ELECTRICAL THEFT

D'AMORA AGOSTINO (N46006564) – ESPOSITO MARCELLO (N46006315)

L'idea del progetto "*Electrical Theft*" è quello di implementare un sistema che permetta all'utente finale di poter visualizzare e consultare delle tabelle in cui vi sono le medie di prezzo a kWh dei fornitori in una determinata posizione.

L'utente interagisce con la base di dati tramite l'uso di un bot Telegram.

Quando viene effettuata una richiesta al bot, esso si spetta in ingresso la Posizione sulla quale si vogliono ottenere i costi medi (espressa mediante Provincia e Regione)

• SPECIFICHE TECNICHE

Il bot sarà implementato in python, esso sarà hostato in un cloud serverless, sfruttando la possibilità di funzionare tramite hook e non con il classico metodo del polling,
ed utilizzando varie query, sarà in grado di inserire, aggiornare e consultare i dati in base alle richieste.
La necessità di avere dei valori (bollette), ha portato a pensare alla distinzione nella progettazione di due figure, ovvero:

- **Utente**: può confrontare i prezzi delle bollette, e può inserire dei valori di bolletta, che prima di essere resi validi dovranno essere validati dall'Admin

- **Admin**: può effettuare le operazioni di validazione delle bollette
(in futuro questo processo potrà anche essere automatizzato tramite intelligenza artificiale)

Oltretutto, un utente che deve inserire le bollette dovrà essere obbligatoriamente registrato, mentre un utente che deve effettuare il solo controllo dei dati potrà non essere loggato.

• SPECIFICHE SUI DATI

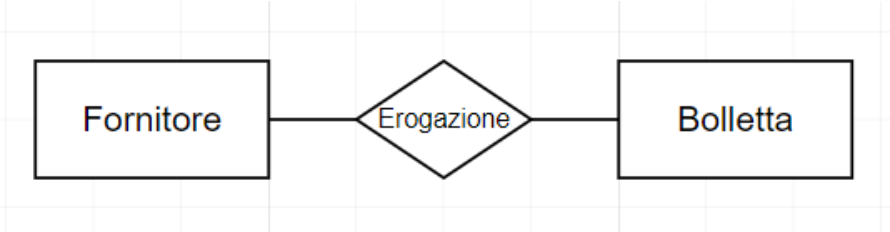
La base dati sarà implementata in modo da contenere sia le bollette (Validate e non), sia le persone che interagiscono attivamente con il bot (Utente e Admin):

- Degli utenti bisogna conservare le informazioni anagrafiche di base (nome, numero, …) e la password di accesso, solo nel caso in cui l'utente effettua la registrazione per caricare le proprie bollette (utente attivo) o sia un admin
- Le bollette devono essere identificate mediante il codice contratto e la data, dovranno avere il consumo fatturato, il costo totale ed il fornitore, inoltre dovranno essere temporaneamente non attive fino al controllo di un admin
- I fornitori devono contenere le proprie generalità di base
- La geolocalizzazione deve essere basata su Regione e Provincia

- ## SPECIFICHE SULLE OPERAZIONI
    - Inserimento di un nuovo utente
    - Inserimento di una nuova bolletta in stage
    - Aggiornamento bolletta da stage a verificata
    - Aggiornamento informazioni utente (tra cui la password)
    - Verifica delle bollette in stage (1 volta a settimana con possibile automazione)
    - Visualizzazione costi medi bolletta dei fornitori in una posizione
    - Rimozione bollette con più di 12 mesi
    - Rimozione totale utente e relative bollette quando richiesto esplicitamente

- ## TIPOLOGIE DI PERMESSI
    - Utente attivo, tramite bot Telegram può aggiungere le proprie bollette e contratti
    - Utente ghost, può solo visualizzare i costi (può non essere registrato)
    - Admin, verifica correttezza registrazioni utenti e bollette

