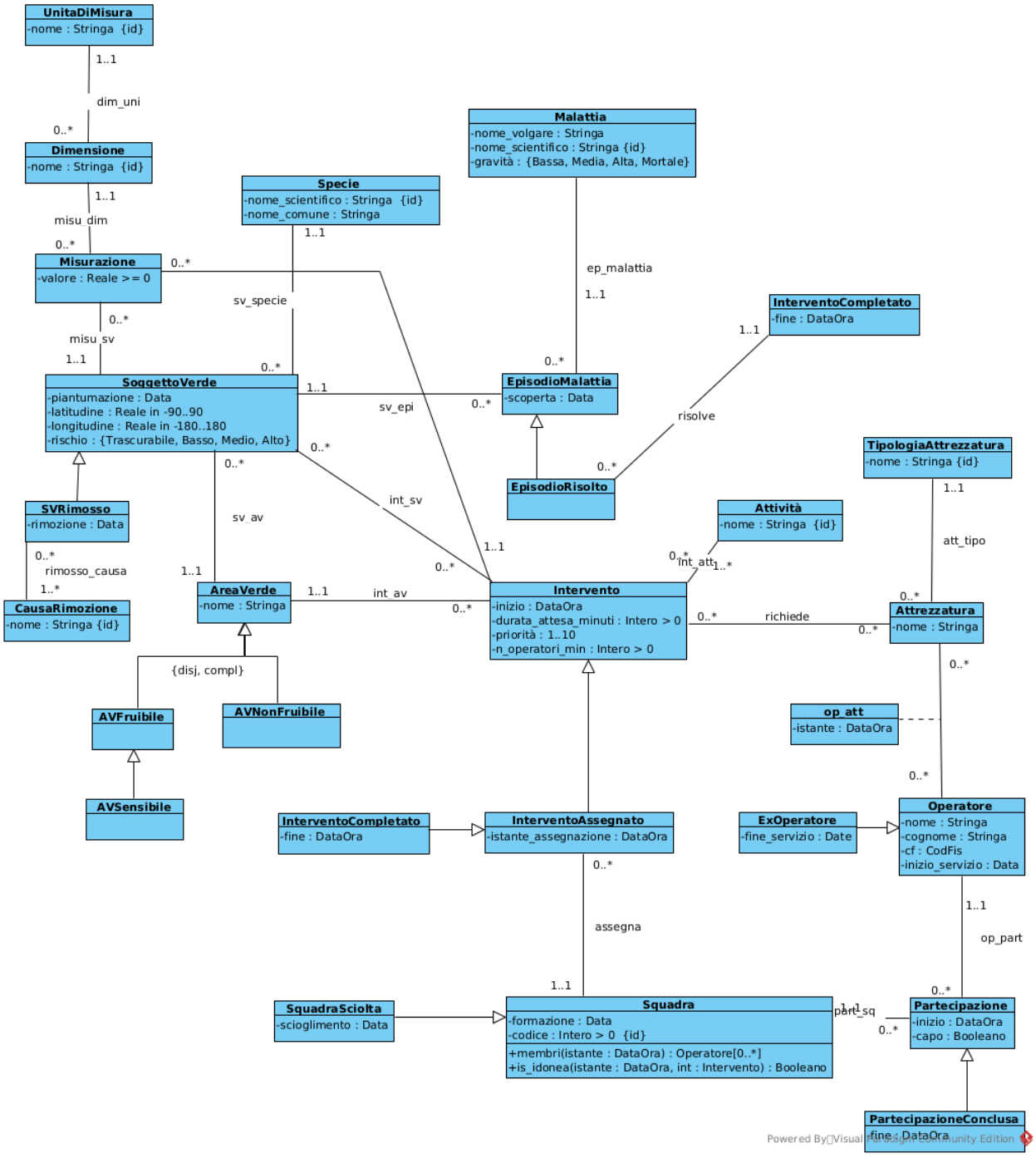
# Città Verde

## Assunzioni

1. Un operatore non può avere più di un periodo di servizio (no assunzione-licenziamento-assunzione- ...)
2. Un operatore può appartenere a più di una squadra nello stesso istante
3. Due interventi distinti possono trattare la stessa area o lo stesso soggetto verde
4. Due interventi di due squadre che hanno operatori in comune possono sovrapporsi nel tempo

