Università degli Studi di Udine –
Dipartimento di Scienze Matematiche,
Informatiche e Fisiche

Base di dati per la gestione della rete di stazioni per il rifornimento di carburante presenti nel territorio del Friuli Venezia Giulia

Ilaria Fenos
matr. 142494
fenos.ilaria@spes.uniud.it

Andrea Antonutti
matr. 142792
antonutti.andrea001@spes.uniud.it

Andrea Roncali
matr. 133087
roncali.andrea@spes.uniud.it

Jia Ping Huang
matr. 144762
huang.jiaping@spes.uniud.it

A.A. 2020/2021

Indice

1 Richiesta

Si vuole progettare una base di dati di supporto alla gestione della rete di stazioni per il rifornimento di carburante presenti sul territorio della regione Friuli Venezia Giulia.
Ogni stazione di rifornimento sia identificata univocamente da un codice e sia caratterizzata dall'azienda che la possiede, da una coppia di coordinate geografiche, che identificano la sua posizione, e dal comune di appartenza. Ogni stazione offra diversi tipi di carburante (benzina, gasolio, gas, ..) e disponga di un certo numero di pompe per l'erogazione del carburante. Si assuma che non necessariamente ogni stazione offra tutti i tipi possibili di carburante. Si vuole tener traccia delle (poche) stazioni che erogano gas.
Per ogni tipo di carburante disponibile presso una data stazione, si memorizzi la capacità massima del relativo serbatoio e la quantità correntemente disponibile. All'interno di una determinata stazione, ogni pompa sia caratterizzata da un numero (pompa numero 1, pompa numero 2, ..) e sia caratterizzata dal tipo di carburante erogato.
Ogni azienda sia identificata univocamente da un codice. Di ogni azienda vogliamo memorizzare il responsabile per la regione Friuli Venezia Giulia e il comune ove si trova l'ufficio regionale di riferimento.
Si assuma che un'azienda possa possedere più stazioni di rifornimento e che ogni stazione di rifornimento appartenga ad un'unica azienda.
Ogni azienda disponga di un certo numero di persone che prestano servizio presso le stazioni di rifornimento di sua proprietà. Ogni dipendente di un'azienda sia identificato univocamente dal suo codice fiscale e sia caratterizzato da nome e cognome, residenza e recapito telefonico. Si assuma che un dipendente possa lavorare presso più stazioni di rifornimento dell'azienda. Di ogni dipendente che lavora presso più stazioni, vogliamo memorizzare il piano di lavoro settimanale (si assuma che un dipendente non possa spostarsi da una stazione all'altra in uno stesso giorno della settimana, ma possa lavorare presso stazioni diverse in giorni diversi).

1.1 Fasi del progetto

Il progetto per la realizzazione della base di dati dovrà consistere nelle seguenti fasi:

- raccolta e analisi dei requisiti,
- progettazione concettuale,
- progettazione logica,
- progettazione fisica,
- implementazione,
- analisi dei dati in R.

2 Raccolta e analisi dei requisiti

La prima fase del progetto consiste nella raccolta e analisi dei requisiti sulla base della richiesta di progetto.

2.1 Raccolta dei requisiti

2.1.1 Glossario dei termini

Termine	Descrizione	Sinonimi	Collegamenti
Stazione di rifornimento	Stazione di rifornimento carburante presente nel territorio del FVG.	Stazione	Azienda Comune Piano di lavoro Serbatoio
Azienda	Possiede una o più stazioni nel territorio del FVG.		Dipendente Responsabile regionale Comune Ufficio regionale Stazione
Tipo di carburante	Tipo di carburante (metano, GPL, benzina e gasolio) contenuto in un serbatoio.		Serbatoio
Serbatoio	Un serbatoio contiene un tipo di carburante in una stazione di rifornimento.		Pompa Stazione Tipo carburante
Pompa	Eroga il carburante di un serbatoio in una stazione di rifornimento.		Serbatoio
Dipendente	E' assunto da un'azienda e lavora per essa.	Persona che presta servizio	Azienda Responsabile regionale Piano di lavoro
Piano di lavoro	Serve per tenere traccia delle mansioni dei dipendenti di un'azienda presso le loro stazioni di rifornimento.		Dipendente Stazione
Comune	Comune presente nel territorio del Friuli Venezia-Giulia.		Azienda Stazione
Responsabile regionale	Responsabile regionale del FVG di un'azienda.		Azienda Dipendente
Ufficio regionale	Un'azienda fa riferimento a uno degli uffici regionali del FVG. E' collocato in un determinato comune.		Azienda Comune

2.1.2 Frasi

Le frasi individuate che definiscono le entità principali della base di dati sono:

- **stazione di rifornimento** sia identificata univocamente da un codice e sia caratterizzata dall'azienda che la possiede, da una coppia di coordinate geografiche, che identificano la sua posizione, e dal comune di appartenenza.
- per ogni tipo di carburante disponibile presso una data stazione, che dispone di un certo numero di **pompe** per l'erogazione di esso, si memorizzi la capacità massima del relativo **serbataio** e la quantità correntemente disponibile.
- **azienda** sia identificata univocamente da un codice. Di ogni azienda vogliamo memorizzare il responsabile per la regione Friuli Venezia Giulia e il comune ove si trova l'ufficio regionale di riferimento.
- **dipendente** sia identificato univocamente dal suo codice fiscale e sia caratterizzato da nome e cognome, residenza e recapito telefonico.
- di ogni dipendente vogliamo memorizzare il **piano di lavoro** settimanale.

2.1.3 Regole relazionali

Le regole relazionali che sono state individuate durante la raccolta dei requisiti sono:

- ogni stazione non necessariamente possiede tutti i tipi di carburante;
- un'azienda può possedere più stazioni di rifornimento ma una stazione di rifornimento è posseduta da una sola azienda;

- ogni azienda dispone di un certo numero di persone che prestano servizio presso le stazioni di rifornimento di sua proprietà;
- un dipendente di una azienda può lavorare presso più stazioni di rifornimento dell'azienda però nello stesso giorno della settimana non può spostarsi da una stazione all'altra.

2.1.4 Operazioni principali

Le operazioni principali nella base di dati che sono state individuate durante la raccolta dei requisiti sono:

- tenere traccia delle (poche) stazioni che erogano gas;
- aggiornamento della quantità correntemente disponibile di carburante in un serbatoio;
- memorizzazione del responsabile regionale e del comune ove si trova l'ufficio regionale di riferimento di un'azienda;
- memorizzazione del piano di lavoro settimanale di un dipendente.

2.2 Analisi dei requisiti

2.2.1 Caratteristiche delle singole entità

Una **stazione di rifornimento**

- è identificata univocamente da un *codice*;
- è caratterizzata da una coppia di *coordinate* geografiche;
- è posseduta da un'*azienda*;
- è situata in un *comune*.

Un'**azienda**

- è identificata univocamente da un *codice*;
- è gestita da *responsabile regionale*;
- fa riferimento a un *ufficio regionale*, del quale si vuole memorizzare il comune ove è situato.

Un **serbatoio**

- contiene un *tipo di carburante*;
- è installato in una *stazione di rifornimento*;
- presenta una *capacità massima* e una *quantità residua* di carburante.

Una **pompa**

- eroga un *tipo di carburante* intingendo da un *serbatoio*;
- è identificata da un *numero*.

Un **dipendente**

- è assunto da un' *azienda*;
- è identificato univocamente dal proprio *codice fiscale*;
- si tiene traccia del suo *nome*, *cognome*, *residenza* e *recapito telefonico*.

2.2.2 Regole relazionali, vincoli di integrità

Attraverso i dati raccolti dal testo della richiesta e con delle ricerche sul web sono stati individuati i seguenti requisiti per una corretta definizione delle funzionalità e vincoli della base di dati:

- nella definizione del suo piano di lavoro settimanale un dipendente di un'azienda durante la giornata deve lavorare in un'unica stazione di rifornimento dove esegue diverse mansioni;
- una stazione dispone di diverse pompe per l'erogazione dei tipi di carburante (non per forza li possiede tutti);
- una pompa eroga un solo tipo di carburante attingendo dal serbatoio che lo contiene;

- possono esistere più serbatoi nella stessa stazione di rifornimento per lo stesso tipo di carburante;
- ogni azienda ha un unico responsabile per la regione del Friuli Venezia Giulia e fa riferimento a un unico ufficio regionale situato in un determinato comune;
- nella regione Friuli Venezia Giulia non esistono comuni con lo stesso nome;
- il responsabile regionale di un'azienda è un dipendente specializzato;
- un dipendente lavora solo per le stazioni di rifornimento possedute dall'azienda che lo ha assunto.