- ## DIAGRAMMA SCHELETRO

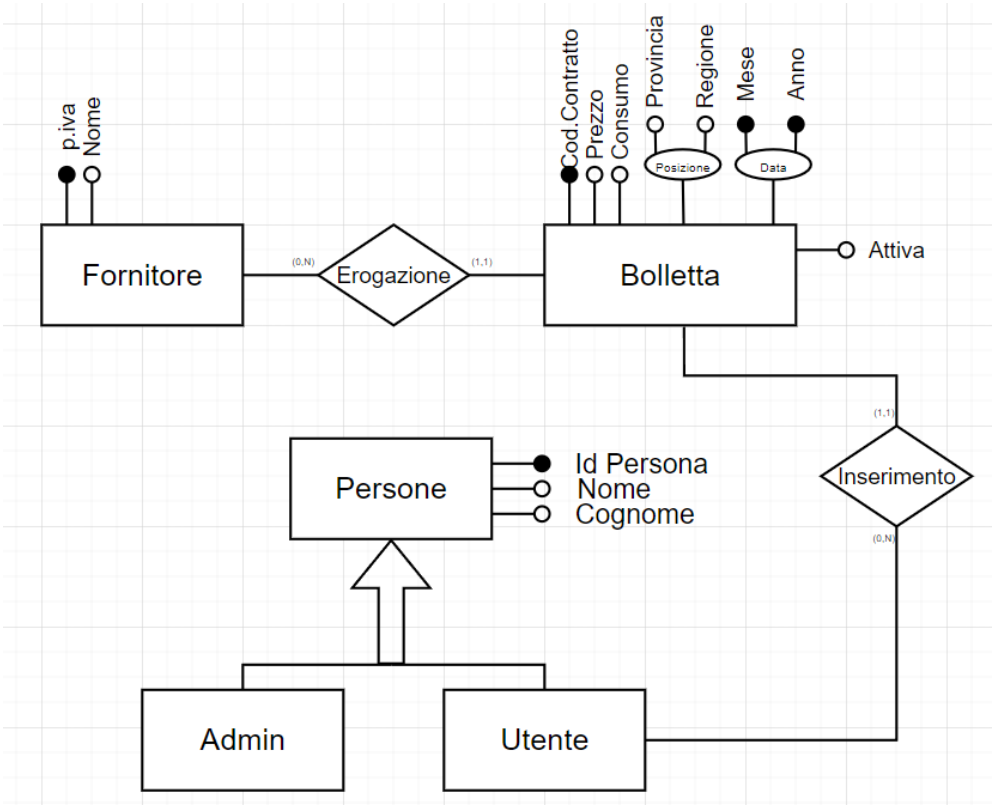


Il seguente schema scheletro è pensato basandosi sulla filosofia centrale del Progetto, ovvero, quella di poter visualizzare il Prezzo medio a kWh di un fornitore in base alla posizione. Lo schema scheletro, infatti, tiene conto dei fornitori e delle bollette.

- ## GLOSSARIO

CONCETTO	SINONIMI	DESCRIZIONE
Bolletta	Fattura	Documento sul consumo energetico mensile
Utente attivo	Uploader, Iscritto	Utente che può inserire bollette
Admin	Validatore	Utente che può validare i dati inseriti dagli utenti
Utente ghost	Utente non necessariamente iscritto	Utente che può solo visualizzare i costi
Fornitore	Erogatore	Ente che fornisce energia
Erogazione	Fornitura	Erogazione della bolletta da parte del fornitore