# 1. Analisi Concettuale

## 1.1 Diagramma UML concettuale delle classi



## 1.2. Specifica dei tipi di dato

- CodFis: Stringa secondo standard

## 1.3. Specifica delle classi

### 1.3.1. Specifica della classe SoggettoVerde

[V.SoggettoVerde.EpisodioMalattia.Scoperta_dopo_piantumazione]
$\forall\, s,\, e,\, p,\, sc\; SoggettoVerde(s) \;\wedge\; EpisodioMalattia(e) \;\wedge$
$sv\_epi(s,\, e) \;\wedge\; piantumazione(s,\, p) \;\wedge\; scoperta(e,\, sc) \;\rightarrow$
$p \;<\; sc \;\wedge\; (\forall\, r\; rimozione(s,\, r) \;\rightarrow\; sc \;\leq\; r)$

### 1.3.2. Specifica della classe SVRimosso

[V.SVRimosso.dopo_piantumazione]
$\forall s,\, p,\, r\;\; SVRimosso(s) \;\wedge\; piantumazione(s,\, p) \;\wedge\; rimozione(s,\, r) \;\rightarrow\; p \;<\; r$

### 1.3.3. Specifica della classe Squadra

#### is_idonea(istante: DataOra, int: Intervento): Booleano

Precondizioni:

- Sia f tale che formazione(this, f). Deve essere vero che f <= istante
- $\forall\, s\; scioglimento(this,\, s) \;\rightarrow\; istante \;\leq\; s$

Postcondizioni
Sia n_min tale da soddisfare *n_operatori_min(int)*
*| membri(this, istante) | < n_min → result = False*
O, più formalmente
*|{ op | membri(this, istante, op) }| < n_min → result = False*

result = True ⇔ [
$\forall\, a \quad richiede(int,\, a) \quad \rightarrow \quad [\exists\, op,\, ia\; membri(this,\, istante,\, op) \;\wedge\; op\_att(op,\, a) \;\wedge\; istante(op,\, a,\, ia) \;\wedge\; ia \;<$
]

membri(istante: DataOra): Operatore [0..*]

Precondizioni:

- Sia f tale che formazione(this, f). Deve essere vero che f <= istante
- $\forall\, s\ scioglimento(this,\ s)\ \rightarrow\ istante\ \leq\ s$

Postcondizioni:*Sfrutta il vincolo V.Operatore.partecipazione.in_servizio*

$result\ =\ \{\ op\ |\ \exists\ p,\ ip\ Partecipazione(p)\ \wedge\ op\_part(op,\ p)\ \wedge\ part\_sq(this,\ p)$
$\wedge\ inizio(p,\ ip)\ \wedge\ ip\ \leq\ istante\ \wedge\ (\forall fp\ fine(p,\ fp)\ \rightarrow fp\ \geq\ istante)\}$

1.3.4. Specifica della classe operatore

*Un operatore non può partecipare a una squadra in un istante se non è in servizio in tale istante* [V.Operatore.partecipazione.in_servizio]

$\forall op,\ is,\ p,\ ip\ inizio\_servizio(op,\ is)\ \wedge\ inizio(p,\ ip)\ \rightarrow$
$ip\ \geq\ is\ \wedge\ (\forall\ fs\ fine\_servizio(op,\ fs)\ \rightarrow\ \exists\ fp\ fine(p,\ fp)\ \wedge\ fp\ \leq\ fs$

# 1.4. Specifica dei Vincoli Esterni

**overlaps($i_1$: DataOra, $f_1$: DataOra, $i_2$: DataOra, $f_2$: DataOra): Booleano**
Precondizioni: $i_1$ <= $f_1$ AND $i_2$ <= $f_2$
Postcondizioni:

Result = True ⇔[ $\exists\, t\ DataOra(t)\ \wedge i_1\ \leq t \leq f_1\ \wedge\ i_2 \leq t\ \leq f_2$]

