Proprietario Passato**” → Tra Veicolo (0:N) e Proprietario (0:N) - Un veicolo può avere o no più proprietari passati - Un proprietario può avere o no più veicoli nel passato - Data acquisto e data vendita sono attributi che identificano la data dell’acquisto e vendita del veicolo al proprietario corrente - Se un proprietario compra e vende la stessa macchina due volte allora si registrano solo le date dell’ultima occorrenza.

“Proprietario Passato” ci risolve la questione della “vita” di un veicolo

## Schema ER finale

## Lista dei Vincoli

### Vincoli generali

1. La data di acquisto deve essere antecedente rispetto alla data di vendita.
2. La data di acquisto deve essere successiva rispetto alla data di immatricolazione.
3. La data di vendita deve essere antecedente rispetto alla data di acquisto del veicolo successivo.
4. La data di acquisto di un proprietario passato deve essere precedente alla data di acquisto del proprietario corrente.
5. Un proprietario non può vendere un veicolo che non possiede.
6. Un veicolo non può essere acquistato da un proprietario se è già stato acquistato da un altro proprietario nello stesso giorno.

## Regole di Gestione

Di seguito sono elencate le regole di gestione usate nello schema ER. 1) num\_veicoli\_prodotti: numero di veicoli prodotti dalla fabbrica

Altre idee di regole di gestione che potrebbero essere utilizzate: 1) L’età di un veicolo è la differenza tra la data odierna e la data di immatricolazione. 2) Il numero di anni di proprietà di un veicolo è la differenza tra la data di acquisto e la data di vendita.

# Progettazione Logica

## Operazioni richeste

* **Op1**: Aggiunta nuovo veicolo prodotto [15 al giorno]
* **Op2**: Calcolare tutti i dati relativi alla fabbrica soprattutto il numero dei veicoli prodotti [2 al giorno] ### Tabella volumi | Concetto | Tipo | Volume | |— | — | — | | Veicolo | E | 90000 | | Proprietario | E | 125000 | | Combustibile | E | 5 | | Modello | E | 200 | | Fabbrica | E | 10 | | ProprietarioCorrente | R | 90000 | | ProprietariPassati | R | 225000 | | Appartiene | R | 90000 | | Prodotto | R | 200 | | Utilizza | R | 90000 |

Questa tabella ci fornisce una visione chiara della dimensione del sistema che stiamo progettando. Notiamo che:

* Il rapporto tra veicoli e proprietari è di circa 0.75, indicando che in media una persona possiede meno di un veicolo.
* Ogni veicolo ha avuto in media 2.5 proprietari nel passato, suggerendo un’alta frequenza di cambi di proprietà.

### Analisi delle ridondanze

**Caso studio: Numero di veicoli prodotti per Fabbrica**

#### Presenza di ridondanza

Consideriamo l’attributo “numeroVeicoliProdotti” nell’entità Fabbrica. Questo è un dato derivabile ma potrebbe essere utile mantenerlo come ridondanza per migliorare le prestazioni.

Per eseguire il calcolo delle operazione in presenza di ridondanze si fa il calcolo di ogni micro processo:

* OP1: Aggiunta nuovo veicolo (15 volte al giorno):
  + Memorizzo il nuovo veicolo
  + memorizzo la coppia veicolo-modello
  + cerco il modello e per risalire alla fabbrica
  + cerca la fabbrica di interesse
  + incremento di uno i veicoli prodotti

| Concetto | Costrutto | Accessi | Tipo |
| --- | --- | --- | --- |
| Veicolo | E | 1 | S |
| Appartiene | R | 1 | S |
| Modello | E | 0 |  |
| Prodotto | R | 1 | L |
| Fabbrica | E | 1 | L |
| Fabbrica | E | 1 | S |

Totale: 3 scritture + 2 letture = 8 accessi/operazione  
Costo giornaliero: 8 \* 15 = 120 accessi

* OP2: Visualizzazione dati Fabbrica (2 volte al giorno):
  + Leggere gli attributi della fabbrica

| Concetto | Costrutto | Accessi | Tipo |
| --- | --- | --- | --- |
| Fabbrica | E | 1 | L |