- # DIAGRAMMA E/R: PRIMA RIFINITURA

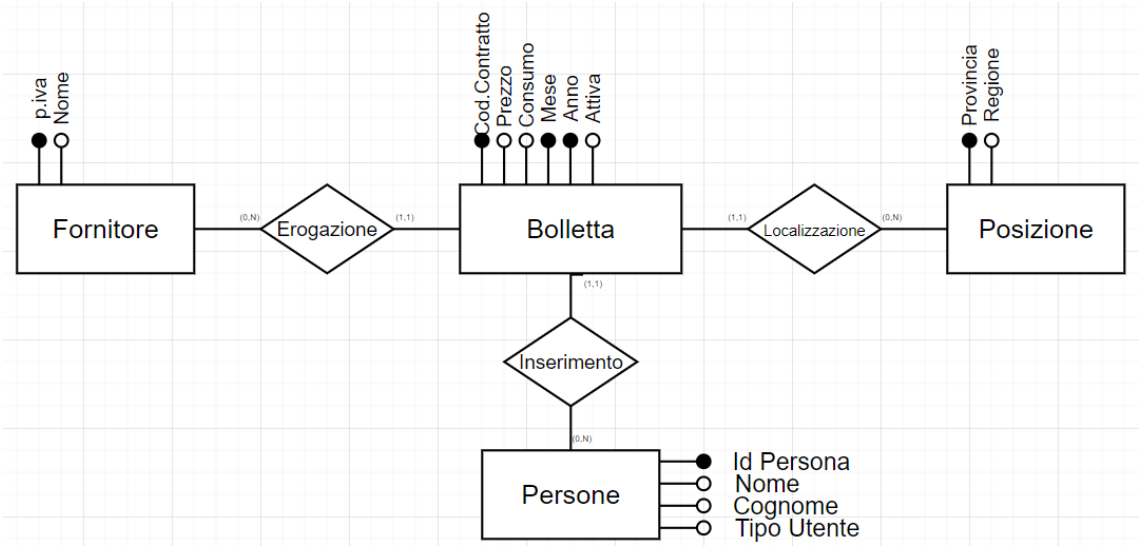


Partendo dal diagramma scheletro, si è aggiunta la parte di gestione degli utenti e degli admin in modo da poterne conservare i dati anagrafici.

- # VINCOLI

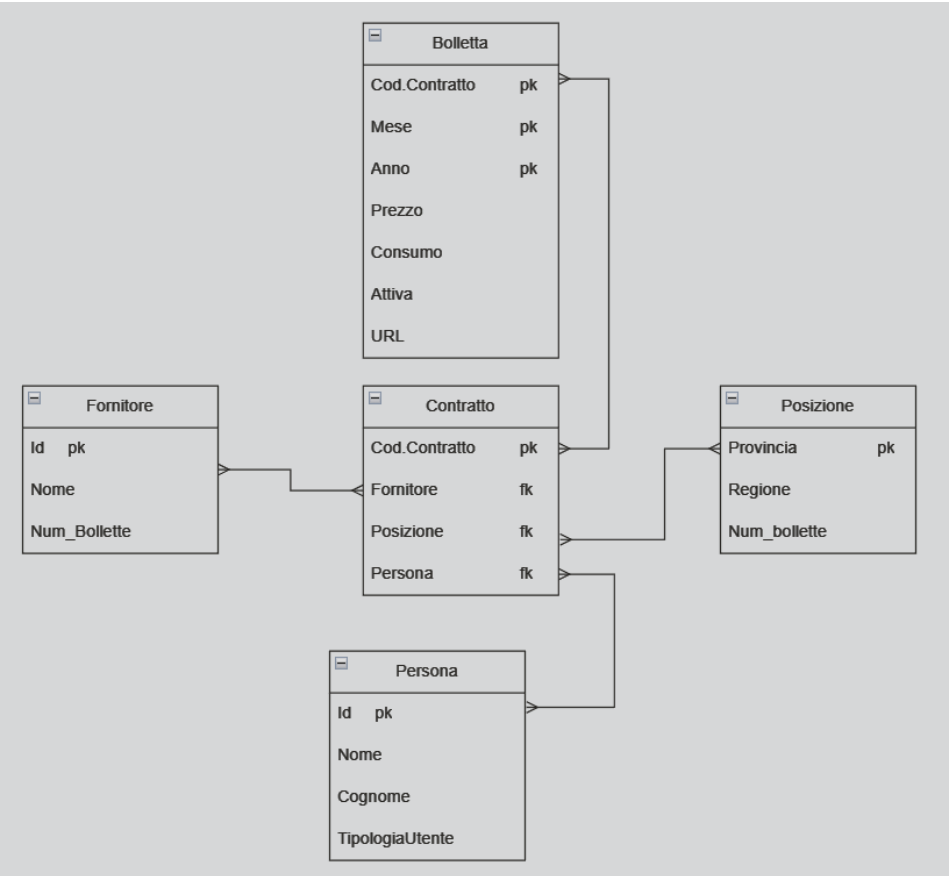
NOME	DESCRIZIONE
B1	Una bolletta deve appartenere ad una ed una sola persona
B2	Una bolletta deve essere associata ad uno ed un solo fornitore
U1	Un utente può non avere bollette
F1	Un fornitore può non aver erogato bollette
D1	Il valore dell'attributo mese deve essere compreso tra 1 e 12

- **DIAGRAMMA E/R RIFINITO: SECONDA RIFINITURA**



In questa seconda rifinitura, si pongono le basi per l'implementazione logica, andando a scomporre gli attributi composti (cercando di ottimizzare la gestione della posizione) e rimuovendo l'entità generalizzata traducendola con l'apposito attributo Tipo_Utente.