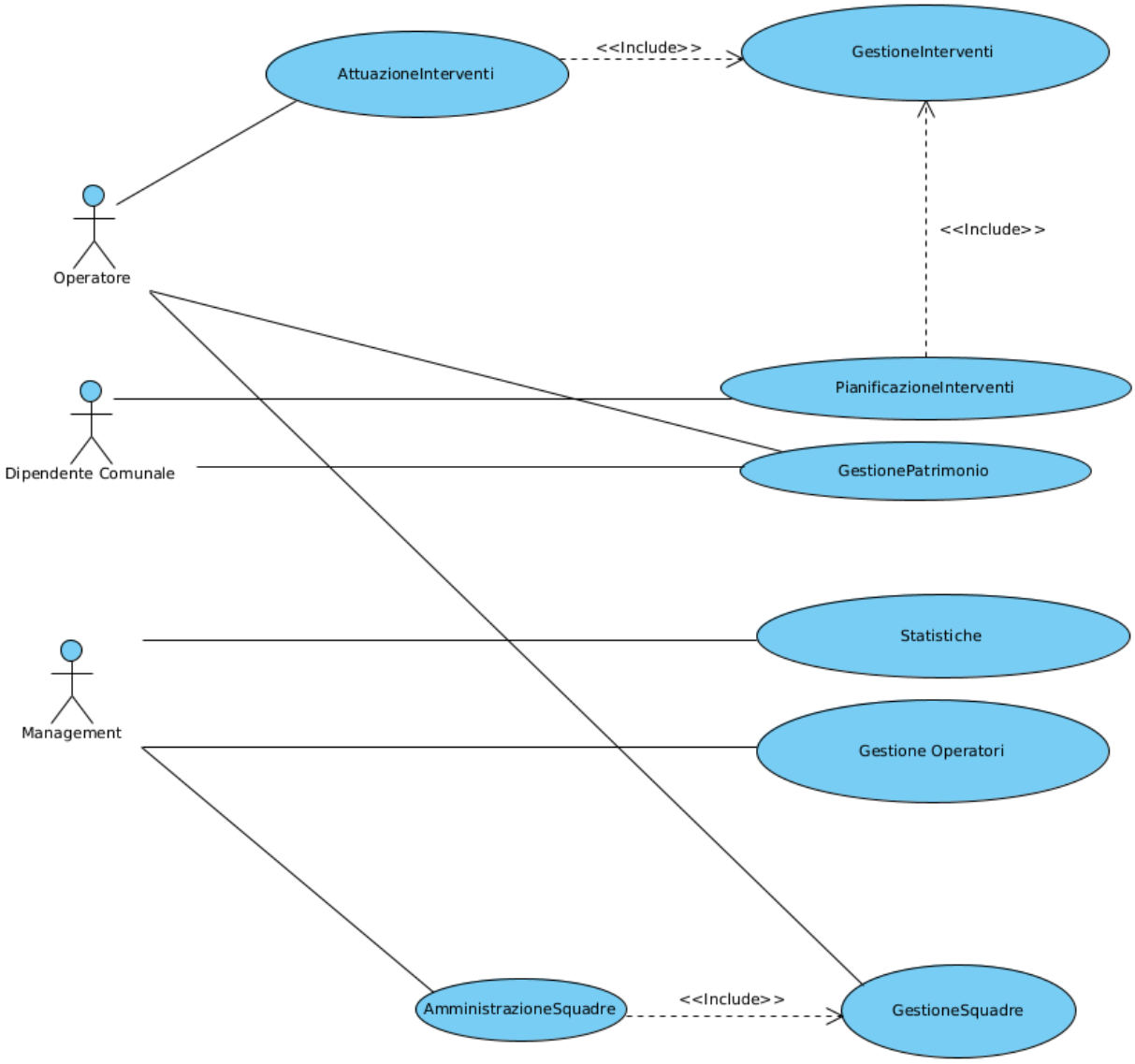
[V.Intervento.SV.in_area]
$\forall\ i,\ s,\ a\ \ int\_av(i,\ a)\ \wedge\ int\_sv(i,\ s)\ \rightarrow\ sv\_av(s,\ a)$

*C'è un vincolo simile su misurazione, intervento e soggettoVerde*

[V.Intervento.assegnato.Squadra.solo_se_idonea]
$\forall\ int,\ s,\ i\ InterventoAssegnato(int)\ \wedge\ istante\_assegnazione(int,\ i)\ \wedge$
$assegna(int,\ s)\ \rightarrow\ is\_idonea(s,\ i,\ int,\ True)$