Costo giornaliero: 1 \* 2 = 2 accessi

**Costo totale giornaliero con ridondanza: 122 accessi**

### Assenza di ridondanza

* OP1: Aggiunta nuovo veicolo (15 volte al giorno):
  + Memorizzo il nuovo veicolo
  + Memorizzo la coppia veicolo modello

| Concetto | Costrutto | Accessi | Tipo |
| --- | --- | --- | --- |
| Veicolo | E | 1 | S |
| Appartiene | R | 1 | S |
| Modello | E | 0 |  |
| Prodotto | R | 0 |  |
| Fabbrica | E | 0 |  |

Totale: 2 scritture = 4 accessi/operazione  
Costo giornaliero: 4 \* 15 = 60 accessi

* OP2: Visualizzazione dati Fabbrica (2 volte al giorno):
  + Per calcolare il numero di veicoli prodotti da una fabbrica dobbiamo accedere alla relazione “prodotto” un numero di volte pare al numero medio di veicoli prodotti da una certa fabbrica (dalla fabbrica): nrModelli/nrFabbriche (200/10) **e per ogni di questi modelli** bisogna accedere un nr di volte pari al numero medio di veicoli appertenenti ad un modello : nrVeicoli /nrModelli (90000/200)

| Concetto | Costrutto | Accessi | Tipo |
| --- | --- | --- | --- |
| Fabbrica | E | 1 | L |
| Prodotto | E | 20 | L |
| Appartiene | E | 9000 (20\*450) | L |

Costo per operazione: 1 + 20 + 9000 = 9021 letture  
Costo giornaliero: 9021 \* 2 = 18042 accessi

**Costo totale giornaliero senza ridondanza: 60 + 18042 = 18102 accessi**

### Costi operazione

Presenza di ridondanza ⟹

Assenza di ridondanza ⟹

### **Conclusione dell’analisi**

Mantenere la ridondanza comporta un costo giornaliero di 122 accessi, mentre eliminarla porta a 18102 accessi.  
La differenza è significativa: mantenere la ridondanza riduce il carico di lavoro di circa il 99.3%. Pertanto, è altamente consigliabile mantenere l’attributo ridondante “numeroVeicoliProdotti” nell’entità Fabbrica.

### **Eliminazione delle generalizzazioni**

In questa fase del progetto sono state gestite le generalizzazioni presenti eliminando le gerarchie. In particolare sono state trasformate le seguenti parti:

**Veicolo**

Abbiamo optato per una strategia di accorpamento nel genitore. Questa scelta è motivata dal fatto che la maggior parte delle operazioni coinvolgerà attributi comuni a tutti i tipi di veicolo.

**Proprietario**

Anche per proprietario abbiamo scelto la stessa strategia.

### Schema ER Finale senza generalizzazioni

### **Partizionamento o accorpamento**

Per quanto riguarda l’entità Proprietario, abbiamo optato per mantenere l’attributo ‘indirizzo’ come un campo di testo unico, invece di partizionarlo verticalmente in componenti separate (come via, numero civico, CAP, città). Questa decisione è stata presa considerando che l’indirizzo viene generalmente utilizzato come un’unica unità informativa nelle operazioni più frequenti, e la sua scomposizione non offrirebbe vantaggi significativi in termini di prestazioni o funzionalità per il nostro specifico caso d’uso.

### **Selezione degli identificatori**

| Entità | Chiavi |
| --- | --- |
| Veicolo | Targa |
| Combustibile | codiceCombustibile |
| Proprietario | idProprietario |
| Modello | idModello |
| Fabbrica | idFabbrica |

La scelta degli identificatori è stata fatta considerando l’unicità, l’immutabilità e la semplicità di gestione.

### Traduzione modello logico

* fabbrica {***id\_fabbrica*** (PK), nome, numero\_veicolo\_prodotti}
* modello {***id\_modello*** (PK), nome\_modello, numero\_versioni, ***fabbrica\_di\_produzione*** (FK → fabbrica.\_id\_fabbrica\_)}
* combustibile {***codice\_combustibile*** (PK), tipo\_combustibile}
* proprietario {***id\_proprietario*** (PK), indirizzo}
* privato {***id\_proprietario*** (PK, FK → proprietario.\_id\_proprietario\_), cf, nome, cognome, data\_nascita}
* societa {***id\_proprietario*** (PK, FK → proprietario.\_id\_proprietario\_), partita\_iva}
* veicolo {***targa*** (PK), cavalli, velocita, numero\_posti, data\_immatricolazione, cilindrata, data, ***modello*** (FK → modello), ***codice\_combustibile*** (FK → combustibile.\_codice\_combustibile\_), ***proprietario*** (FK → proprietario.\_id\_proprietario\_)}
* proprietari\_passati {***targa*** (PK, FK → veicolo.\_targa\_), ***id\_proprietario*** (PK, FK → proprietario.\_id\_proprietario\_), data\_vendita, data\_acquisto}
* automobile {***targa*** (PK, FK → veicolo.\_targa\_), tipologia}
* ciclomotore {***targa*** (PK, FK → veicolo.\_targa\_), bauletto}
* camion {***targa*** (PK, FK → veicolo.\_targa\_), numero\_assi}
* rimorchio {***targa*** (PK, FK → veicolo.\_targa\_), tipologia, carico}