- **DIAGRAMMA LOGICO**



È stata prevista un ulteriore tabella "Contratto" poiché in fase di implementazione sono state notate delle problematiche legate all'aggiornamento dei dati.
Di conseguenza sì è effettuata una normalizzazione della "tabella critica" (Bolletta) ottenendo il precedente schema logico.

Inoltre a titolo informativo abbiamo aggiunto dei campi ridondanti, per il conteggio del numero delle bollette in riferimento al fornitore ed alla posizione.

- # TRADUZIONE SQL: ORACLE LIVE

TABELLE

Tabella Fornitore

```sql
CREATE TABLE Fornitore(
    p_iva int,
    Nome VARCHAR(50) NOT NULL,
    Num_bollette NUMBER DEFAULT 0, -- dato ridondante

    CONSTRAINT pk_fornitore PRIMARY KEY(p_iva)
);
```

Tabella Posizione

```sql
CREATE TABLE Posizione(
    Provincia   VARCHAR(2),
    Regione     VARCHAR(30) NOT NULL,
    Num_bollette NUMBER DEFAULT 0, -- dato ridondante

    CONSTRAINT pk_posizione PRIMARY KEY(Provincia)
);
```

Tabella Persona

```sql
CREATE TABLE Persona(
    IdPersona   INT DEFAULT seq_persona.NEXTVAL,
    NickName    VARCHAR(30),
    Nome        VARCHAR(30) NOT NULL,
    Cognome     VARCHAR(30) NOT NULL,
    TipologiaUtente CHAR(1) NOT NULL,   -- Utente Attivo o Admin
    Password    VARCHAR(64), -- password salvate con codifica SHA-256

    CONSTRAINT pk_persona PRIMARY KEY(IdPersona)
);
```

Tabella Bolletta

Per la Primary Key è stato scelto di utilizzare anche la data, oltre che al contratto, visto che ad un contratto possono corrispondere più bollette

```
CREATE TABLE Bolletta(
    CodContratto INT,
    Prezzo  DECIMAL NOT NULL,
    Consumo DECIMAL NOT NULL,
    Mese    INT NOT NULL,
    Anno    INT NOT NULL,
    Attiva  CHAR(1) DEFAULT 'N',
    URL VARCHAR(32),

    CONSTRAINT pk_bolletta PRIMARY KEY(CodContratto, Mese, Anno),
    CONSTRAINT fk_bolletta_contratto FOREIGN KEY(CodContratto) REFERENCES
CONTRATTO(CodContratto),
    CONSTRAINT k_mese CHECK(Mese < 13 AND Mese > 0),
    CONSTRAINT k_anno CHECK(Anno > 0)
);
```

Tabella di gestione del Contratto

```
CREATE TABLE CONTRATTO(
    CodContratto    INT DEFAULT seq_contratti.NEXTVAL,
    Fornitore   INT,
    Posizione   VARCHAR(2),
    Persona INT,

    CONSTRAINT pk_contratto PRIMARY KEY(CodContratto),
    CONSTRAINT fk_contratto_fornitore FOREIGN KEY(Fornitore) REFERENCES
FORNITORE(p_iva),
    CONSTRAINT fk_contratto_posizione FOREIGN KEY(Posizione) REFERENCES
POSIZIONE(Provincia),
    CONSTRAINT fk_contratto_persona FOREIGN KEY(Persona) REFERENCES
PERSONA(IdPersona)
);
```

