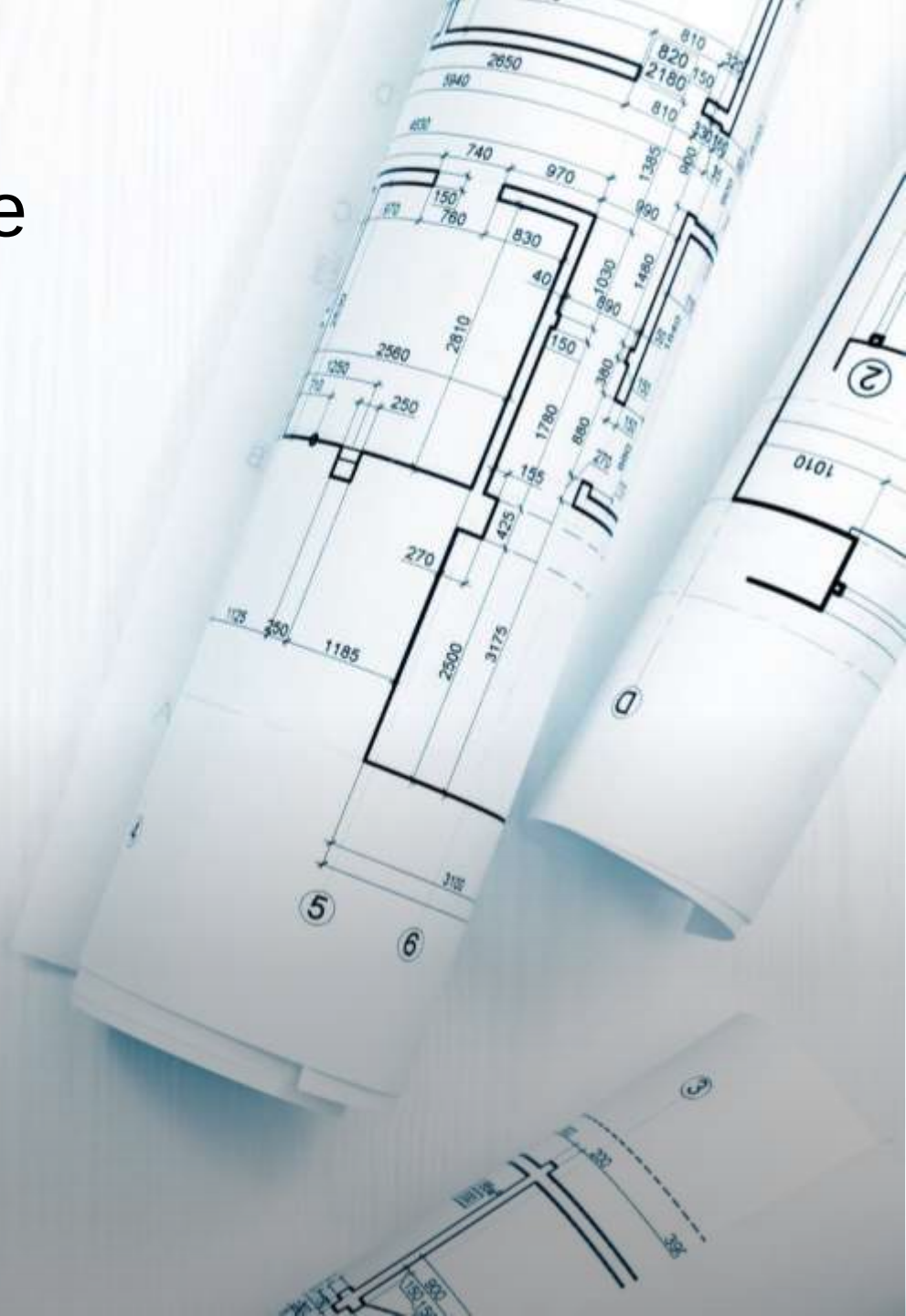
Progetto DADAU Disinfettazione Agile Di Aree Urbane

Gestione di campagne di disinfettazione agile di aree urbane (strade, mezzi, ecc.) in situazioni di emergenza socio-sanitaria