**IMPORTANTE:** Le chiavi primarie (PK) e le chiavi esterne (FK) non possono essere NULL

### Vincoli di Chiave

##### Chiavi Primarie (PK): Devono essere not null e univoche

* Veicolo.Targa
* Modello.idModello
* Fabbrica.idFabbrica
* Combustibile.codiceCombustibile
* Proprietario.IdProprietario

##### Chiavi Esterne (FK):

* Veicolo.Modello → Modello.idModello
* Veicolo.CodiceCombustibile → Combustibile.codiceCombustibile
* Veicolo.Proprietario → Proprietario.IdProprietario
* Modello.FabbricaDiProduzione → Fabbrica.idFabbrica
* ProprietariPassati.Targa → Veicolo.Targa
* ProprietariPassati.IdProprietario → Proprietario.IdProprietario

### Vincoli di Entità

(Questi dati sono importanti e devono essere presenti) - Veicolo.dataImmatricolazione NOT NULL - Veicolo.dataAquisto NOT NULL - Veicolo.Cilindrata: NOT NULL  
- Veicolo.Cavalli: NOT NULL  
- Veicolo.Velocità: NOT NULL  
- Veicolo.NumeroPosti: NOT NULL - Modello.numeroVersioni: NOT NULL - Proprietario.indirizzo: NOT NULL -

Privato.CF: NOT NULL

* Privato.nome NOT NULL
* Privato.cognome NOT NULL
* Società.partitaIva NOT NULL
* ProprietariPassati.dataAcquisto NOT NULL
* ProprietariPassati.dataVendita NOT NULL

### Vincoli di Dominio

* Modello.numeroVersioni > 0
* Veicolo.cilindrata > 0 (se Rimorchio allora 0)
* Veicolo.cavalli > 0 (se Rimorchio allora 0)
* Veicolo.numeroPosti > 0 (se Rimorchio allora 0)
* Veicolo.velocita’ > 0 (se Rimorchio allora 0)

### Vincoli di Generalizzazione

**Totalità e disgiunzione** - Ogni proprietario deve essere solo Privato o Società - Ogni veicolo deve comparire in esattamente una tabella figlia: (Automobile, Camion, Ciclomotore, Rimorchio)

### Vincoli di Integrità Referenziale

* Non posso cancellare nessun oggetto (record) se viene puntato da una chiave esterna
  + esempio: non posso cancellare un proprietario se lui possiede una veicolo che si trova nel database
  + altro esempio: non posso cancellare una fabbrica se esiste un modello che e’ stato prodotto da quella fabbrica

### Vincoli di Tupla

* ProprietariPassati: CHECK (dataVendita > dataAcquisto)
* ProprietariPassati: UNIQUE (Targa, IdProprietario)
  + se un proprietario compra e vende più di una volta lo stesso veicolo si registra solo l’ultima occorrenza

### Vincoli di partecipazione

* Affinchè una fabbrica sia presente nel database deve comparire almeno in uno dei modelli presenti
* Affinchè un modello sia presente nel database deve comparire almeno in uno dei veicoli presenti
* Affinchè un tipo di combustibile sia presente nel database deve comparire almeno in uno dei veicoli presenti
* Un veicolo deve per forza avere un proprietario corrente (uno solo)
* Un proprietario può non avere un veicolo al momento

### Vincoli Inter-tabella (Ridondanza Controllata)

* Trigger per aggiornare Fabbrica.numeroVeicoliProdotti quando viene inserito un nuovo veicolo

## Progettazione Fisica

### Definizione database in SQL

In seguito abbiamo creato la base di dati. Sotto mostriamo tutto il codice SQL.