[V.Squadra.un_solo_capo_alla_volta]
*Per ogni squadra non devono esistere due partecipazioni distinte che si sovrappongono nel tempo e con capo = True. (lasciata come esercizio)*

## 1.5. Diagramma UML degli Use-Case



## 1.6. Specifica degli Use-Case

### 1.6.1. Specifica dello Use-Case *PianificazioneInterventi*

squadre_idonee(int: InterventoPianificato): Squadra [0..*]

Precondizioni: not InterventoAssegnato(int)
Postcondizioni:

- L'operazione non modifica i dati
- Il risultato *result* è definito come segue:

*result = { s | Squadra(s) ∧ not SquadraSciolta(s) ∧ is_idonea(s, adesso, int, True) }*

## 1.6.2. Specifica dello Use-Case *GestionePatrimonio*

### aree_verdi_no_intervento(inizio: Data, fine: Data): AVSensibile [0..*]

Precondizioni: inizio <= fine
Postcondizioni

- L'operazione non modifica i dati
- Il risultato *result* è così definito

$$result = \{av \mid AVSensibile(av) \land \neg\exists\, int, i, Intervento(int) \land int\_av(int, av) \land inizio(int, i) \land i \leq fine \land (\forall f\ fine(int, f) \rightarrow f \geq inizio\}$$

### tasso_malattie(inizio: Data, fine: Data, M: Malattia [1..*]): (Malattia, Reale in 0..1)[1..*]

Precondizioni:

- inizio <= fine
- $|\{ sv \mid SoggettoVerde(sv) \land \exists\, e, es\ EpisodioMalattia(e) \land sv\_epi(sv, e) \land$
- $scoperta(e, es) \land es \leq fine \land (\forall\, ic, f\ InterventoConcluso(ic) \land fine(ic, f) \land$
- $\land\ risolve(ic, e) \rightarrow f \geq inizio)\}| > 0$

Postcondizioni

- L'operazione non modifica i dati
- Il risultato *result* è così definito

$$SM = \{ sv \mid SoggettoVerde(sv) \land \exists\, e, es, m\ EpisodioMalattia(e) \land sv\_epi(sv, e) \land \land\ ep\_malattia(e, m) \land scoperta(e, es) \land es \leq fine \land (\forall\, ic, f\ InterventoConcluso(ic) \land fine(ic, f) \land \land\ risolve(ic, e) \rightarrow f \geq inizio)\}$$

$$result = \{(m, z) \mid m \in M \land z = |\{sv \mid SoggettoVerde(sv) \land \exists\, e, es\ EpisodioMalattia(e) \land sv\_epi(sv, e) \land scoperta(e, es) \land ep\_malattia(e, m) \land es \leq fine \land (\forall\, ic, f\ InterventoConcluso(ic) \land fine(ic, f) \land risolve(ic, e) \rightarrow f \geq inizio)\}| / |SM| \}$$

Versione con operazione ausiliaria
Result = { (m, z/ |SM|) | m \in M AND z = num_sv_malati_periodo(inizio, fine, m) }

num_sv_malati_periodo(inizio: Data, fine: Data, m: Malattia): Reale in 0..1

Result = $|\{sv \mid SoggettoVerde(sv) \ \wedge \ \exists \, e, \, es \, EpisodioMalattia(e) \ \wedge sv\_epi(sv, \, e) \ \wedge scoperta(e, \, es) \ \wedge ep\_malattia(e, \, m) \wedge \ es \ \leq fine \ \wedge \ (\forall \, ic, \, f \, InterventoConcluso(ic) \ \wedge fine(ic, \, f) \ \wedge \ risolve(ic, \, e \,) \ \rightarrow \ f \ \geq \ inizio)\}|$