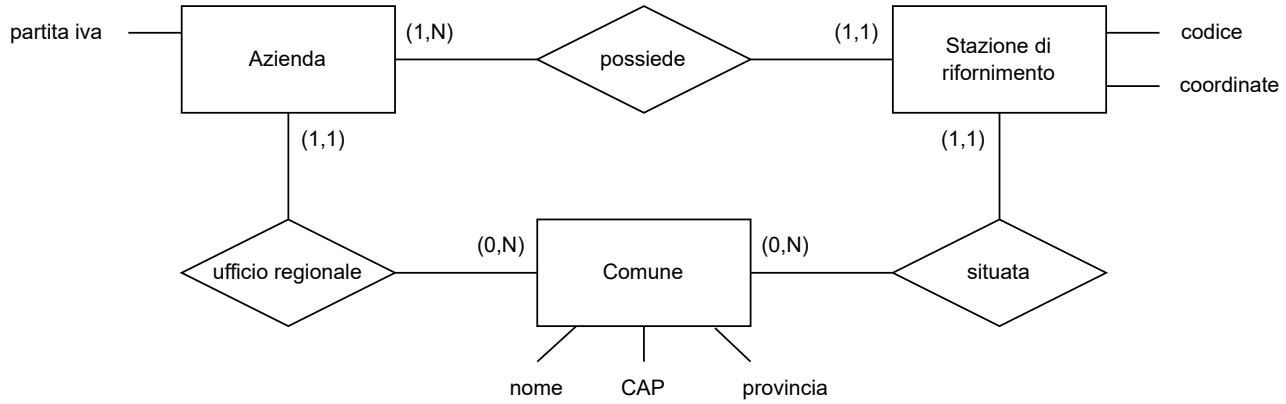
3 Progettazione concettuale

La progettazione concettuale mira a trasformare le informazioni ottenute durante la raccolta e analisi dei requisiti da un linguaggio naturale a una forma astratta in modo da focalizzarsi sul significato dei dati. Così facendo, con il modello concettuale Entità - Relazione, vengono individuate le entità e le relazioni presenti nella base di dati, con le loro proprietà.

3.1 Definizione singole entità e relazioni fra esse

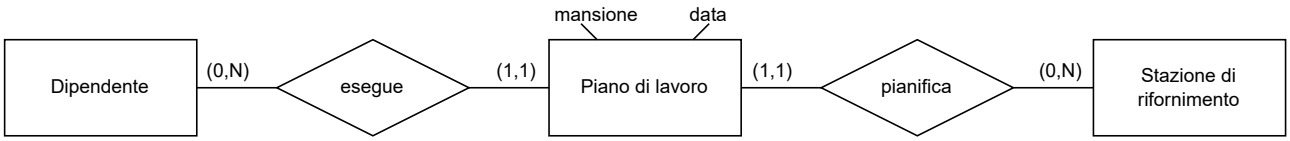
3.1.1 Azienda, stazione di rifornimento e comune

Un'azienda possiede una o più stazioni di rifornimento e una stazione di rifornimento è posseduta da una sola azienda. Un comune può non avere stazioni di rifornimento e può non avere un ufficio regionale a cui fa riferimento un'azienda.



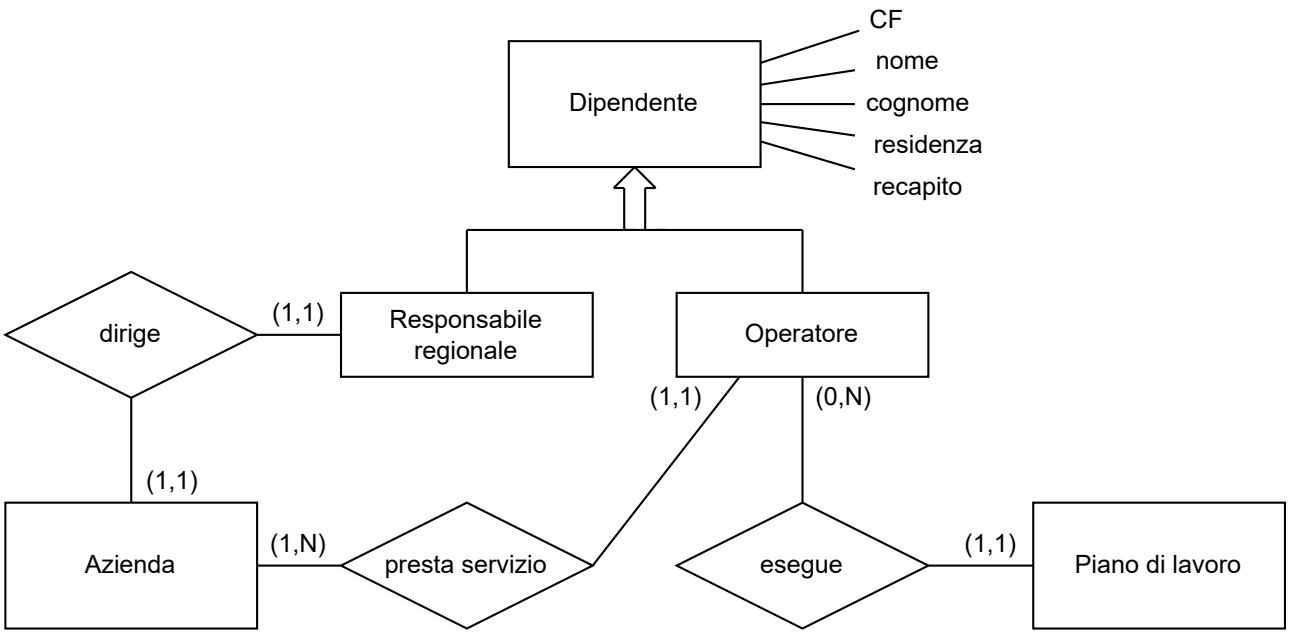
3.1.2 Piano di lavoro

Un dipendente può non aver mai eseguito alcuna mansione presso una stazione di rifornimento e quest'ultima può non aver ancora pianificato alcuna mansione.



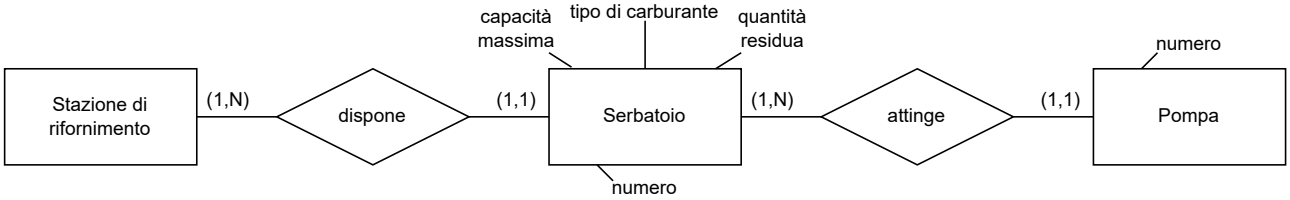
3.1.3 Generalizzazione di Dipendente

Un dipendente può essere distinto o come un responsabile regionale di un'azienda o come un operatore il quale può eseguire delle mansioni presso le stazioni di rifornimento dell'azienda per cui presta servizio.