#### Creazione database

CREATE DATABASE "registro automobilistico"  
 WITH  
 OWNER = postgres  
 ENCODING = 'UTF8'  
 LOCALE\_PROVIDER = 'libc'  
 CONNECTION LIMIT = -1  
 IS\_TEMPLATE = False;

#### Creazione tabelle

CREATE TABLE fabbrica (  
 id\_fabbrica INT PRIMARY KEY,  
 nome VARCHAR(50) NOT NULL,  
 numero\_veicoli\_prodotti INT DEFAULT 0  
);  
  
CREATE TABLE modello (  
 id\_modello INT PRIMARY KEY,  
 nome\_modello VARCHAR(50),  
 numero\_versioni INT NOT NULL CHECK (numero\_versioni > 0),  
 fabbrica\_di\_produzione INT NOT NULL,  
 FOREIGN KEY (fabbrica\_di\_produzione) REFERENCES fabbrica(id\_fabbrica)  
);  
  
CREATE TABLE combustibile (  
 codice\_combustibile VARCHAR(15) PRIMARY KEY,  
 tipo\_combustibile VARCHAR(20)  
);  
  
CREATE TABLE proprietario (  
 id\_proprietario INT PRIMARY KEY,  
 indirizzo VARCHAR(255) NOT NULL  
);  
  
CREATE TABLE privato (  
 id\_proprietario INT PRIMARY KEY,  
 cf VARCHAR(16) NOT NULL UNIQUE,  
 nome VARCHAR(50) NOT NULL,  
 cognome VARCHAR(50) NOT NULL,  
 data\_nascita DATE,  
 FOREIGN KEY (id\_proprietario) REFERENCES proprietario(id\_proprietario)  
);  
  
CREATE TABLE societa (  
 id\_proprietario INT PRIMARY KEY,  
 partita\_iva VARCHAR(16) NOT NULL UNIQUE,  
 FOREIGN KEY (id\_proprietario) REFERENCES proprietario(id\_proprietario)  
);  
  
CREATE TABLE veicolo (  
 targa VARCHAR(10) PRIMARY KEY,  
 cilindrata INT NOT NULL,  
 cavalli INT NOT NULL,  
 velocita INT NOT NULL,  
 numero\_posti INT NOT NULL,  
 data\_immatricolazione DATE NOT NULL,  
 data DATE NOT NULL,  
 modello INT NOT NULL,  
 codice\_combustibile VARCHAR(15) NOT NULL,  
 proprietario INT NOT NULL,  
 CHECK (  
 (cilindrata >= 0) AND   
 (cavalli >= 0) AND   
 (velocita >= 0) AND   
 (numero\_posti >= 0)  
 ),  
 FOREIGN KEY (modello) REFERENCES modello(id\_modello),  
 FOREIGN KEY (codice\_combustibile) REFERENCES combustibile(codice\_combustibile),  
 FOREIGN KEY (proprietario) REFERENCES proprietario(id\_proprietario)  
);  
  
CREATE TABLE automobile (  
 targa VARCHAR(10) PRIMARY KEY,  
 tipologia VARCHAR(20),  
 FOREIGN KEY (targa) REFERENCES veicolo(targa)  
);  
  
CREATE TABLE ciclomotore (  
 targa VARCHAR(10) PRIMARY KEY,  
 bauletto BOOLEAN,  
 FOREIGN KEY (targa) REFERENCES veicolo(targa)  
);  
  
CREATE TABLE camion (  
 targa VARCHAR(10) PRIMARY KEY,  
 numero\_assi INT,  
 FOREIGN KEY (targa) REFERENCES veicolo(targa)  
);  
  
CREATE TABLE rimorchio (  
 targa VARCHAR(10) PRIMARY KEY,  
 tipologia VARCHAR(20),  
 carico INT,  
 FOREIGN KEY (targa) REFERENCES veicolo(targa)  
);  
  
CREATE TABLE proprietari\_passati (  
 targa VARCHAR(10),  
 id\_proprietario INT,  
 data\_acquisto DATE NOT NULL,  
 data\_vendita DATE NOT NULL,  
 PRIMARY KEY (targa, id\_proprietario),  
 CHECK (data\_vendita > data\_acquisto),  
 FOREIGN KEY (targa) REFERENCES veicolo(targa),  
 FOREIGN KEY (id\_proprietario) REFERENCES proprietario(id\_proprietario)  
);