# 2. Progettazione della base dati

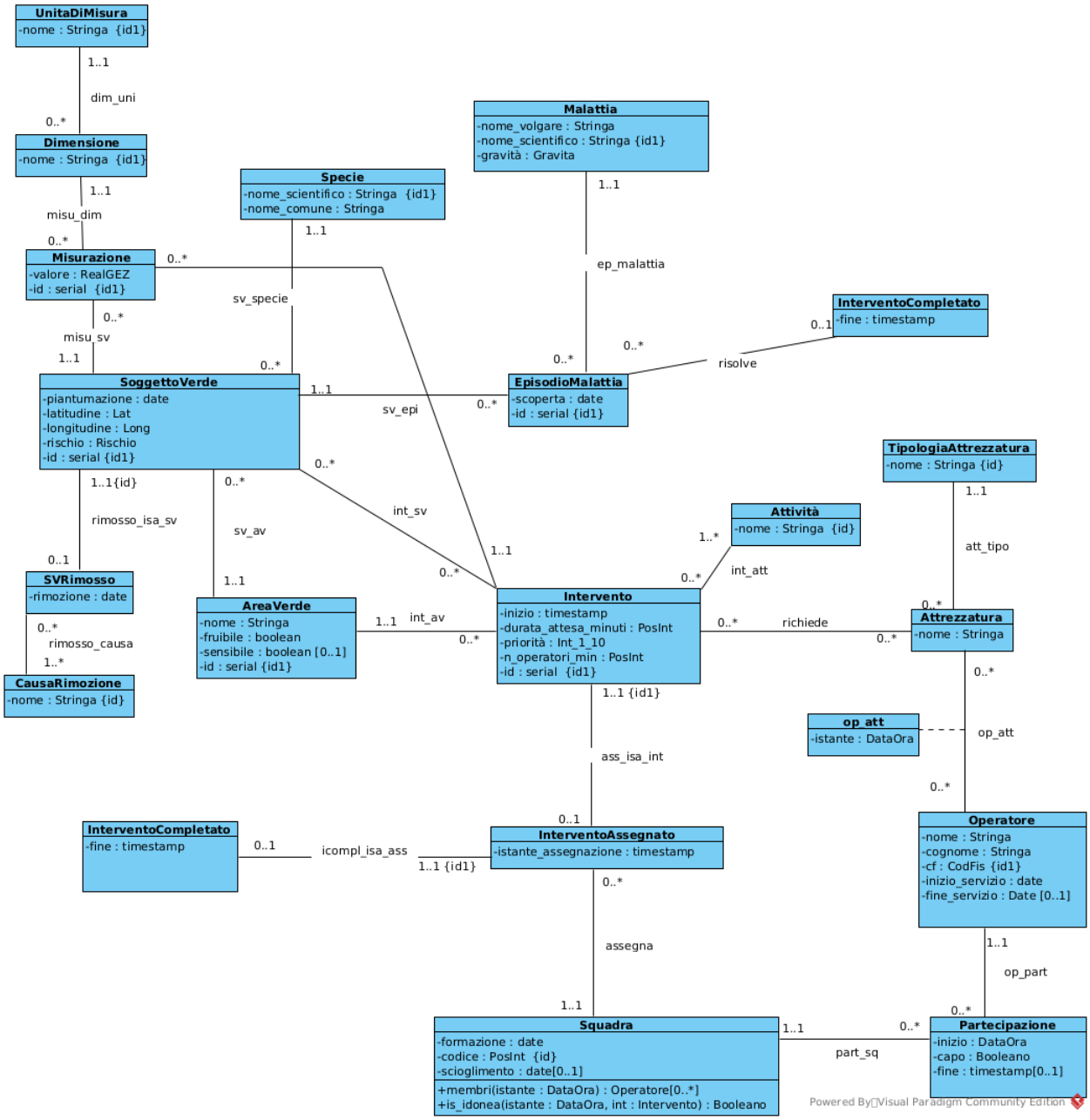
## 2.1 Specifica dei tipi di dato

```
CREATE DOMAIN Stringa AS varchar;
CREATE DOMAIN PosInt AS integer
     check (value >= 0);
CREATE DOMAIN RealGEZ AS real
     check (value >= 0);
CREATE DOMAIN CodFis AS Stringa
     check (value ~ '...');
CREATE DOMAIN Int_0_10 AS integer
     Check (value >= 0 and value <= 10);
CREATE DOMAIN Real_0_1 AS real
     Check (value >= 0 and value <= 1);
CREATE DOMAIN Lat AS real
     check (value >= -90 and value <= 90);
CREATE DOMAIN Long AS real
     check (value >= -180 and value <= 180);

CREATE TYPE Gravita AS ENUM
('Bassa', 'Media', 'Alta', 'Mortale');

CREATE TYPE Rischio AS ENUM
('Trascurabile', 'Basso', 'Medio', 'Alto');
```