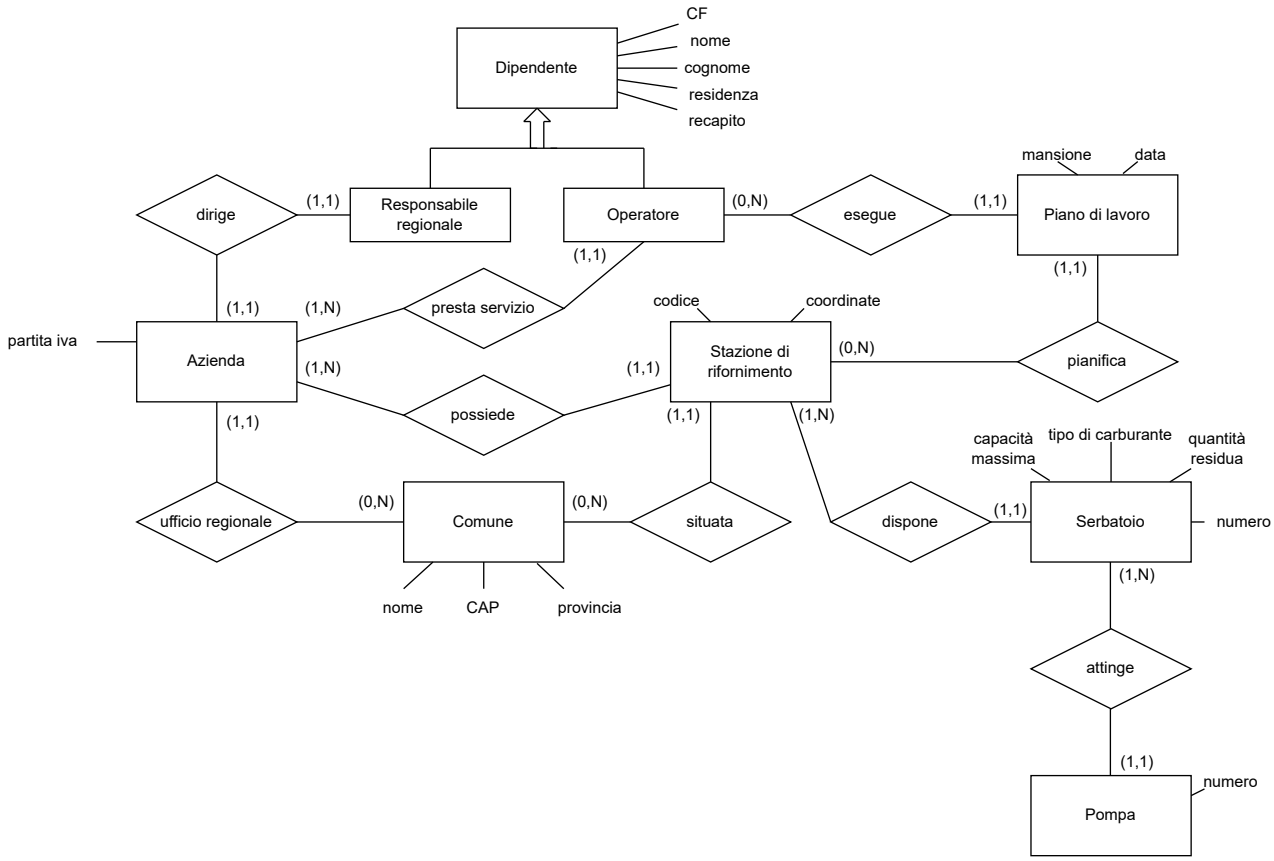


3.1.4 Serbatoio, pompa e tipo di carburante

Una stazione di rifornimento possiede uno o più serbatoi, da ognuno dei quali possono attingere una o più pompe.



3.2 Schema E-R


Dipendente
CF
nome
cognome
residenza
recapito
Responsabile regionale
Operatore
dirige
(1,1)
(0,N)
esegue
(1,1)
Piano di lavoro
mansione
data
(1,1)
(1,1)
(1,N)
presta servizio
(1,1)
partita iva
Azienda
(1,N)
possiede
(1,1)
codice
coordinate
Stazione di rifornimento
(0,N)
pianifica
(1,1)
(1,1)
(1,N)
capacità massima
tipo di carburante
quantità residua
ufficio regionale
(0,N)
Comune
(0,N)
situata
dispone
(1,1)
Serbatoio
numero
nome
CAP
provincia
(1,N)
attinge
(1,1)
Pompa
numero

4 Progettazione logica

Prima di costruire lo schema logico deve essere effettuata un'analisi sullo schema E-R ottenuto tenendo presenti fattori come il carico applicativo che la nostra base di dati avrà sul DBMS. Tale analisi porterà ad una eventuale ristrutturazione dello schema concettuale (E-R) ottenuto inizialmente, ottimizzandolo. Le fasi che portano a una ristrutturazione dello schema E-R e alla creazione dello schema logico sono:

- l'analisi delle ridondanze;
- l'eliminazione delle generalizzazioni;
- il partizionamento/accoppiamento delle entità e relazioni;
- la scelta delle chiavi primarie.

4.1 Analisi delle ridondanze

L'analisi delle ridondanze viene effettuata per verificare se le ridondanze emerse nella fase di progettazione concettuale debbano essere mantenute o eliminate ristrutturando lo schema concettuale iniziale.
Le ridondanze possono risultare utili nel caso di operazioni frequenti alla base di dati dove la presenza di ridondanza risulta essere meno onerosa rispetto al dover derivare degli attributi attraverso le relazioni fra tabelle. Un'operazione alla nostra basi di dati dove può risultare utile la presenza di una ridondanza risulta essere la visualizzazione delle **stazioni di rifornimento che erogano gas**, con una frequenza stimata di circa 10 richieste giornaliere. Per semplificare questa interrogazione è stato inserito nello schema E-R l'attributo "gas" nell'entità "stazioni di rifornimento". Tale attributo indica se la stazione di rifornimento ha almeno una pompa che eroghi gas ed è una ridondanza perché tale informazione potrebbe essere derivata tramite la relazione "dispone".
L'analisi delle ridondanze viene effettuata tramite lo studio di quattro concetti:

- la tabella dei volumi,
- la tabella delle frequenze,
- la tabella degli accessi,
- i costi di memoria.

Nella **tabella dei volumi** per ogni concetto dello schema E-R ne viene stimato il volume ed è rappresentata in seguito:

Concetto	Ruolo	Volume	Formula di ottenimento
Stazione di rifornimento	E	492	https://www.fabiodisconzi.com/open-carburanti/
Serbatoio	E	1968	4 serbatoi per stazione in media (uno per ogni tipo di carburante) * n. stazioni di rifornimento
Pompa	E	5904	Serbatoi * 3 postazioni in media per ogni stazione
Azienda	E	10	Numero di aziende nella provincia
Dipendente	E	1486	Responsabili regionali + Operatori
Comune	E	215	Numero di comuni del Friuli Venezia-Giulia (fonte: Wikipedia)
Operatore	E	1476	Stazione * (1 amministratore + 2 subordinati)
Responsabile regionale	E	10	Equivalente al numero di aziende
Piano di lavoro	E	7.407.158.400	Operatori * stazioni * n medio masioni giornaliere (2) * 255 (giorni lavorativi) * anni di vita del database stimati (20)
esegue	R	7.527.600	Operatori * n medio di mansioni giornaliere (2) * 255 giorni lavorativi * anni di servizio (in media 10)
pianifica	R	15.055.200	Stazioni * n medio di mansioni giornaliere (2) * 3 operatori * 255 giorni lavorativi * 20 anni di vita di una stazione in media
ufficio regionale	R	10	Equivalente al numero di aziende
situata	R	492	Equivalente al numero di stazioni di rifornimento
attinge	R	5904	Equivalente al numero di pompe
dispone	R	1968	Equivalente al numero di serbatoi
dirige	R	10	Equivalente al numero di responsabili regionali

Come **costi di memoria**, la ridondanza data da un attributo "gas" nella tabella "Stazioni di rifornimento", rappresentato da un booleano, creerebbe un' occupazione di memoria aggiuntiva di circa

$$1bit * 492 = 492bit \simeq 62byte.$$

Le **operazioni che coinvolgono la ridondanza** sono le seguenti:

- Inserimento di una pompa (che eroga gas);
- Cancellazione di una pompa (che eroga gas);
- Ricerca delle stazioni che erogano gas: questa operazione è il motivo principale per cui si è deciso di inserire la ridondanza. Senza la ridondanza per eseguire questa operazione si dovrebbero interrogare due tabelle, "Stazione di rifornimento" relazionata a "Serbatoio", mentre con la ridondanza basterebbe interrogare solo la prima.

La modifica di una pompa viene effettuata con una cancellazione seguita da un inserimento.

La **tabella delle frequenze** di tali operazioni risulta essere

Operazione	**Tipo**	**Frequenza media**
Inserimento serbatoio con gas	I	2/anno
Eliminazione serbatoio con gas	I	0.5/anno
Ricerca stazioni che erogano gas	I	10/giorno

In seguito vengono rappresentate le **tabelle degli accessi** per ogni operazione, con e senza la presenza della ridondanza.