#### Definizione trigger

-- Trigger --  
--controllo nr\_veicoli\_prodotti--  
CREATE OR REPLACE FUNCTION aggiorna\_conteggio\_veicoli()   
RETURNS TRIGGER AS $$  
BEGIN  
 UPDATE fabbrica  
 SET numero\_veicoli\_prodotti = numero\_veicoli\_prodotti + 1  
 WHERE id\_fabbrica = (  
 SELECT fabbrica\_di\_produzione   
 FROM modello   
 WHERE id\_modello = NEW.modello  
 );  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER trigger\_conteggio\_veicoli  
AFTER INSERT ON veicolo  
FOR EACH ROW EXECUTE FUNCTION aggiorna\_conteggio\_veicoli();  
  
-------------------------------------------------------------------------------  
-- Trigger per verificare mutua esclusione PRIVATO  
CREATE OR REPLACE FUNCTION check\_privato\_mutua\_esclusione()  
RETURNS TRIGGER AS $$  
BEGIN  
 -- Verifica se esiste già in societa  
 IF EXISTS (SELECT 1 FROM societa WHERE id\_proprietario = NEW.id\_proprietario) THEN  
 RAISE EXCEPTION 'Mutua esclusione violata: ID % è già registrato come società', NEW.id\_proprietario;  
 END IF;  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER trigger\_check\_privato  
BEFORE INSERT OR UPDATE ON privato  
FOR EACH ROW EXECUTE FUNCTION check\_privato\_mutua\_esclusione();  
  
-- Trigger per verificare mutua esclusione SOCIETA  
CREATE OR REPLACE FUNCTION check\_societa\_mutua\_esclusione()  
RETURNS TRIGGER AS $$  
BEGIN  
 -- Verifica se esiste già in privato  
 IF EXISTS (SELECT 1 FROM privato WHERE id\_proprietario = NEW.id\_proprietario) THEN  
 RAISE EXCEPTION 'Mutua esclusione violata: ID % è già registrato come privato', NEW.id\_proprietario;  
 END IF;  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER trigger\_check\_societa  
BEFORE INSERT OR UPDATE ON societa  
FOR EACH ROW EXECUTE FUNCTION check\_societa\_mutua\_esclusione();

### Popolazione base di dati

### Query

1. Il veicolo con il maggior numero di cavalli che ha avuto 1 e un solo proprietario.

SELECT v.\*   
FROM veicolo v  
LEFT JOIN proprietari\_passati pp ON v.targa = pp.targa  
WHERE pp.targa IS NULL  
ORDER BY v.cavalli DESC  
LIMIT 1;

*Output:*

1. Le società che è un proprietario passato di esattamente 2 veicoli

SELECT s.partita\_iva, COUNT(pp.targa) AS num\_veicoli  
FROM societa s  
JOIN proprietari\_passati pp ON s.id\_proprietario = pp.id\_proprietario  
GROUP BY s.partita\_iva  
HAVING COUNT(pp.targa) = 2;

*Output:*

1. Tutti i veicoli prodotti da fabbriche che hanno prodotto esattamente 3 modelli.

SELECT v.\*   
FROM veicolo v  
JOIN modello m ON v.modello = m.id\_modello  
WHERE m.fabbrica\_di\_produzione IN (  
 SELECT fabbrica\_di\_produzione   
 FROM modello   
 GROUP BY fabbrica\_di\_produzione   
 HAVING COUNT(\*) = 3  
);

SELECT V.targa  
FROM veicolo AS V  
JOIN modello AS M ON V.modello = M.id\_modello  
JOIN fabbrica AS F ON M.fabbrica\_di\_produzione = F.id\_fabbrica  
WHERE F.id\_fabbrica IN (  
 SELECT F2.id\_fabbrica  
 FROM fabbrica F2  
 WHERE EXISTS (  
 SELECT \*  
 FROM modello M2  
 WHERE M2.fabbrica\_di\_produzione = F2.id\_fabbrica  
 AND EXISTS (  
 SELECT \*  
 FROM modello M3  
 WHERE M3.fabbrica\_di\_produzione = F2.id\_fabbrica  
 AND M3.id\_modello <> M2.id\_modello  
 AND EXISTS (  
 SELECT \*  
 FROM modello M4  
 WHERE M4.fabbrica\_di\_produzione = F2.id\_fabbrica  
 AND M4.id\_modello <> M2.id\_modello  
 AND M4.id\_modello <> M3.id\_modello  
 AND NOT EXISTS (  
 SELECT \*  
 FROM modello M5  
 WHERE M5.fabbrica\_di\_produzione = F2.id\_fabbrica  
 AND M5.id\_modello <> M2.id\_modello  
 AND M5.id\_modello <> M3.id\_modello  
 AND M5.id\_modello <> M4.id\_modello  
 )  
 )  
 )  
 )  
);