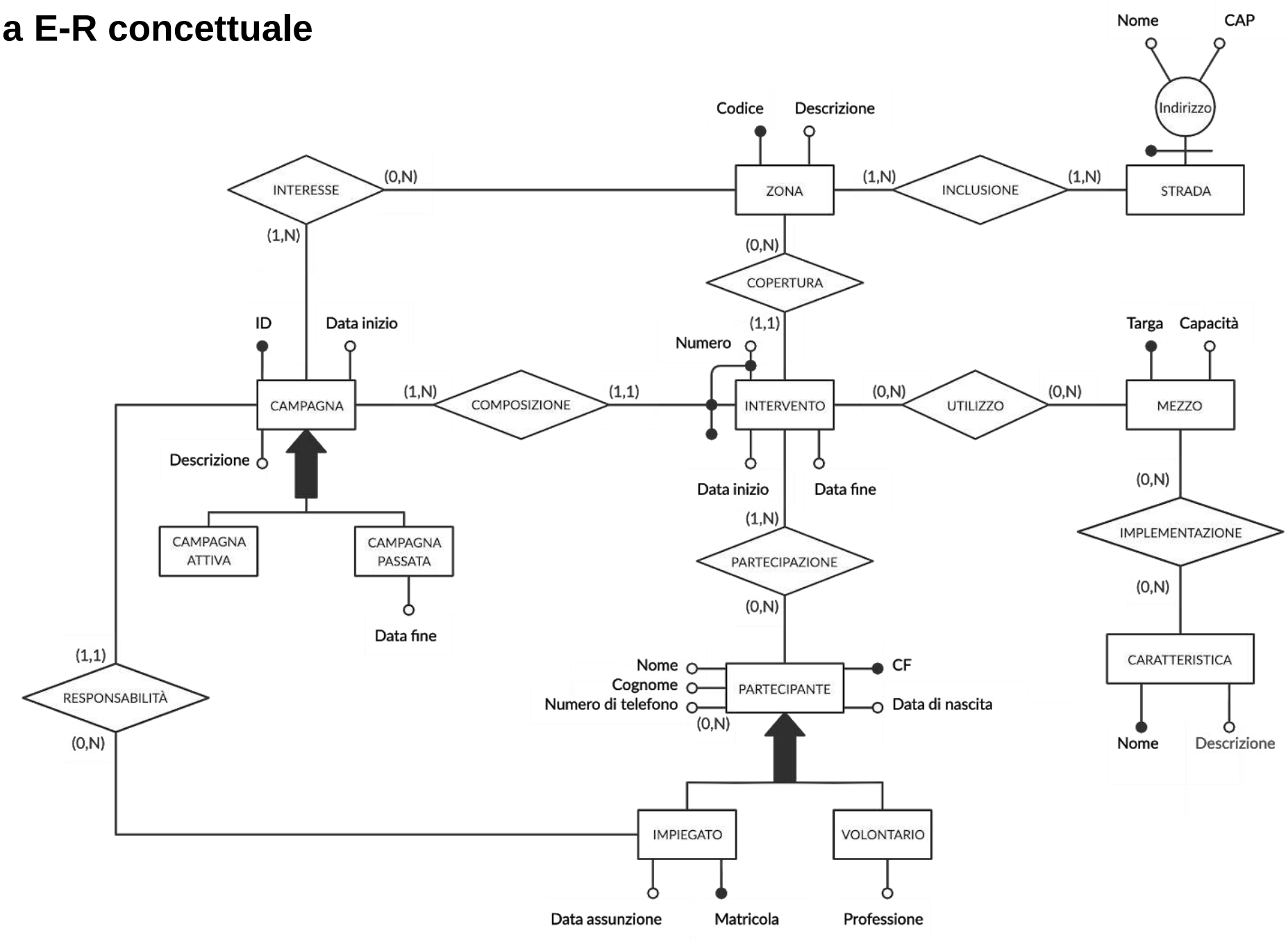
A cura del Gruppo 21, composto da:
Vincenzo Riccio - 0612703920
Giancarlo Sorrentino - 0612704179
Lia Trapanese - 0612704183
Emanuele Triuzzi - 061270400