Operazione 1: Inserimento di un serbatoio con gas

Concetto	**Tipo di operazione**	**Numero di accessi**
Con ridondanza		
Serbatoio	L	1
Serbatoio	S	1
Stazione di rifornimento	L	2
Stazione di rifornimento	S	1
Senza ridondanza		
Serbatoio	L	1
Serbatoio	S	1
Stazione di rifornimento	L	1

Quando si aggiunge un serbatoio di gas con la presenza di ridondanza vengono effettuate in più un'operazione di lettura e una di scrittura in stazione di rifornimento, per controllare se il flag "gas" è già impostato a "true" e in caso scriverlo tale.

Operazione 2: Cancellazione di un serbatoio con gas

Concetto	Tipo di operazione	Numero di accessi
Con ridondanza		
Serbatoio	L	1
Serbatoio	S	1
dispone	L	4
Stazione di rifornimento	S	1
Senza ridondanza		
Serbatoio	L	1
Serbatoio	S	1

Quando viene eliminato un serbatoio di gas, con la presenza di ridondanza, prima di settare il flag a "false" bisogna cercare se la stazione di rifornimento ha altri serbatoi di gas, quindi vengono effettuate 4 letture aggiuntive (serbatoi presenti in media in una stazione di rifornimento) e una scrittura in più nel caso di esito negativo della ricerca.

Operazione 3: Ricerca delle stazioni che erogano gas

Concetto	Tipo di operazione	Numero di accessi
Con ridondanza		
Stazione di rifornimento	L	492
Senza ridondanza		
Stazione di rifornimento	L	492
Serbatoio	L	1968

La ricerca delle stazioni che erogano gas è una delle interrogazioni più frequenti nel database e viene fatta in media 10 volte al giorno. La presenza della ridondanza permette di effettuare 1968 letture in meno, ovvero controlla solo il flag "gas" nella tabella "Stazione di rifornimento" ed evita di accedere a "Serbatoio" circa 492*4=1968 volte per controllare tutti i serbatoi delle stazioni di rifornimento presenti nella regione.

Analisi conclusiva sulla presenza di ridondanza
Per decidere se mantenere la ridondanza o meno si deve tenere in considerazione che le scritture sono più onerose delle letture, dato che bloccano l'accesso a tale risorsa e rallentano il sistema.

Concludendo, il totale di letture e scritture che otteniamo con la presenza di ridondanza o meno sono:

Con ridondanza		
Operazione	**Letture**	**Scritture**
1	3	2
2	5	2
3	492	0
	500	**4**

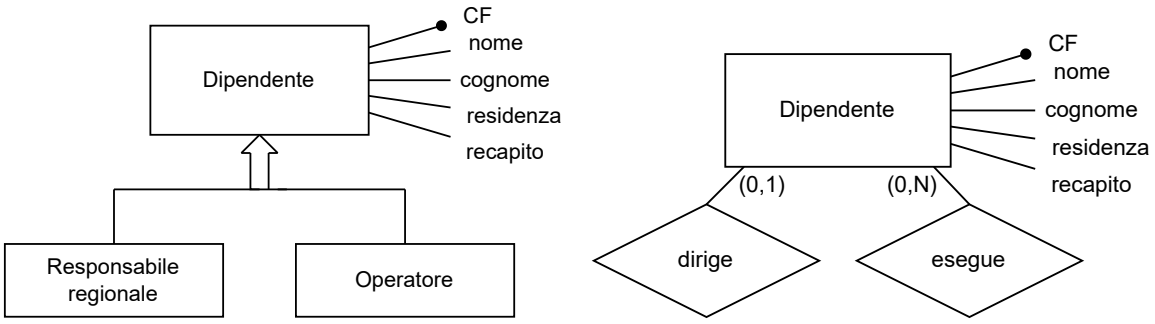
Senza ridondanza		
Operazione	**Letture**	**Scritture**
1	2	1
2	1	1
3	2460	0
	2463	**2**

Con la presenza di ridondanza si hanno 4 scritture al posto di 2 ma circa 1/5 di letture in meno. La terza operazione viene eseguita molto più frequentemente rispetto alle altre due e non

coinvolge scritture. La presenza della ridondanza ridurrebbe le letture della terza operazione di circa 1/5. Dato che la ricerca di stazioni che erogano gas viene effettuata più volte nella giornata e i costi di memoria aggiuntivi per l'attributo "gas" sono di soli 62byte (trascurabili per i sistemi moderni) è stato deciso di mantenere la ridondanza.

4.2 Eliminazione delle generalizzazioni

Lo schema E-R presenta solo una generalizzazione, cioè l'entità padre Dipendente con le entità figlie Responsabile regionale e Operatore. Dato che "Responsabile regionale" e "Operatore" hanno tutti gli attributi in comune, cioè quelli specificati nell'entità padre "Dipendente", verrà mantenuta solo l'entità padre e verranno distinti tramite la loro relazione con l'entità "Azienda".



N.B. Un dipendente può essere responsabile regionale di una sola azienda e in tal caso non esegue alcuna mansione presso le stazioni di rifornimento dell'azienda che dirige.

4.3 Scelta delle chiavi primarie

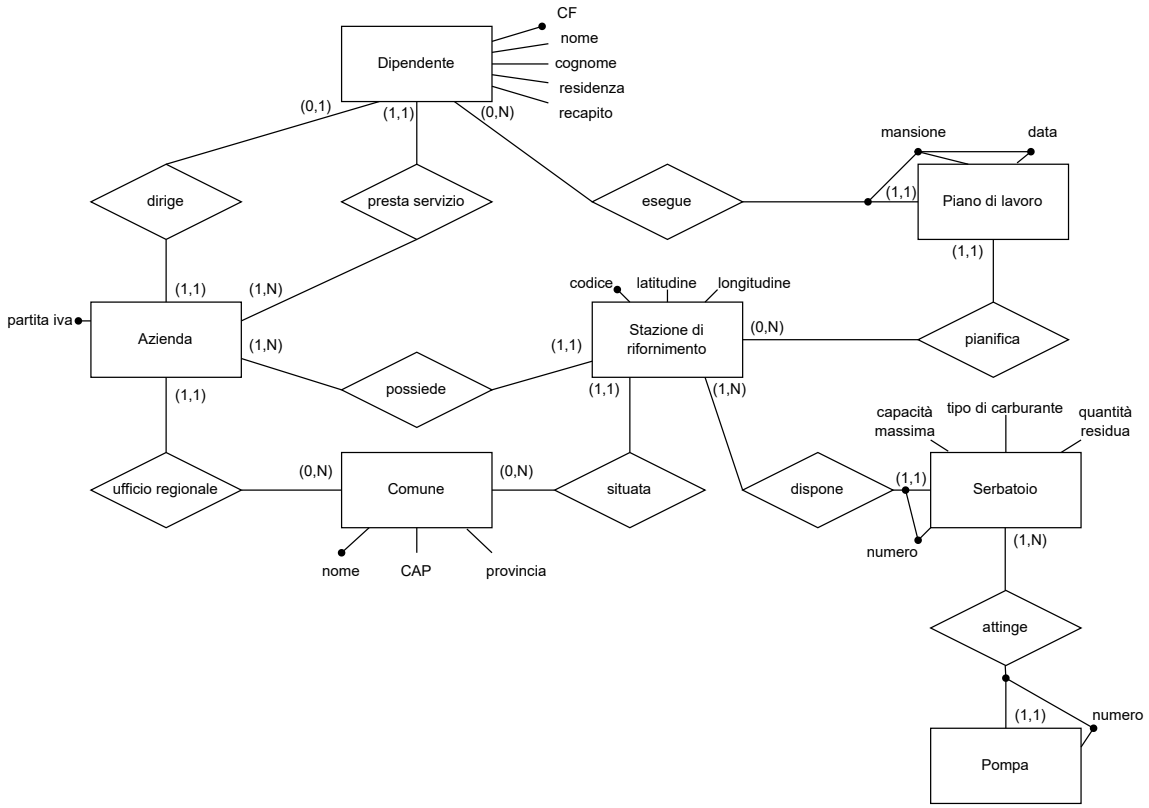
Fra le diverse chiavi candidate di ogni entità si è scelto come chiave primaria quella con il minor numero di attributi coinvolti, ovvero:

- per **Comune** l'attributo "nome", dato che nel territorio del FVG non esistono due comuni con lo stesso nome;
- per **Dipendente** l'attributo "codice fiscale", come definito della richiesta;
- per **Azienda** l'attributo "partita iva", dato che è un identificativo unico usato per le attività commerciali;
- per **Stazione di rifornimento** l'attributo "codice", come definito nella richiesta;
- per **Piano di lavoro** l'accoppiamento degli attributi "dipendente", "mansione" e "giorno", dato che un dipendente può lavorare in una sola stazione di rifornimento nella stessa giornata, eseguendo diverse mansioni;
- per **Serbatoio** la coppia di attributi "stazione" e "numero", dato che il numeramento dei serbatoi è indipendente per ogni stazione di rifornimento;
- per **Pompa** la chiave primaria è stata costruita unendo la chiave primaria di Serbatoio, cioè "stazione" e "numero" del serbatoio, con il numero della pompa collegata ad esso (tale numeramento è indipendente dagli altri serbatoi della stazione di rifornimento).