*Output:*

1. Il numero dei veicoli in cui il proprietario corrente è anche un proprietario passato

SELECT COUNT(\*) AS numero\_veicoli  
FROM veicolo v  
JOIN proprietari\_passati pp ON v.targa = pp.targa  
WHERE v.proprietario = pp.id\_proprietario;

*Output:*

1. La fabbrica con il massimo numero di veicoli elettrici.

SELECT f.nome, COUNT(\*) AS num\_elettrici  
FROM veicolo v  
JOIN modello m ON v.modello = m.id\_modello  
JOIN fabbrica f ON m.fabbrica\_di\_produzione = f.id\_fabbrica  
WHERE v.codice\_combustibile = 'ELET'  
GROUP BY f.nome  
ORDER BY num\_elettrici DESC  
LIMIT 1;

oppure con le viste

CREATE VIEW veicoli\_elettrici\_per\_fabbrica AS (  
 SELECT m.fabbrica\_di\_produzione, COUNT(v.targa) AS numero\_veicoli\_elettrici  
 FROM veicolo v  
 JOIN modello m ON v.modello = m.id\_modello  
 WHERE v.codice\_combustibile = 'ELET'  
 GROUP BY m.fabbrica\_di\_produzione  
);  
SELECT fabbrica.nome, numero\_veicoli\_elettrici  
FROM veicoli\_elettrici\_per\_fabbrica, fabbrica  
WHERE fabbrica.id\_fabbrica=fabbrica\_di\_produzione AND numero\_veicoli\_elettrici = (  
 SELECT MAX(numero\_veicoli\_elettrici)  
 FROM veicoli\_elettrici\_per\_fabbrica  
);

*Output:*

1. Data una targa, cercare il veicolo o, il proprietario corrente (Cf se privato altrimenti IVA) e tutti i proprietari passatti (Cf se privati altrimenti IVA) e le date di aquisto e vendita. *Servirebbe per rapportare i dati prima di cancellare un veicolo*

SELECT --Recupero del proprietario corrente, deve anche trovare se e' un privato o una societa'  
 v.targa,  
 p.id\_proprietario,  
 CASE  
 WHEN pr.id\_proprietario IS NOT NULL   
 THEN CONCAT('Privato CF ', pr.cf)  
 WHEN s.id\_proprietario IS NOT NULL   
 THEN CONCAT('Società P.IVA ', s.partita\_iva)  
 END AS proprietario,  
 v.data AS data\_acquisto,  
 NULL AS data\_vendita,  
 'Ultimo proprietario' AS stato  
FROM veicolo v  
JOIN proprietario p ON v.proprietario = p.id\_proprietario  
LEFT JOIN privato pr ON p.id\_proprietario = pr.id\_proprietario  
LEFT JOIN societa s ON p.id\_proprietario = s.id\_proprietario  
WHERE v.targa = 'GC908BT'  
  
UNION ALL  
  
SELECT --Recupero dei proprietario correnti, deve anche trovare se sono dei privati o delle societa'  
 pp.targa,  
 pp.id\_proprietario,   
 CASE  
 WHEN pr.id\_proprietario IS NOT NULL   
 THEN CONCAT('Privato CF ', pr.cf)  
 WHEN s.id\_proprietario IS NOT NULL   
 THEN CONCAT('Società IVA ', s.partita\_iva)  
 END AS proprietario,  
 pp.data\_acquisto,  
 pp.data\_vendita,  
 'Ex proprietario' AS stato  
FROM proprietari\_passati pp  
JOIN proprietario p ON pp.id\_proprietario = p.id\_proprietario  
LEFT JOIN privato pr ON p.id\_proprietario = pr.id\_proprietario  
LEFT JOIN societa s ON p.id\_proprietario = s.id\_proprietario  
WHERE pp.targa = 'GC908BT'  
  
ORDER BY data\_acquisto DESC;

