PROGETTO ORTI BD

Parte I – Progettazione logica

Requisiti ristrutturati in modo da eliminare ambiguità

- Interpretazione di GRUPPO:
  Gruppo è un insieme di piante. Se queste sono di specie individuate con lo scopo di Biomonitoraggio, per ciascuna pianta "sporca" piantata, ce n'è una corrispondente "pulita" del gruppo di Controllo, sempre con scopo Biomonitoraggio, con la quale si confrontano i progressi;
- Scopo può essere Biomonitoraggio o Fitobonifica:
  - In Biomonitoraggio abbiamo tante piante "sporche" (stress) quante "pulite" (controllo)
  - In Fitobonifica abbiamo solo piante "pulite"
- "*Si considerano un certo numero di specie […] per i diversi scopi*", ovvero ogni specie può avere un solo scopo (Biomonitoraggio o Fitobonifica)
- "*in caso di biomonitoraggio le repliche del gruppo di controllo ("nel pulito") dovranno essere lo stesso numero di quelle del gruppo per cui vogliamo monitorare lo stress ambientale*": solo i gruppi di Stress ambientale hanno una relazione con un gruppo di Controllo di riferimento, ma non viceversa;
- Per "Replica" si intende una singola specifica pianta. All'interno dell'orto possono esserci tante repliche dello stesso tipo di pianta (Specie);
- La richiesta "*ogni scuola dovrebbe concentrarsi su tre specie e ogni gruppo dovrebbe contenere 20 repliche*" è stata interpretata nel seguente modo:
  - ogni scuola segue da 0 a 3
  - ogni gruppo può arrivare a contenere 20 repliche, ma è possibile che non tutte siano ancora state piantate
- Ogni gruppo è formato da più repliche, tutte della stessa specie;
- Il nome di ciascun orto è univoco tra tutti quelli inseriti nel sistema;
- Scuola e Istituto rappresentano lo stesso concetto/entità.

Progetto concettuale

Schema E/R

COMMENTI DI SUPPORTO

- La cardinalità (1, 2) sugli attributi di RILEVAZIONE indicano le misure sulla pianta di stress e, se presente, sulla corrispondente pianta di controllo (secondo l'allegato 2).
  Ogni rilevazione ha una pianta (quindi la cardinalità ha minimo 1) ma può avere anche la pianta di controllo (max 2);
- Lo 0 nel vincolo (0, n) in CLASSE – PIANTA rappresenta una classe che ha appena aderito al progetto ma non ha ancora piantato alcuna pianta;
- Nella relazione tra FINANZIAMENTO, SCUOLA e PERSONA, l'attributo Partecipante è un bool che indica se la persona specificata è solo referente o anche partecipante;
- Lo 0 nel vincolo (0, n) su SCUOLA – CLASSE rappresenta una possibile scuola che ha vinto un finanziamento per il progetto ma non ha ancora l'elenco delle classi partecipanti;
- Una Persona inserita nella base dati lavora in almeno una Scuola, ma può avere contratti con anche più di una: da qui cardinalità (1, n);
- Gli attributi in RILEVATORE sono Boolean perché indicano la capacità dello strumento di misurare tali caratteristiche;
- Gli attributi *opzione1_ opzione2* sono da interpretare come Boolean che se true vale nome1, altrimenti nome2;

- Un gruppo di piante di biomonitoraggio può essere senza gruppo di controllo poiché piantato per essere esso stesso di controllo per qualcun altro;
- Le misure del peso secco/fresco sono messe come opzionali (0, 1) poiché non può essere entrambi contemporaneamente.

Per lo schema, vedere allegato 1 (pg 5).

Documentazione relativa ai domini degli attributi (dizionario dati ed entità)

PERSONA	
Nome	Varchar(255)
Cognome	Varchar(255)
Email	Varchar(100)
Cellulare	Varchar(16)
Ruolo	Varchar(30)

SCUOLA	
CodMecc	Numeric
Nome	Varchar(100)
Comune	Varchar(50)
Provincia	Varchar(50)
CicloIstr	Numeric

ORTO	
Nome	Varchar(255)
Coordinate	Varchar(255)
Campo_Vaso	Boolean
Condizioni	Varchar(255)
Superficie	Numeric

CLASSE	
Classe	Numeric
Sezione	Char(2)
Ordine	Numeric
Tipo	Varchar(100)

RILEVATORE	
NumeroSerie	Numeric
Luce	Boolean
Fertilità	Boolean
Temperatura	Boolean
Umidità	Boolean
App	Boolean
NSensoreOrto	Numeric

Arduino	
MicroSD	Boolean
Nome	Varchar(100)
Peso	Float
Voltaggio	Float

PIANTA	
NReplica	Numeric
Esposizione	Varchar(100)
DataMessaADimora	datetime
Terriccio_Suolo	Boolean