4.4 Disaccoppiamento di attributi composti

L'entità "Stazione di rifornimento" presenta l'attributo composto "coordinate" che viene suddiviso in due attributi non composti: latitudine e longitudine.

4.5 Schema E-R ristrutturato



4.6 Traduzione da schema E-R a logico

4.6.1 Relazione "dirige"

La relazione "dirige" indica il responsabile regionale di un'azienda, il quale è un dipendente. Tale relazione verrà indicata come un campo "responsabile regionale" nella tabella Azienda che punterà a un elemento della tabella Dipendente.

4.6.2 Relazione "ufficio regionale"

La relazione "ufficio regionale" verrà indicata come un campo della tabella Azienda che punterà a un elemento della tabella Comune.

4.7 Schema logico

COMUNE(**nome**, provincia, CAP)

DIPENDENTE(**codice fiscale**, nome, cognome, comune, indirizzo, telefono, data assunzione, azienda)
DIPENDENTE(comune) $\longrightarrow$ COMUNE(nome)
DIPENDENTE(azienda) $\longrightarrow$ AZIENDA(partita iva)

AZIENDA(**partita iva**, nome, ufficio regionale, responsabile regionale, data inizio incarico)
AZIENDA(responsabile regionale) $\longrightarrow$ DIPENDENTE(codice fiscale)
AZIENDA(ufficio regionale) $\longrightarrow$ COMUNE(nome)

STAZIONE DI RIFORNIMENTO(**codice**, latitudine, longitudine, gas, comune, azienda)
UNIQUE: (latitudine, longitudine)
STAZIONE DI RIFORNIMENTO(comune) $\longrightarrow$ COMUNE(nome)
STAZIONE DI RIFORNIMENTO(azienda) $\longrightarrow$ AZIENDA(partita iva)

SERBATOIO(**stazione**, **numero**, tipo carburante, capacità massima, quantità residua)
SERBATOIO(stazione) $\longrightarrow$ STAZIONE DI RIFORNIMENTO(codice)

POMPA(**stazione**,**numero serbatoio**, **numero pompa**)
POMPA(numero serbatoio) $\longrightarrow$ SERBATOIO(numero)
POMPA(stazione) $\longrightarrow$ STAZIONE DI RIFORNIMENTO(codice)

PIANO DI LAVORO(**dipendente**,**mansione**, **giorno**, stazione)
PIANO DI LAVORO(dipendente) $\longrightarrow$ DIPENDENTE(codice fiscale)
PIANO DI LAVORO(stazione) $\longrightarrow$ STAZIONE DI RIFORNIMENTO(codice)

N.B. Gli attributi in grassetto indicano i componenti della **chiave primaria**, mentre quelli in rosso le chiavi esterne.

5 Progettazione fisica

Per la progettazione fisica del database è stato utilizzato PostgreSQL con il supporto grafico di pgAdmin 4.

La progettazione fisica ha incluso la definizione di

1. domini;
2. tabelle;
3. vincoli delle tabelle (chiavi e controlli);
4. indici.

5.1 Domini

1) Tipo carburante
Dato che i tipi di carburante sono solo quattro (GPL, metano, benzina e gasolio) è stato deciso di definirli tramite un dominio, anziché come una tabella.

```
1 CREATE DOMAIN public.tipo_carburante
2     AS character varying(10);
3
4 ALTER DOMAIN public.tipo_carburante
5     ADD CONSTRAINT carburante_valido CHECK (VALUE::text = ANY (ARRAY
      ['gpl'::character varying::text, 'benzina'::character varying::
      text, 'gasolio'::character varying::text, 'metano'::character
      varying::text])));
```

2) CAP
Definisce il formato dell'attributo CAP (codice di avviamento postale) in un comune.

```
1 CREATE DOMAIN public.cap
2     AS character(5);
3
4 ALTER DOMAIN public.cap
5     ADD CONSTRAINT cap_check CHECK (VALUE ~ '^\d{5}$'::text);
```

3) CF
Definisce il formato dell'attributo CF (codice fiscale) di un dipendente.

```
1 CREATE DOMAIN public.cf
2     AS character(16);
3
4 ALTER DOMAIN public.cf
5     ADD CONSTRAINT cf_check CHECK (VALUE ~ '^[a-zA-Z]{6}[0-9]{2}[a-
      zA-Z][0-9]{2}[a-zA-Z][0-9]{3}[a-zA-Z]$'::text);
```

4) PIVA
Definisce il formato dell'attributo PIVA (partita iva) di un'azienda.

```
1 CREATE DOMAIN public.piva
2     AS character(11);
3
4 ALTER DOMAIN public.piva
5     ADD CONSTRAINT piva_check CHECK (VALUE ~ '^\d{11}$'::text);
```

5.2 Vincoli

Analizziamo, come esempio, la tabella **stazione di rifornimento**.

```
CREATE TABLE IF NOT EXISTS public.stazione_di_rifornimento
(
    codice smallint NOT NULL DEFAULT nextval('
    stazione_di_rifornimento_codice_seq'::regclass),
    gas boolean NOT NULL,
    comune character varying(50) COLLATE pg_catalog."default",
    azienda piva COLLATE pg_catalog."default",
    latitudine double precision,
    longitudine double precision,
    CONSTRAINT stazione_di_rifornimento_pkey PRIMARY KEY (codice),
    CONSTRAINT coordinate UNIQUE (latitudine, longitudine),
    CONSTRAINT stazione_di_rifornimento_azienda_fkey FOREIGN KEY (
    azienda)
        REFERENCES public.azienda (partita_iva) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT stazione_di_rifornimento_comune_fkey FOREIGN KEY (
    comune)
        REFERENCES public.comune (nome) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

La **chiave primaria** è data dall'attributo *codice* che è univoco per ogni tupla. Però anche la coppia delle coordinate *latitudine* e *longitudine* sono univoche per ogni tupla, quindi si è aggiunto il vincolo UNIQUE ad esse.
Le **chiavi esterne** presenti nella tabella sono *azienda* e *comune*. Le istruzioni ON UPDATE CASCADE e ON DELETE CASCADE indica che, quando il riferimento nella tabella genitore verrà modificato o cancellato, nella tabella figlio verrà modificato o cancellato in conseguenza.

5.2.1 Controllo su "quantità residua" in Serbatoio

Nella tabella **serbatoio** è stato aggiunto un controllo per la quantità residua presente nel serbatoio.