Docenti: Matteo Gaeta, Giuseppe D'Aniello

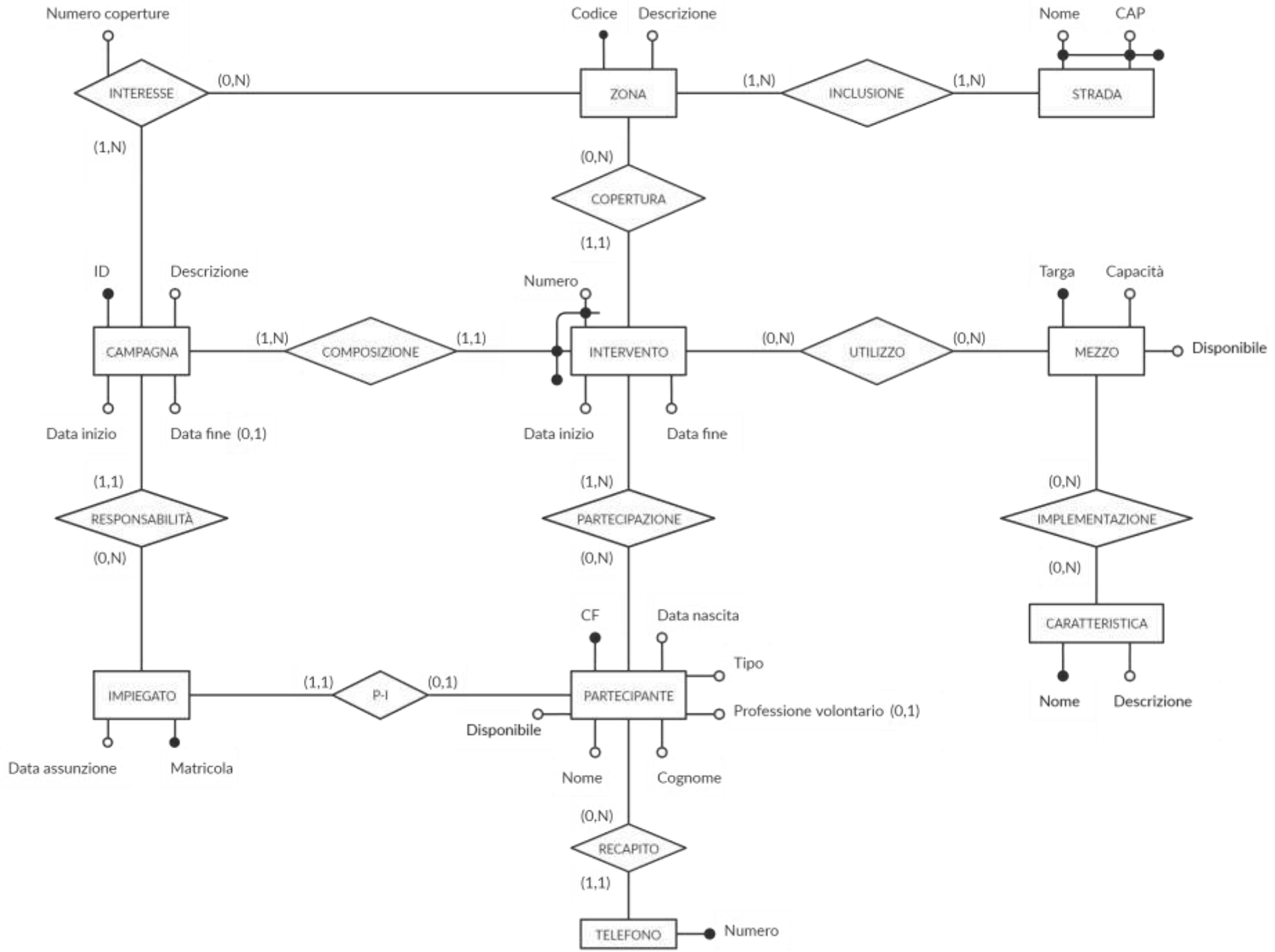


Università degli Studi di Salerno
Basi di Dati - A.A. 2019/2020

Schema E-R concettuale

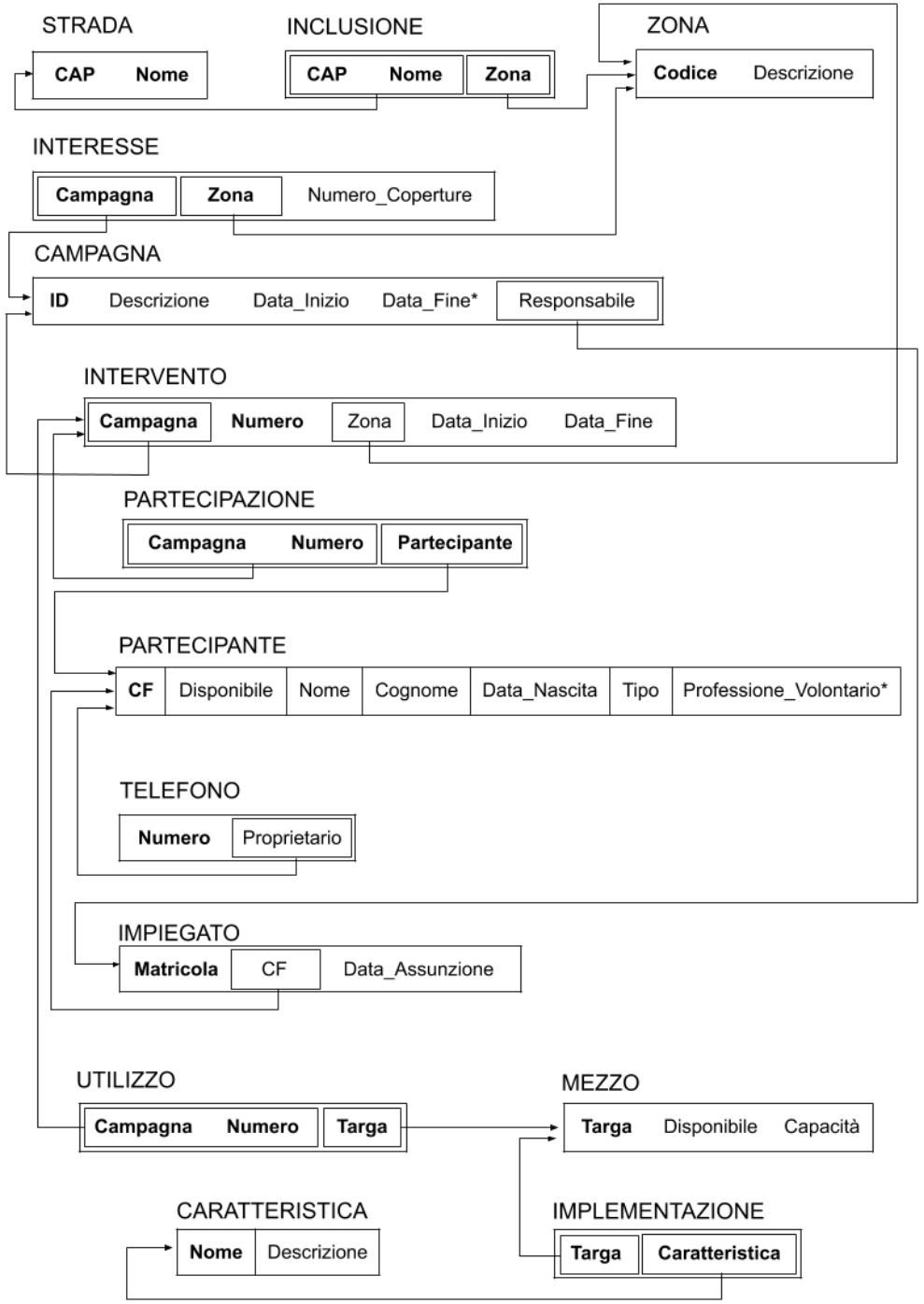


Schema E-R ristrutturato



Schema Logico e Vincoli di Foreign Key

Inclusione(Zona) REFERENCES Zona(Codice);
Inclusione(CAP, nome) REFERENCES Strada(CAP, nome);
Interesse(Campagna)REFERENCES Campagna(ID);
Interesse(Zona) REFERENCES Zona(Codice);
Campagna(Responsabile) REFERENCES Impiegato(Matricola);
Intervento(Campagna) REFERENCES Campagna(ID);
Intervento(Zona) REFERENCES Zona(Codice);
Partecipazione(Partecipante) REFERENCES Partecipante(CF);
Partecipazione(Campagna, Numero) REFERENCES Intervento(Campagna, Numero);
Impiegato(CF) REFERENCES Partecipante(CF);
Telefono(Proprietario) REFERENCES Partecipante(CF);
Utilizzo(Targa) REFERENCES Mezzo(Targa);
Utilizzo(Campagna, Numero) REFERENCES Intervento(Campagna, Numero);
Implementazione(Targa) REFERENCES Mezzo(Targa);
Implementazione(Caratteristica) REFERENCES Caratteristica(Nome);