GRUPPO	
Id	Numeric
Controllo_Stress	Boolean

Sensore	
Tipo	Numeric

RILEVAZIONE		
Id		Numeric
Num		Numeric
DataRilevazione		datetime
DataInserimento		datetime
Note		Varchar(255)
App		Boolean
Danni	*NFoglieDann*	Numeric
	NFruttiDann	Numeric
Fiori_Frutti	*PesoSecco*	Float
	PesoFresco	Float
	NFiori	Numeric
	NFrutti	Numeric
Biomassa	*LarghezzaFoglia*	Float
	LunghezzaFoglia	Float
	PesoFresco	Float
	PesoSecco	Float
	AltezzaPianta	Float
	LunghezzaRadice	Float
Suolo	*Ph*	Float
	Temperatura	Float
	Umidità	Float

SPECIE	
NomeScientifico	Varchar(255)
Nome	Varchar(100)
Esposizione	Varchar(100)
Biomonitoraggio_Fitobonifica	Boolean

Finanziamento	
Id	Numeric
Tipo	Varchar(255)
Valore	Float

Vincoli non esprimibili nel diagramma

- Ogni scuola deve piantare piante appartenenti alle sole massimo 3 specie legate a tale scuola (trigger);
- Ogni rilevazione ha un responsabile di inserimento e, se diverso, uno della rilevazione. Tale responsabile può essere un'intera classe o un singolo individuo, ma o uno o l'altro (trigger);
- Ruolo in PERSONA può essere uno tra: 'docente, 'dirigente', 'animatore digitale', 'referente', 'segretario', 'finanziatore', 'consulente', 'bidello' (check);
- Solo le istanze PERSONA con l'attributo ruolo = docente possono avere una relazione con CLASSE in quanto rappresentante (trigger). Inoltre, il docente rappresentante di una classe x deve essere dipendente della scuola x (trigger);
- In RILEVAZIONE:
  - DataInserimento >= DataRilevazione (check)
  - DataRilevazione >= Pianta.DataMessaADimora (trigger)
- Le piante del gruppo di controllo devono essere state piantate con lo scopo di Biomonitoraggio (trigger);
- Se una RILEVAZIONE è stata inserita tramite App, controllare che il Rilevatore che l'ha inserito abbia la modalità App attiva (trigger);
- In RILEVAZIONE, possono essere rilevate "contemporaneamente" due piante: una "effettiva" e una seconda di controllo, in caso di biomonitoraggio. Bisogna effettuare diversi controlli (trigger) nel caso venga inserita la pianta di controllo:
  - la pianta di controllo deve essere della stessa specie dell'altra inserita per prima e tale specie deve essere stata piantata con scopo di biomonitoraggio
  - la pianta di controllo deve far parte del gruppo di controllo per la prima pianta

Specifica dei tipi di gerarchie di generalizzazione

Nel nostro schema concettuale è presente una gerarchia:

1. *Arduino* e *Sensori* di *RILEVATORE*
   Gerarchia di tipo *totale* ed *esclusiva* (t,e): totale poiché ogni rilevatore è o di un tipo o dell'altro, esclusiva poiché non può essere di entrambi contemporaneamente.

Progetto logico

Schema E/R *ristrutturato*

Vedere allegato 2 (pg 6).

Eventuali modifiche dei domini degli attributi e informazioni sui domini di eventuali attributi introdotti

- L'attributo Num di RILEVAZIONE ha come valori possibili {1, 2}: se è 1 i dati inseriti fanno riferimento alla pianta su cui si è effettuata la rilevazione, se è 2 indica la pianta di controllo su cui si fanno i confronti (check).
  Non può esistere un dato con 2 se non esiste già la stessa chiave con 1 all'Id (trigger).

Modifiche all'elenco di vincoli del modello concettuale (nuovi vincoli, eventuali vincoli eliminati o modificati)

- Non possono esistere due istanze di cui una appartenente a Sensore e una appartenente ad Arduino con lo stesso numero di serie. Inoltre, la pk in Sensore e in Arduino composta da {NSensoreOrto, CodMecc, Orto.Nome} non può avere due valori uguali, anche se uno è in Sensore e l'altro in Arduino (trigger);
- Ogni RILEVAZIONE può essere effettuata con un sensore o con un Arduino: non più di uno strumento per volta, però uno deve essere stato selezionato (check).

Documentazione relativa alle scelte fatte per eliminare le gerarchie di generalizzazione

Avendo su *RILEVATORE* una gerarchia di tipo totale ed esclusiva, è risultato ottimale eliminare l'entità padre *RILEVATORE* e inserire gli attributi di *RILEVATORE* in ciascuna delle due entità figlie

SENSORI e *ARDUINO*. Inoltre, ogni associazione a cui partecipava *RILEVATORE* (RILEVAZIONE – RILEVATORE, PIANTA – RILEVATORE) è stata sostituita con 2 nuove associazioni, una per *Sensore* (RILEVAZIONE – SENSORI, PIANTA – SENSORI) e una per *Arduino* (RILEVAZIONE – ARDUINO, PIANTA – ARDUINO).