```
CONSTRAINT ck_quantita CHECK (quantita_residua >= 0::double
    precision AND quantita_residua <= capacita_massima::double
    precision)
```

5.3 Indici

Al fine di ottimizzare le operazioni al database eseguite con più frequenza si è deciso di implementare i seguenti indici:

1. stazione_di_rifornimento.comune e stazione_di_rifornimento.azienda;
2. piano_di_lavoro.stazione e piano_di_lavoro.data;
3. dipendente.lavora_in, dipendente.nome e dipendente.cognome.

Viene utilizzato il comando EXPLAIN ANALYZE per ottenere il tempo di esecuzione delle query con e senza indici.

5.3.1 Indice su stazione_di_rifornimento.comune e stazione_di_rifornimento.azienda

Data la query e l'indice

```
1 EXPLAIN ANALYSE
2 SELECT codice,
3     latitudine,
4     longitudine
5 FROM stazione_di_rifornimento
6 WHERE comune = 'Udine'
7     AND azienda = 'Agip Eni';
8
9 CREATE INDEX aziende_comune_index ON stazione_di_rifornimento(comune
    , azienda);
```

i tempi di esecuzione risultano essere

- Senza indice: 0.091 ms
- Con indice: 0.038 ms

I tempi di esecuzione della query con l'uso dell'indice si dimezzano circa.

5.3.2 Indice su piano_di_lavoro.stazione e piano_di_lavoro.data

Data la query e l'indice

```
1 EXPLAIN ANALYSE
2 SELECT *
3 from piano_di_lavoro
4 WHERE data = '2022-10-16'
5     and stazione = 354;
6
7 CREATE INDEX piano_di_lavoro_dipendenti_stazione_index ON
    piano_di_lavoro(data, stazione);
```

i tempi di esecuzione con e senza indice risultano essere

- Senza indice: 0.642 ms
- Con indice: 0.064 ms

I tempi di esecuzione della query con l'uso dell'indice si riducono a un decimo rispetto che senza.

5.3.3 Indice su dipendente.lavora_in, dipendente.nome e dipendente.cognome

Data la query e l'indice

```
1 EXPLAIN ANALYSE
2 SELECT *
3 FROM dipendente
4 WHERE lavora_in = '08254089087'
5 ORDER BY nome,
6     cognome;
7
8 CREATE INDEX dipendenti_index ON dipendente(lavora_in, nome, cognome
    );
```

i tempi di esecuzione con e senza indice risultano essere

- Senza indice: 0.642 ms
- Con indice: 0.539 ms

I tempi di esecuzione della query con l'uso dell'indice si riducono di circa 0.1 ms.

6 Implementazione

L'implementazione del database ha incluso la definizione di trigger, query e il popolamento delle tabelle del database.

6.1 Trigger

6.1.1 Trigger check_gas_flag_insertion

Il seguente trigger controlla ogni qual volta viene inserito un nuovo serbatoio che contiene gas (gpl o metano) se la stazione a cui appartiene possiede il flag "gas" settato su TRUE. In caso contrario lo aggiorna.

```
1 CREATE FUNCTION public.check_gas_flag_insertion() RETURNS trigger
2     LANGUAGE plpgsql
3     AS $$
4 DECLARE n INTEGER;
5
6 BEGIN
7 SELECT COUNT(*) into n
8 FROM serbatoio S
9 WHERE S.stazione = new.stazione
10     and (
11         S.carburante = 'metano'
12         or S.carburante = 'gpl'
13     );
14
15     if n > 0 THEN
16     UPDATE stazione_di_rifornimento S
17     set gas = true
18     where S.codice = new.stazione;
19
20     end if;
21
22     return new;
23
24 end;
25
26 $$;
27
28 CREATE TRIGGER check_gas_flag_insertion_trigger AFTER INSERT OR
    UPDATE ON public.serbatoio FOR EACH ROW EXECUTE FUNCTION public.
    check_gas_flag_insertion();
```

6.1.2 Trigger check_gas_flag_deletion

Il seguente trigger controlla ogni qual volta viene cancellato un nuovo serbatoio che contiene gas (gpl o metano) se la stazione a cui appartiene possiede il flag "gas" settato su FALSE. In caso contrario lo aggiorna.

```
1 CREATE FUNCTION check_gas_flag_deletion() RETURNS trigger LANGUAGE
    plpgsql AS $$
2 DECLARE n INTEGER;
3 BEGIN
```

```
 4 SELECT COUNT(*) into n
 5 FROM serbatoio S
 6 WHERE S.stazione = old .stazione
 7     and (
 8         S.carburante = 'metano'
 9         or S.carburante = 'gpl'
10     );
11
12     if n = 0 THEN
13     UPDATE stazione_di_rifornimento S
14     set gas = FALSE
15     where S.codice = old .stazione;
16
17     end if;
18
19     return new;
20
21 end;
22
23 $$;
24
25 CREATE TRIGGER check_gas_flag_deletion_trigger AFTER DELETE ON
       public.serbatoio FOR EACH ROW EXECUTE FUNCTION public.
       check_gas_flag_deletion();
```

6.1.3 Trigger check_valid_data

Il seguente trigger controlla che non vengano inseriti piani di lavoro con data precedente alla data di inserimento del record (data odierna).

```
1 CREATE FUNCTION public.check_valid_data() RETURNS trigger LANGUAGE
      plpgsql AS $$
2 BEGIN
3     IF CURRENT_DATE <= new .data
4     THEN return new;
5     ELSE raise exception 'Non si puo' creare piani di lavoro
      precedenti alla data odierna';
6     return null;
7     end if;
8
9 end;
10
11 $$;
12
13 CREATE TRIGGER check_valid_data_trigger BEFORE INSERT ON public.
       piano_di_lavoro FOR EACH ROW EXECUTE FUNCTION public.
       check_valid_data();
```

6.1.4 Trigger check_valid_stazione_di_lavoro

Il seguente trigger controlla che ad ogni dipendente venga assegnata la giornata di lavoro solo in una stazione che appartiene all'azienda da cui è stato assunto.

```
1 CREATE FUNCTION public.check_valid_stazione_di_lavoro() RETURNS
      trigger LANGUAGE plpgsql AS $$
2     DECLARE AZIENDA_STAZ piva;
3     DECLARE AZIENDA_DIP piva;
4 BEGIN
5     SELECT azienda into AZIENDA_STAZ
6     FROM stazione_di_rifornimento S
7     WHERE S.codice = new.stazione;
8
9     SELECT lavora_in into AZIENDA_DIP
10    FROM dipendente D
11    WHERE D.codice_fiscale = new.dipendente;
12
13    IF AZIENDA_STAZ = AZIENDA_DIP
14    THEN return new;
15    ELSE raise exception 'Azienda dipendete e azienda stazione sono
      diverse';
16
17    return null;
18
19    end if;
20
21 end;
22
23 $$;
24
25 CREATE TRIGGER check_valid_stazione_di_lavoro_trigger BEFORE INSERT
      ON public.piano_di_lavoro FOR EACH ROW EXECUTE FUNCTION public.
      check_valid_stazione();
```

6.1.5 Trigger check_valid_responsabile

Il seguente trigger controlla che un dipendente può essere assegnato come responsabile regionale solo per la stessa azienda che lo ha assunto.

```
1 CREATE FUNCTION public.check_valid_responsabile() RETURNS trigger
      LANGUAGE plpgsql AS $$
2     DECLARE AZIENDA_RESP piva;
3 BEGIN
4     SELECT lavora_in into AZIENDA_RESP
5     FROM dipendente D
6     WHERE D.codice_fiscale = new.responsabile_regionale;
7
8     if (AZIENDA_RESP = new.partita_iva) OR (AZIENDA_RESP IS NULL)
9     THEN return new;
10
11    ELSE raise exception 'Un responsabile deve lavorare per l
      azienda';
12
13    return null;
14
15    end if;
16
```

```
17 end;
18
19 $$;
20
21 CREATE TRIGGER check_valid_responsabile_trigger BEFORE INSERT OR
       UPDATE ON public.azienda FOR EACH ROW EXECUTE FUNCTION public.
       check_valid_responsabile();
```

6.2 Popolamento della base di dati

Il popolamento della base di dati è stato effettuato mediante due script sviluppati in linguaggio Python.
Abbiamo scelto questo linguaggio perchè la sua semplicità e immediatezza ci ha permesso di creare in poco tempo i dati di cui avevemo bisogno, grazie all'ausilio di librerie ad hoc pronte all'uso.

6.2.1 Generazione dei dati

La prima operazione che abbiamo effettuato è stata quella di generare una buona quantità di dati per ognuna delle tabelle del database e salvarli in un file json per le elaborazioni successive.
I dati dei comuni sono stati ottenuti dal seguente file json "Dati dei comuni italiani" che contiene per ogni provincia del FVG la lista dei comuni con il relativo codice CAP.
I dati delle aziende sono stati ottenuti tramite delle ricerche su internet e inseriti manualmente all'interno dello script.
I dati dei dipendenti sono stati generati in modo casuale avvalendoci della libreria Faker di python che permette di generare dati anagrafici fittizi.
I dati delle restanti tabelle sono stati generati in modo casuale, combinando i dati generati in precedenza e verificando che i vincoli imposti dal database venissero rispettati.

6.2.2 Inserimento dei dati nel database

Dopo aver generato una discreta quantità di dati e aver salvato il tutto in un file json denominato "fake_data.json" abbiamo creato il secondo script con il compito specifico di inserire i dati all'interno della base di dati.
Lo script utilizza la libreria `psycopg2` nella quale è sviluppato il driver che permette di connettersi ai database PostgreSQL.
Per effettuare le query la libreria si avvale di un cursore che è un oggetto contiene tutte le utility necesserie per effettuare le transazioni nel database.
La funzione `cur.executemany()` esegue la query inserita come primo argomento per ognuno dei blocchi di dati contenuti nella lista passata come secondo argomento.
Al termine, tramite `cur.commit()`, i dati vengono salvati nella base di dati.