QUERY

```sql
-- Op8: Ricerca dei mezzi disponibili in un dato intervallo temporale
CREATE OR REPLACE FUNCTION mezzi_disponibili (ora_inizio TIMESTAMP, ora_fine TIMESTAMP)
RETURNS TABLE (targa T_TARGA, capacità UNSIGNED_INT, caratteristica VARCHAR(20)) AS $$
    BEGIN
    RETURN QUERY
        SELECT M.targa, M.capacità, M.caratteristica_implementata AS Caratteristica
        FROM mezzi_con_caratteristiche AS M
        WHERE M.disponibile AND M.targa NOT IN (
            SELECT U.Targa --Ottengo tutti i mezzi utilizzati nell'intervallo di interesse
            FROM intervento I
            JOIN utilizzo U
            ON (I.campagna = U.campagna AND I.numero = U.numero)
            WHERE (I.data_inizio BETWEEN ora_inizio AND ora_fine) OR (I.data_fine BETWEEN ora_inizio AND ora_fine)
        )
        ORDER BY M.targa;
    END;
$$
LANGUAGE 'plpgsql';

-- Op9: Ricerca di partecipanti disponibili in un dato intervallo temporale
CREATE OR REPLACE FUNCTION partecipanti_disponibili (ora_inizio TIMESTAMP, ora_fine TIMESTAMP)
RETURNS TABLE (Tipo VARCHAR(10), CF T_CF, Nome VARCHAR (20), Cognome VARCHAR(20), Data_Di_Nascita DATE, Professione_Volontario VARCHAR(15)) AS $$
    BEGIN
    RETURN QUERY
        SELECT P.tipo, P.CF, P.Nome, P.Cognome, P.Data_Nascita, P.Professione_Volontario
        FROM partecipante P
        WHERE P.disponibile AND NOT EXISTS (
            SELECT PZ.partecipante
            FROM intervento I
            JOIN partecipazione PZ
            ON (I.campagna = PZ.campagna AND I.numero = PZ.numero)
            WHERE PZ.partecipante = P.CF AND I.data_inizio BETWEEN ora_inizio AND ora_fine OR I.data_fine BETWEEN ora_inizio AND ora_fine
        )
        ORDER BY P.tipo, P.cognome;
    END;
$$
LANGUAGE 'plpgsql';
```

```sql
-- Query insiemistica che restituisce tutte le zone che
-- non sono mai state coperte da un intervento in una campagna
SELECT Z.codice, Z.descrizione
FROM interesse I
JOIN zona Z ON (I.zona = Z.codice)
WHERE I.campagna = 'DISINF-01'
EXCEPT
SELECT Z.codice, Z.descrizione
FROM intervento I
JOIN zona Z ON (I.zona = Z.codice)
WHERE I.Campagna = 'DISINF-01';
```

```sql
--Numero di interventi effettuati sulle zone
--coperte almeno due volte, in base alla campagna
SELECT zona as Zona_Coperta, COUNT(zona) as interventi_effettuati, campagna
FROM intervento
GROUP BY campagna, zona
HAVING COUNT(zona)>=2
ORDER BY zona, campagna;
```

```sql
--Media del numero di partecipanti per gli interventi di una campagna che hanno coinvolto almeno due mezzi
SELECT R.campagna, CAST(AVG(R.numero_partecipanti) as numeric(10,2)) as media_partecipanti
FROM (
    SELECT I.campagna AS Campagna, I.Numero AS Intervento, count(DISTINCT P.partecipante) AS Numero_Partecipanti
    FROM intervento I, partecipazione P, utilizzo U
    WHERE I.campagna = P.campagna AND I.numero = P.Numero AND I.Campagna = U.Campagna AND I.Numero = U.Numero
    GROUP BY I.campagna, I.numero
    HAVING count(DISTINCT U.targa)>=2
) AS R
GROUP BY R.campagna;
```

VISTA e RELATIVA QUERY

```sql
-- Numero di interventi effettuati da ogni partecipante (se >0) in ogni campagna
CREATE OR REPLACE VIEW partecipazioni_campagna (id_campagna, partecipante, interventi_effettuati) AS
    SELECT campagna, partecipante, count(*) AS numero_partecipazioni
    FROM partecipazione
    GROUP BY campagna, partecipante
    ORDER BY numero_partecipazioni DESC;


--Partecipante più attivo in ogni campagna
SELECT PC.id_campagna, P.cf, P.nome, P.cognome, PC.interventi_effettuati
FROM partecipazioni_campagna PC
JOIN partecipante P
ON PC.partecipante = P.cf
WHERE (id_campagna, interventi_effettuati) IN (
    --Numero massimo di interventi effettuati da un singolo partecipante
    --nell'ambito di una stessa campagna
    SELECT id_campagna, MAX(interventi_effettuati)
    FROM partecipazioni_campagna
    GROUP BY id_campagna
);
```