# 2.2. Diagramma UML ristrutturato delle classi



### 2.2.1. Vincoli esterni introdotti in ristrutturazione

[V.AreaVerde.sensibile.solo_se_fruibile]

$\forall\ av\ AreaVerde(av)\ \rightarrow\ [\exists\ s\ sensibile(av,\ s)\ \leftrightarrow\ fruibile(av,\ True)]$

## 2.3. Schema relazionale (da completare)

```
CREATE DOMAIN Stringa AS varchar;
CREATE DOMAIN PosInt AS integer
    check (value >= 0);
CREATE DOMAIN RealGEZ AS real
    check (value >= 0);
CREATE DOMAIN CodFis AS char(16)
    check (value ~
'[A-Z]{6}[0-9]{6}[A-Z][0-9]{2}[A-Z][0-9]{3}[A-Z]');
CREATE DOMAIN Int_0_10 AS integer
    Check (value >= 0 and value <= 10);
CREATE DOMAIN Real_0_1 AS real
    Check (value >= 0 and value <= 1);
CREATE DOMAIN Lat AS real
    check (value >= -90 and value <= 90);
CREATE DOMAIN Longit AS real
    check (value >= -180 and value <= 180);

CREATE TYPE Gravita AS ENUM
('Bassa', 'Media', 'Alta', 'Mortale');

CREATE TYPE Rischio AS ENUM
('Trascurabile', 'Basso', 'Medio', 'Alto');


create table areaverde(
    id serial primary key,
    nome stringa not null,
    fruibile boolean not null,
    sensibile boolean,
    check (
        (sensibile is not null)
            = (fruibile = True)
    )

);

create table specie(
    nome_scientifico stringa primary key,
    nome_comune stringa not null
);
```

```
create table soggettoverde(
    id serial primary key,
    piantumazione date not null,
    latitudine lat not null,
    longitudine longit not null,
    rischio rischio not null,
    area integer not null,
    foreign key (area)
        references areaverde(id)
);

create table soggettoverderimosso (
    soggettoverde integer primary key,
    rimozione date not null,
    foreign key (soggettoverde)
        references soggettoverde(id)
    -- v. incl. (soggettoverde)
        -- occorre in rimosso_causa(soggettoverde)
);

create table causarimozione (
    nome stringa primary key
);


create table rimosso_causa (
    soggettoverde integer not null,
    causa stringa not null,
    foreign key (soggettoverde)
        references soggettoverderimosso(soggettoverde),
    foreign key (causa) references causarimozione(nome),
    primary key (soggettoverde, causa)
);




create table malattia (
    n_volgare stringa not null,
    n_scientifico stringa primary key,
    gravita gravita not null
);
```

```
create table attivita(
    nome stringa primary key
);

create table intervento(
    id serial primary key,
    inizio timestamp not null,
    durata_attesa_minuti PosInt not null,
    n_operatori_min PosInt not null
    -- v. incl. (id) occorre in int_att(intervento)
);

create table int_att (
    intervento integer not null,
    attivita stringa not null,
    foreign key (intervento)
        references intervento(id),
    foreign key (attivita)
        references Attivita(nome),
    primary key (intervento, attivita)
);


create table squadra (
    codice serial primary key,
    formazione date not null,
    scioglimento date,
    check (scioglimento is null
        or scioglimento >= formazione)
);

create table interventoassegnato (
    intervento integer primary key,

    foreign key (intervento)
        references intervento(id),
    squadra integer not null,
    foreign key (squadra)
        references squadra(codice),
    istante_assegnazione timestamp not null
);

create table interventoconcluso (
    intervento integer primary key,
```

```
    foreign key (intervento)
        references interventoassegnato(intervento),
    fine timestamp not null
    -- trigger fine >= istante_assegnazione
);

create table episodiomalattia(
     id serial primary key,
     scoperta date not null,
     malattia stringa not null,
     foreign key (malattia)
         references malattia(n_scientifico),
     soggetto integer not null,
     foreign key (soggetto)
         references soggettoverde(id),
     risolve integer,
     foreign key (risolve)
         references interventoconcluso(intervento)
);



create table tipologiaattrezzatura (
    nome stringa primary key
);

create table attrezzatura (
    id serial primary key,
    nome stringa not null,
    tipologia stringa not null,
    foreign key (tipologia)
        references tipologiaattrezzatura(nome)
);

create table richiede (
    intervento integer not null,
    attrezzatura integer not null,
    primary key (intervento, attrezzatura),
    foreign key    (attrezzatura)
        references attrezzatura(id),
    foreign key (intervento)
        references intervento(id)
);

create table operatore(
```

```
    cf CodFis primary key,
    nome stringa not null,
    cognome stringa not null,
    inizio_servizio date not null,
    fine_servizio date,
    check (fine_servizio is null
        or fine_servizio >= inizio_servizio)
);

create table op_att(
    operatore CodFis not null,
    attrezzatura integer not null,
    istante timestamp not null,
    primary key (operatore, attrezzatura),
    foreign key (operatore)
        references operatore(cf),
    foreign key    (attrezzatura)
        references attrezzatura(id)
);

create table partecipazione(
    id serial primary key,
    inizio timestamp not null,
    fine timestamp,
    check (fine is null or fine >= inizio),
    capo boolean not null,
    operatore CodFis not null,
    squadra integer not null,
    foreign key (operatore)
        references operatore(cf),
    foreign key (squadra)
        references squadra(codice)
);
```