```
1 import psycopg2
2 import json
3
4 conn = psycopg2.connect(
5     host="localhost",
6     dbname="progetto_basi_di_dati",
7     user="postgres",
8     password="password",
9     port="5432")
10
11 cur = conn.cursor()
12
```

```
13 with open('fake_data.json', 'r') as f:
14     data = json.load(f)
15
16     cur.executemany(
17         """INSERT INTO comune(nome,cap,provincia) VALUES (%(nome)s,
       %(cap)s,%(provincia)s)""", data["comuni"])
18
19     cur.executemany(
20         """INSERT INTO dipendente(codice_fiscale,nome,cognome,comune
       ,indirizzo,telefono) VALUES (%(codice_fiscale)s,%(nome)s,%(
       cognome)s,%(comune)s,%(indirizzo)s,%(telefono)s)""", data["
       dipendenti"])
21
22     cur.executemany(
23         """INSERT INTO azienda(partita_iva,nome,
       responsabile_regionale,ufficio_regionale,data_inizio_incarico)
       VALUES (%(partita_iva)s,%(nome)s,%(responsabile_regionale)s,%(
       ufficio_regionale)s,%(data_inizio_incarico)s)""", data["aziende"
       ])
24
25     cur.executemany(
26         """UPDATE dipendente set data_assunzione = %(
       data_di_assunzione)s, lavora_in = %(lavora_in)s where
       codice_fiscale = %(codice_fiscale)s""", data["dipendenti"])
27
28     cur.executemany(
29         """INSERT INTO stazione_di_rifornimento(codice,gas,comune,
       azienda,longitudine,latitudine) VALUES (%(codice)s,%(gas)s,%(
       comune)s,%(azienda)s,%(longitudine)s,%(latitudine)s)""", data["
       stazioni_di_rifornimento"])
30
31     cur.executemany(
32         """INSERT INTO serbatoio(quantita_residua,capacita_massima,
       stazione,numero,carburante) VALUES (%(quantita_residua)s,%(
       capacita_massima)s,%(stazione)s,%(numero)s,%(carburante)s)""",
       data["serbatoi"])
33
34     cur.executemany(
35         """INSERT INTO pompa(numero,stazione_serbatoio,
       numero_serbatoio) VALUES (%(numero)s,%(stazione_serbatoio)s,%(
       numero_serbatoio)s)""", data["pompe"])
36
37     cur.executemany(
38         """INSERT INTO piano_di_lavoro(data,mansione,dipendente,
       stazione) VALUES (%(data)s,%(mansione)s,%(dipendente)s,%(stazione
       )s)""", data["piani_di_lavoro"])
39
40     conn.commit()
41
42 conn.close()
```

7 Analisi dei dati in R

7.1 Connessione al database

Per realizzare le interrogazioni sul database abbiamo utilizzato la libreria DBI che al suo interno implementa tutte le funzione di collegamento, sessione e di richiesta dati necessarie per realizzarle.
Il collegamento avviene tramite la funzione `dbConnect()` che come parametri richiede un driver in grado di interfacciarsi con i database PostgreSQL e tutte le credenziali necessarie per poterci accedere.

```
1 library(DBI)
2 library(dplyr)
3
4 db <- 'progetto_basi_di_dati'
5 host_db <- 'localhost''
6 db_port <- 5432
7 db_user <- 'postgres'
8 con <- dbConnect(RPostgres::Postgres(), dbname = db, host=host_db,
    port=db_port, user=db_user, password=rstudioapi::askForPassword("
    Database password"))
```

7.2 Risultati delle query

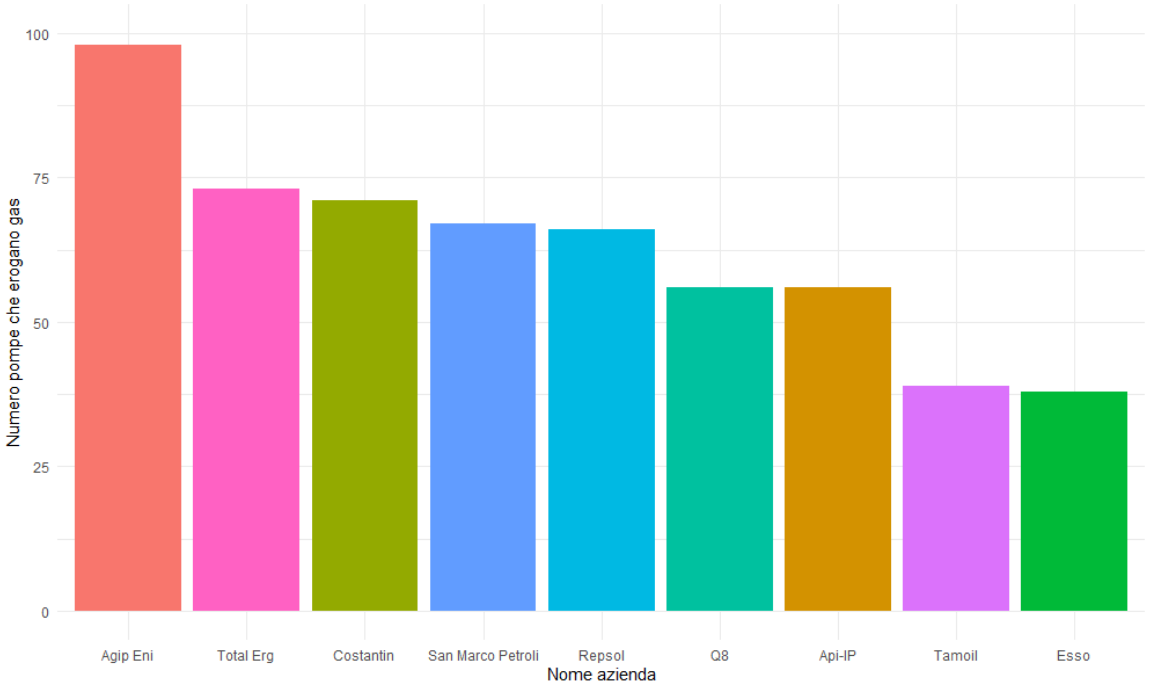
Le query sono state eseguite tramite la funzione `dbGetQuery()` implementata all'interno della libreria DBI e che richiede come primo argomento una sessione di connessione ottenuta con il metodo descritto nella sezione precedente e come secondo uno "statement" ovvero una stringa contenente un'interrogazione in linguaggio SQL.

7.2.1 Numero di pompe che erogano gas raggruppate per azienda