TRIGGER

```sql
CREATE OR REPLACE FUNCTION update_campagna()
RETURNS TRIGGER AS $$
    BEGIN
        IF NEW.responsabile NOT IN (
            SELECT I.matricola
            FROM impiegato I
            WHERE I.matricola = NEW.responsabile AND I.cf IN (
                SELECT P.cf
                FROM partecipante P
                WHERE P.tipo = 'impiegato' AND P.disponibile
            )
        ) THEN
            RAISE EXCEPTION 'Il responsabile selezionato non è disponibile';
        END IF;

        IF NEW.data_fine IS NOT NULL THEN
            IF EXISTS (
                SELECT *
                FROM intervento I
                WHERE I.campagna = NEW.id AND I.data_fine >= LOCALTIMESTAMP
            ) THEN
                RAISE EXCEPTION 'Non è possibile concludere una campagna i cui interventi non sono ancora conclusi';
            END IF;

            IF EXISTS (
                SELECT *
                FROM interesse I
                WHERE I.campagna = NEW.id AND I.numero_coperture = 0
            ) THEN
                RAISE EXCEPTION 'Non è possibile concludere una campagna le cui zone non sono state coperte da almeno un intervento';
            END IF;
        END IF;
        RETURN NEW;
    END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER trig_update_campagna AFTER UPDATE ON campagna
FOR EACH ROW
EXECUTE PROCEDURE update_campagna();
```

TRIGGER

```
CREATE OR REPLACE FUNCTION update_partecipante()
RETURNS TRIGGER AS $$
    BEGIN
        IF OLD.disponibile = TRUE AND NEW.disponibile = FALSE THEN
            IF NEW.tipo = 'impiegato' AND EXISTS (
                SELECT *
                FROM campagna C
                WHERE C.data_fine IS NULL AND C.responsabile IN (
                    SELECT I.matricola
                    FROM impiegato I
                    WHERE NEW.cf = I.cf
                )
            ) THEN
                RAISE EXCEPTION 'Il partecipante indicato è responsabile di una campagna attiva';
            END IF;
            IF NEW.disponibile = FALSE AND NEW.cf IN (
                SELECT P.partecipante
                FROM intervento I
                JOIN partecipazione P ON (I.campagna = P.campagna AND i.numero = P.numero)
                WHERE LOCALTIMESTAMP <= I.data_fine
            ) THEN
                RAISE EXCEPTION 'Non è possibile modificare la disponibilità di un partecipante coinvolto in un intervento ancora non concluso';
            END IF;
        END IF;

        IF OLD.tipo = 'volontario' AND NEW.tipo = 'impiegato' THEN
            IF NOT EXISTS (
                SELECT *
                FROM impiegato I
                WHERE I.cf = NEW.cf
            ) THEN
                RAISE EXCEPTION 'Impossibile cambiare da volontario a impiegato se questo non esiste già nella tabella impiegato';
            END IF;
        END IF;
```

TRIGGER

```
        IF OLD.tipo = 'impiegato' AND NEW.tipo = 'volontario' THEN
            IF EXISTS (
                SELECT *
                FROM campagna
                WHERE responsabile IN (
                    SELECT matricola
                    FROM impiegato I
                    WHERE I.cf = NEW.cf
                )
            ) THEN
                RAISE EXCEPTION 'Impossibile cambiare da impiegato a volontario se questo è stato il responsabile di una campagna';
            END IF;
            DELETE FROM impiegato WHERE cf = NEW.cf;
        END IF;
        RETURN NULL;
    END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER trig_partecipante_update AFTER UPDATE ON partecipante
FOR EACH ROW
EXECUTE PROCEDURE update_partecipante();
```

```
CREATE OR REPLACE FUNCTION delete_intervento()
RETURNS TRIGGER AS $$
    BEGIN
        IF OLD.campagna IN (
            SELECT C.id
            FROM campagna C
            WHERE data_fine IS NOT NULL
        ) THEN
            RAISE EXCEPTION 'Non è possibile rimuovere un intervento da una campagna già terminata';
        END IF;
        UPDATE interesse
        SET numero_coperture = numero_coperture-1
        WHERE zona = OLD.zona AND campagna = OLD.campagna;
        RETURN NULL;
    END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER trig_intervento_delete AFTER DELETE ON intervento
FOR EACH ROW
EXECUTE PROCEDURE delete_intervento();
```