## 2.4. Specifiche realizzative delle operazioni di Use-Case

squadre_idonee(int_id: integer): Set<Integer>

Sia Q il risultato della seguente query:

```
select sq.codice
from squadra sq, intervento i
where sq.scioglimento is null
    and i.id = 1
    and i.n_operatori_min <=ALL (
        select count(*)
        from partecipazione p
        where p.fine is null
            and p.squadra = sq.codice
        )
    and
    --  non esiste un'attrezzatura richiesta
    --  per la quale non esiste un operatore nella
    --  squadra che può utilizzarla
    not exists (
        select *
        from attrezzatura att, richiede r
        where r.intervento = i.id
            and r.attrezzatura = att.id
            and not exists (
                select *
                from operatore op,
                    partecipazione p,
                    op_att
                where p.fine is null
            and p.squadra = sq.codice
            and op.cf = op_att.operatore
            and op_att.attrezzatura = att.id
            and op_att.istante <= current_timestamp
            )
    );
```

tasso_malattie(inizio: date, fine: date, M: Set<varchar>): Set<varchar, Real_0_1>

```
CREATE TEMPORARY TABLE M_INPUT (nome Stringa);
```
*Il seguente ciclo FOR non è in SQL, è pseudocodice! Lo usiamo per inserire dati nella tabella temporanea dall'insieme preso in input.*

```
for each m_name in M do
      INSERT INTO M_INPUT (nome) values (m_name);
```

Sia SM il risultato della seguente query

```
SELECT DISTINCT sv.id

FROM soggettoverde sv,
      Malattia M,
      M_input m_in,
      episodiomalattia em
            LEFT OUTER JOIN
            interventoconcluso ic
                  ON em.risolve = ic.intervento

WHERE m.n_scientifico = m_in.nome
    and em.soggetto = sv.id
    and em.scoperta <= current_date + interval '1 day'
    and (em.risolve IS NULL
            or
            ic.fine >= current_date - interval '1 month'
      );
```

IF SM è vuota
Solleva eccezione *"Impossibile calcolare il tasso: nessun soggetto verde ha avuto episodi delle malattie indicate, nel periodo dato."*

*// Fine delle precondizioni*

```
WITH tot as (
      SELECT COUNT(*) as cnt
      FROM SM
)

SELECT m.n_scientifico, COUNT(DISTINCT sv.id) / tot.cnt
FROM tot,
      soggettoverde sv,
      Malattia m,
      M_input m_in,
      episodiomalattia em
            LEFT OUTER JOIN
            interventocompletato ic
                  ON em.risolve = ic.intervento

WHERE m.n_scientifico = m_in.nome
      and em.soggetto = sv.id
      and em.scoperta <= :fine
      and (
            em.risolve IS NULL
            or
            ic.fine >= :inizio)
      and em.malattia = m.n_scientifico
GROUP BY m.n_scientifico
```

// si può riscrivere, creando una tabella temporanea di tutti gli episodi di malattia di interesse (nel periodo e relativi ad una malattia in M)