PROCEDURE

Procedura di attivazione delle bollette

Dopo che un admin ha controllato tutte le bollette inviate e risultano tutte corrette, allora l'admin potrà eseguire questa procedure, che effettua una validazione a tappeto

```
CREATE OR REPLACE PROCEDURE attivazione_bolletta
IS
    CURSOR curs IS SELECT * FROM BOLLETTA WHERE ATTIVA = 'N'; --tutte le bollette disattive
    riga curs%ROWTYPE;
BEGIN
    OPEN curs;

    LOOP
        FETCH curs INTO riga;
        EXIT WHEN curs%NOTFOUND;

        UPDATE BOLLETTA
        SET ATTIVA = 'Y'
        WHERE CODCONTRATTO = riga.CODCONTRATTO AND MESE = riga.MESE AND ANNO = riga.ANNO;

        DBMS_OUTPUT.PUT_LINE('Aggiornata bolletta: ' || riga.CODCONTRATTO);
    END LOOP;

    CLOSE curs;
END;
```

Procedura di eliminazione di una bolletta

Durante un periodo in cui non vi sono molte visite sul bot, è possibile schedulare un avvio della seguente procedure, che permette la rimozione di tutte le bollette 'scadute' (con più di 12 mesi)

```
CREATE OR REPLACE PROCEDURE elimina_record IS
    CURSOR curs IS SELECT * FROM BOLLETTA WHERE to_char( sysdate, 'mm' ) > to_char(MESE)
AND to_char( sysdate, 'yy' ) > to_char(ANNO);
    riga curs%ROWTYPE;
BEGIN
    OPEN curs;

    LOOP
        FETCH curs INTO riga;
        EXIT WHEN curs%NOTFOUND;

        DELETE FROM BOLLETTA WHERE riga.CODCONTRATTO = CODCONTRATTO AND riga.MESE = MESE
AND riga.ANNO = ANNO;

    END LOOP;
END;
```

Procedura per il refresh della vista

Purtroppo la vista, effettuando operazioni su tutti i dati, nel caso in cui siano molti, richiede molto tempo per essere rielaborata, quindi per non sovraccaricare il DBMS durante gli orari di maggior flusso, è stato scelto di aggiornare la vista materializzata dei 'costi' durante ore di minor affluenza.

```
CREATE OR REPLACE PROCEDURE prc_update_cost IS
BEGIN
    dbms_mview.refresh('COSTI'); -- Comando specifico del DBMS
END;
```

TRIGGER

Trigger di aggiornamento ridondanze (INSERT)

Utilizzato per aggiornare il numero delle bollette di un fornitore / posizione nel caso di inserimenti di bollette attive direttamente (inserimento diretto da admin).

```
CREATE OR REPLACE TRIGGER trg_insert_bolletta
AFTER INSERT ON BOLLETTA
FOR EACH ROW
DECLARE
    cod_fornitore CONTRATTO.FORNITORE%TYPE;
    cod_posizione CONTRATTO.POSIZIONE%TYPE;
BEGIN
   -- Incremento solo nel caso in cui la bolletta inserita sia già attiva
    IF (:NEW.ATTIVA = 'Y') THEN
        SELECT c.fornitore, c.posizione INTO cod_fornitore, cod_posizione
        FROM CONTRATTO c
        WHERE c.codContratto = :NEW.codContratto;

        UPDATE FORNITORE f
        SET f.NUM_BOLLETTE = (f.NUM_BOLLETTE + 1)
        WHERE f.p_iva = cod_fornitore;

        UPDATE POSIZIONE p
        SET p.NUM_BOLLETTE = (p.NUM_BOLLETTE + 1)
        WHERE p.PROVINCIA = cod_posizione;
    END IF;
END;
```

Trigger di aggiornamento ridondanze (UPDATE)