```
1 aziende_numero_pompe_gas = dbGetQuery(conn = con, statement =
2     "SELECT nome,count(*) AS numero_pompe_gas
3     FROM stazione_di_rifornimento AS S
4         JOIN pompa AS P on S.codice = P.stazione_serbatoio
5         JOIN serbatoio AS SE on P.numero_serbatoio = SE.numero
6         JOIN azienda AS A on S.azienda = A.partita_iva AND S.codice
    = SE.stazione
7     WHERE SE.carburante = 'gpl'
8         OR SE.carburante = 'metano'
9     GROUP BY partita_iva
10    ORDER BY numero_pompe_gas DESC;")
```

Risultato:

	nome	numero_pompe_gas
1	Agip Eni	98
2	Total Erg	73
3	Costantin	71
4	San Marco Petroli	67
5	Repsol	66
6	Api-IP	56
7	Q8	56
8	Tamoil	39
9	Esso	38

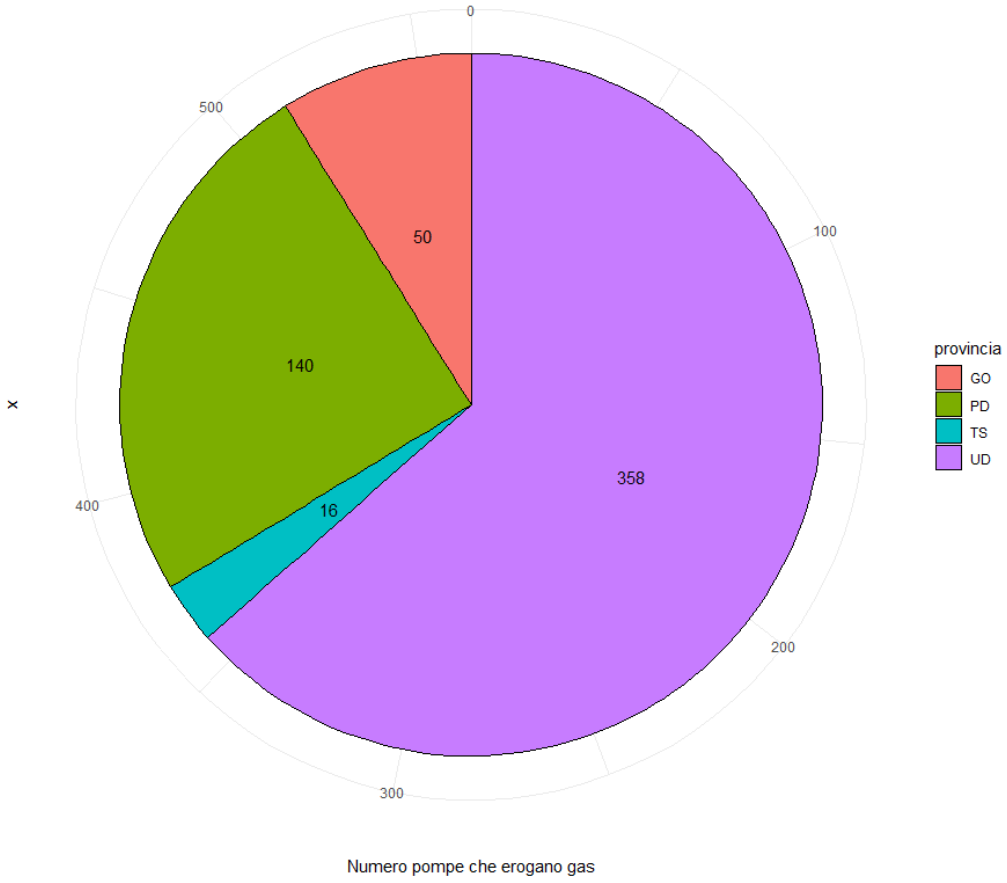


7.2.2 Numero di pompe che erogano gas raggruppate per provincia

```
provincia_numero_pompe_gas = dbGetQuery(conn = con, statement =
    "SELECT provincia,count(*) AS numero_pompe_gas
    FROM stazione_di_rifornimento AS S
        JOIN pompa AS P on S.codice = P.stazione_serbatoio
        JOIN serbatoio AS SE on P.numero_serbatoio = SE.numero
        JOIN comune AS C on S.comune = C.nome
            AND S.codice = SE.stazione
    WHERE SE.carburante = 'gpl'
        OR SE.carburante = 'metano'
    GROUP BY provincia
    ORDER BY numero_pompe_gas DESC;")
```

Risultato:

	provincia	numero_pompe_gas
1	UD	358
2	PD	140
3	GO	50
4	TS	16

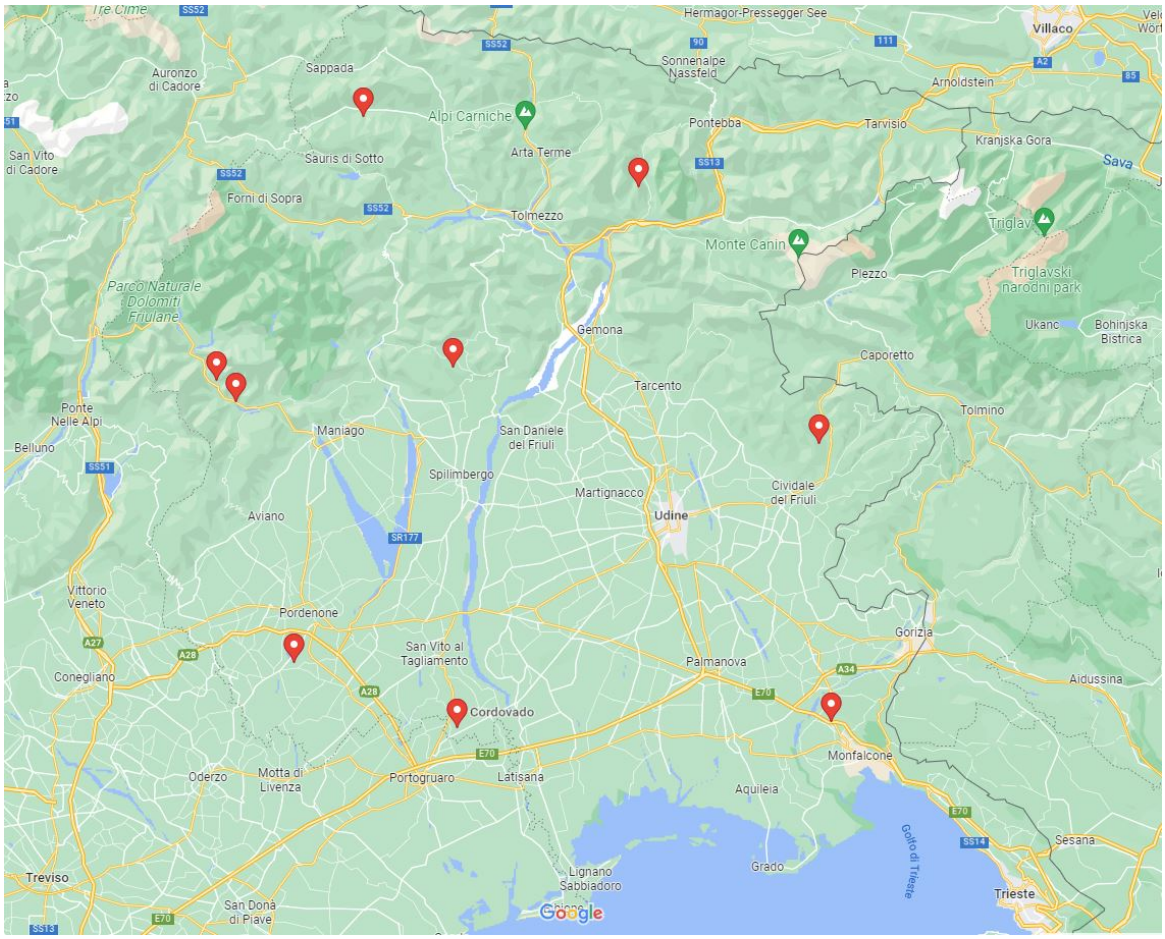


7.2.3 Serbatoi di proprietà di Agip Eni nei quali il carburante si sta esaurendo

```
pompe_eni_in_esaurimento = aziende_numero_pompe_gas = dbGetQuery(conn = con, statement =
 "SELECT stazione,numero as numero_serbatoio,quantita_residua
 FROM azienda AS A
      JOIN stazione_di_rifornimento AS S ON A.partita_iva = S.azienda
      JOIN serbatoio AS SE on S.codice = SE.stazione
 WHERE nome = 'Agip Eni'
      AND quantita_residua < 1000
 ORDER BY quantita_residua;")
```

Risultato:

	stazione	numero_serbatoio	latitudine	longitudine	quantita_residua
1	118	4	45.790	12.864	53.0495
2	118	6	45.790	12.864	320.7847
3	63	1	46.237	12.884	443.6358
4	23	3	46.152	13.470	479.1376
5	395	3	46.437	13.182	496.0997
6	195	5	45.910	12.632	502.4981
7	180	2	45.844	13.487	831.6941
8	248	2	46.515	12.739	841.8962
9	327	1	46.198	12.537	862.6504
10	106	3	46.221	12.506	906.4118



La mappa rappresenta la posizione, all'interno della regione, delle stazioni di rifornimento di Agip Eni che possiedono serbatoi i quali, in base ai dati ottenuti dalla precedente interrogazione, stanno per esaurire il carburante.