*Output:*

### Query di inserimento/cancellazione/aggiornamento

Inserimento e aggiornamento: - passagio di proprieta

WITH vecchio\_proprietario AS (  
 SELECT targa, proprietario, data   
 FROM veicolo   
 WHERE targa = 'AB123CD'  
),  
registrazione\_storico AS (  
 INSERT INTO proprietari\_passati (targa, id\_proprietario, data\_acquisto, data\_vendita)  
 SELECT targa, proprietario, data, CURRENT\_DATE   
 FROM vecchio\_proprietario  
)  
UPDATE veicolo   
SET   
 proprietario = 78,  
 data = CURRENT\_DATE  
WHERE targa = 'AB123CD';

Cancellazione: - eliminare tutti i proprietari (privati e società) senza veicoli attuali o storici

DELETE FROM societa  
WHERE id\_proprietario IN (  
 SELECT p.id\_proprietario  
 FROM proprietario p  
 LEFT JOIN veicolo v ON p.id\_proprietario = v.proprietario  
 LEFT JOIN proprietari\_passati pp ON p.id\_proprietario = pp.id\_proprietario  
 WHERE v.targa IS NULL AND pp.targa IS NULL  
);  
  
DELETE FROM privato  
WHERE id\_proprietario IN (  
 SELECT p.id\_proprietario  
 FROM proprietario p  
 LEFT JOIN veicolo v ON p.id\_proprietario = v.proprietario  
 LEFT JOIN proprietari\_passati pp ON p.id\_proprietario = pp.id\_proprietario  
 WHERE v.targa IS NULL AND pp.targa IS NULL  
);  
  
DELETE FROM proprietario  
WHERE id\_proprietario NOT IN (  
 SELECT id\_proprietario FROM veicolo  
 UNION  
 SELECT id\_proprietario FROM proprietari\_passati  
);  
  
COMMIT;

## Analisi con R

### Numero di modelli prodotti da ogni fabbrica

result <- dbGetQuery(pg\_connection,  
 "SELECT f.nome AS fabbriche,count(m.id\_modello) as numeroModelli  
 FROM fabbrica as f, modello as m  
 WHERE f.id\_fabbrica = m.fabbrica\_di\_produzione  
 GROUP BY f.nome")  
  
result$numeromodelli <- as.integer(result$numeromodelli)  
  
# Creazione del grafico a barre con ggplot2  
ggplot(data = result, aes(x = fabbriche, y = numeromodelli, fill = numeromodelli)) +  
 geom\_bar(stat = "identity", color = "black") +  
 xlab("Fabbriche") +  
 ylab("Numero di Modelli") +  
 ggtitle("Numero di Modelli per Fabbrica") +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +  
 scale\_fill\_gradient(low = "lightblue", high = "darkblue")

### Distribuzione dei veicoli per categoria

result <- dbGetQuery(pg\_connection,  
 "SELECT  
 CASE  
 WHEN targa IN (SELECT targa FROM automobile) THEN 'automobile'  
 WHEN targa IN (SELECT targa FROM camion) THEN 'camion'  
 WHEN targa IN (SELECT targa FROM ciclomotore) THEN 'ciclomotore'  
 WHEN targa IN (SELECT targa FROM rimorchio) THEN 'rimorchio'  
 ELSE 'altro'  
 END AS categoria,  
 COUNT(\*) AS numero\_veicoli  
 FROM veicolo  
 GROUP BY categoria")  
  
ggplot(result, aes(x = "", y = numero\_veicoli, fill = categoria)) +  
 geom\_bar(stat = "identity", width = 1) +  
 coord\_polar("y", start = 0) +  
 labs(x = NULL, y = NULL, fill = "Categoria", title = "Distribuzione dei Veicoli per Categoria") +  
 theme\_void() +  
 theme(plot.title = element\_text(hjust = 0.5))

### Distribuzione dei Veicoli per cavalli

result <- dbGetQuery(pg\_connection,  
 "select cavalli, count(targa) from veicolo group by cavalli")  
  
ggplot(result, aes(x = cavalli, y = count)) +  
 geom\_bar(stat = "identity", fill = "aquamarine4") +  
 labs(title = "Distribuzione dei veicoli per cavalli",  
 x = "Cavalli",  
 y = "Numero di targhe") +  
 theme\_minimal()

## Conclusioni

In conclusione, il progetto ha permesso di sviluppare un sistema di gestione per un registro automobilistico, basato su una progettazione accurata e ottimizzata per garantire efficienza e integrità dei dati. Attraverso l’analisi dei modelli concettuali, logici e fisici, sono state implementate soluzioni robuste per gestire le complessità del dominio, dimostrando l’efficacia di un approccio metodico nella progettazione di basi di dati.