Quando viene aggiornato lo stato di una o più bollette, il seguente trigger si attiva per aggiornare il numero di bollette in base al fornitore / posizione.
Nel caso in cui la bolletta venga attivata il numero viene incrementato, viceversa viene decrementato

```
CREATE OR REPLACE TRIGGER trg_update_bolletta
AFTER UPDATE OF ATTIVA ON BOLLETTA
FOR EACH ROW
DECLARE
    cod_fornitore CONTRATTO.FORNITORE%TYPE;
    cod_posizione CONTRATTO.POSIZIONE%TYPE;
BEGIN
    SELECT c.fornitore, c.posizione INTO cod_fornitore, cod_posizione
    FROM CONTRATTO c
    WHERE c.codContratto = :NEW.codContratto;

  -- Se la bolletta viene attivata, incrementiamo campi ridondanti
    IF (:NEW.ATTIVA = 'Y') THEN
        UPDATE FORNITORE f
        SET f.NUM_BOLLETTE = (f.NUM_BOLLETTE + 1)
        WHERE f.p_iva = cod_fornitore;

        UPDATE POSIZIONE p
        SET p.NUM_BOLLETTE = (p.NUM_BOLLETTE + 1)
        WHERE p.PROVINCIA = cod_posizione;
  -- Se la bolletta viene disattivata, decrementiamo campi ridondanti
    ELSE
        UPDATE FORNITORE f
        SET f.NUM_BOLLETTE = (f.NUM_BOLLETTE - 1)
        WHERE f.p_iva = cod_fornitore;

        UPDATE POSIZIONE p
        SET p.NUM_BOLLETTE = (p.NUM_BOLLETTE - 1)
        WHERE p.PROVINCIA = cod_posizione;
    END IF;
END;
```

Trigger di aggiornamento delle ridondanze (DELETE)

Quando viene rimossa una bolletta attiva, il seguente trigger provvede al decremento delle variabili di conteggio

```
CREATE OR REPLACE TRIGGER trg_delete_bolletta
AFTER DELETE ON BOLLETTA
FOR EACH ROW
DECLARE
    cod_fornitore CONTRATTO.FORNITORE%TYPE;
    cod_posizione CONTRATTO.POSIZIONE%TYPE;
BEGIN
    IF (:OLD.attiva = 'Y') THEN
        SELECT c.fornitore, c.posizione INTO cod_fornitore, cod_posizione
        FROM CONTRATTO c
        WHERE c.codContratto = :OLD.codContratto;

        UPDATE FORNITORE f
        SET f.NUM_BOLLETTE = (f.NUM_BOLLETTE - 1)
        WHERE f.p_iva = cod_fornitore;

        UPDATE POSIZIONE p
        SET p.NUM_BOLLETTE = (p.NUM_BOLLETTE - 1)
        WHERE p.PROVINCIA = cod_posizione;
    END IF;
END;
```

VISTE

Vista per la gestione dei costi

Vista consultata dal bot per inviare le risposte agli utenti in maniera rapida, senza dover far eseguire delle specifiche query al DBMS

```
CREATE MATERIALIZED VIEW COSTI AS
SELECT C.Posizione AS Citta, F.Nome AS Fornitore,
TRUNC(SUM(B.Prezzo)/SUM(B.Consumo), 2) AS Prezzo_in_kWh
-- Trunc per avere max. 2 decimali
FROM Bolletta B
JOIN CONTRATTO C ON B.CodContratto = C.CodContratto
JOIN FORNITORE F ON C.Fornitore = F.p_iva
WHERE B.Attiva = 'Y'
GROUP BY C.Posizione, F.Nome;
```

Vista Regioni

Vista consultata dal bot, per visualizzare le sole regioni evitando di elaborare query apposite

```
CREATE MATERIALIZED VIEW REGIONI AS
SELECT REGIONE
FROM POSIZIONE
GROUP BY REGIONE
ORDER BY REGIONE ASC;
```

SEQUENCE

Sequence ID persona

Generazione di un ID univoco associato all'utente, all'atto della registrazione

```
CREATE SEQUENCE seq_persona
INCREMENT BY 1
START WITH 1
NOMAXVALUE
NOMINVALUE
NOCYCLE
NOCACHE;
```

Sequenza ID Contratto

Genera un codice univoco associato al contratto in modo da non dover immagazzinare un codice di contratto reale, che potrebbe risultare in una complicazione riguardo alle normative sulla privacy EU

```
CREATE SEQUENCE seq_contratti
INCREMENT BY 1
START WITH 1
NOMAXVALUE
NOMINVALUE
NOCYCLE
NOCACHE;
```

Per vedere alcuni esempi di operazioni DML sulla base di dati, è possibile eseguire lo script al seguente link .

Oppure importare il file .sql allegato nella mail

(Per questioni di compatibilità è consigliato fare l'import e non un semplice copia e incolla)