Schema logico

- **PERSONA** (Cognome, Nome, Email, Cellulare$_o$, Ruolo)
- **SCUOLA** (CodMecc, Nome, Provincia, Comune, CicloIstr)
- **DIPENDENTI** (EmailID^{PERSONA}, CodMecc^{SCUOLA}) //Relazione tra Persona e Scuola
- **FINANZIAMENTO** (Id, Tipo, Valore)
- **RAPPRE_FINANZ** (CodMecc^{SCUOLA}, IdF^{FINANZIAMENTO}, EmailID^{PERSONA}, Partecipante) //Ex relazione tripla
- **CLASSE** (Classe, Sezione, Ordine, CodMecc^{SCUOLA}, Rappresentante^{PERSONA}, Tipo)
- **SPECIE** (NomeScientifico, Esposizione, Nome, Biomonitoraggio_Fitobonifica)
- **ORTO** (Nome, IstProprietario^{SCUOLA}, Coordinate, Campo_vaso, Condizioni, Superficie)
- **GRUPPO** (Id, Controllo_stress, GruppoControllo$_O^{GRUPPO}$, Specie^{SPECIE}, Orto^{ORTO})
- **UTILIZZI** (Nome^{ORTO}, Scuola^{SCUOLA})
- **PIANTA** (Nreplica, Specie^{SPECIE}, EsposizioneSpecifica, DataMessaDimora, Terriccio_suolo, Classe^{CLASSE}, Sezione^{CLASSE}, Ordine^{CLASSE}, CodMecc^{CLASSE}, Gruppo^{GRUPPO})
- **RILEVAZIONE** (Id, Num, DataRilevazione, Pianta^{PIANTA}, Specie^{PIANTA}, DataInserimento, Note$_o$, NSens$_o^{SENSORE}$, OrtoSens$_o^{SENSORE}$, NArduino$_o^{ARDUINO}$, OrtoArduino$_o^{ARDUINO}$, App)
- **RESP_RILEVAZIONE** (IdRil^{RILEVAZIONE}, NumRil^{RILEVAZIONE}, ClasseIns$_o^{CLASSE}$, SezioneIns$_o^{CLASSE}$, OrdineIns$_o^{CLASSE}$, CodMeccIns^{SCUOLA}, PersonaIns$_o^{PERSONA}$, ClasseRil$_o^{CLASSE}$, SezioneRil$_o^{CLASSE}$, OrdineRil$_o^{CLASSE}$, CodMeccRil^{SCUOLA}, PersonaRil$_o^{PERSONA}$)
- **DANNI** (Rilevazione^{RILEVAZIONE}, Id, NfoglieDann, NfruttiDann)
- **FIORI_FRUTTI** (Rilevazione^{RILEVAZIONE}, Id, PesoSecco$_o$, PesoFresco$_o$, Nfiori, Nfrutti)
- **SUOLO** (Rilevazione^{RILEVAZIONE}, Id, Temperatura, Umidità, Ph)
- **BIOMASSA** (Rilevazione^{RILEVAZIONE}, Id, LarghezzaFoglia, LunghezzaFoglia, PesoSecco$_o$, LunghezzaRadice, AltezzaPianta, PesoFresco$_o$)
- **ARDUINO** (Nsensore, Orto^{ORTO}, NumeroSerie, Fertilità, Temperatura, Umidità, App, Luce, Peso, MicroSD, Nome, Voltaggio, Pianta$_o^{PIANTA}$, Specie$_o^{PIANTA}$)
- **SENSORE** (Nsensore, Orto^{ORTO}, NumeroSerie, Fertilità, Temperatura, Umidità, App, Luce, Tipo, Pianta$_o^{PIANTA}$, Specie$_o^{PIANTA}$)

Verifica di qualità dello schema ed eventuali ottimizzazioni applicate tenendo in considerazione il carico di lavoro

Poiché tutte le informazioni rappresentate esplicitamente nello schema sono "minimali" e non possono essere derivate da altre informazioni possiamo dire che non sono presenti ridondanze: è necessario mantenere memorizzati tutti i dati presenti nello schema per evitare eventuali perdite di informazioni.
Analizzando il workload, dal punto di vista dell'efficienza, bisogna considerare che dovranno essere effettuate sia operazioni di interrogazione che di modifica: in particolare, quest'ultime verranno effettuate frequentemente a causa degli aggiornamenti ai dati raccolti dalle rilevazioni, i quali sono stati tradotti da attributi presenti nello schema ER a entità a sé stanti (essendo attributi composti e multi-valore).

Lo schema è normalizzato rispetto alla forma normale di Boyce Codd (e di conseguenza anche rispetto alla terza forma normale) poiché ogni entità ha una propria chiave, e tutti gli altri attributi dipendono dalla chiave: essendo la chiave univoca, avendo il valore di una chiave di un'istanza posso ricavare senza ambiguità tutti gli attributi dell'istanza di riferimento.

Ottimizzazione

Si è scelto di rimuovere la relazione "segue" tra SCUOLA e SPECIE. Questa serviva per controllare che ogni scuola seguisse non più di 3 specie. Tuttavia, in PIANTA possiamo risalire sia alla specie, poiché la sua pk fa parte della chiave primaria di questa entità, che alla scuola, considerato che la sua pk fa parte della chiave primaria di CLASSE, la quale si trova in PIANTA come chiave esterna.

Parte II – Realizzazione

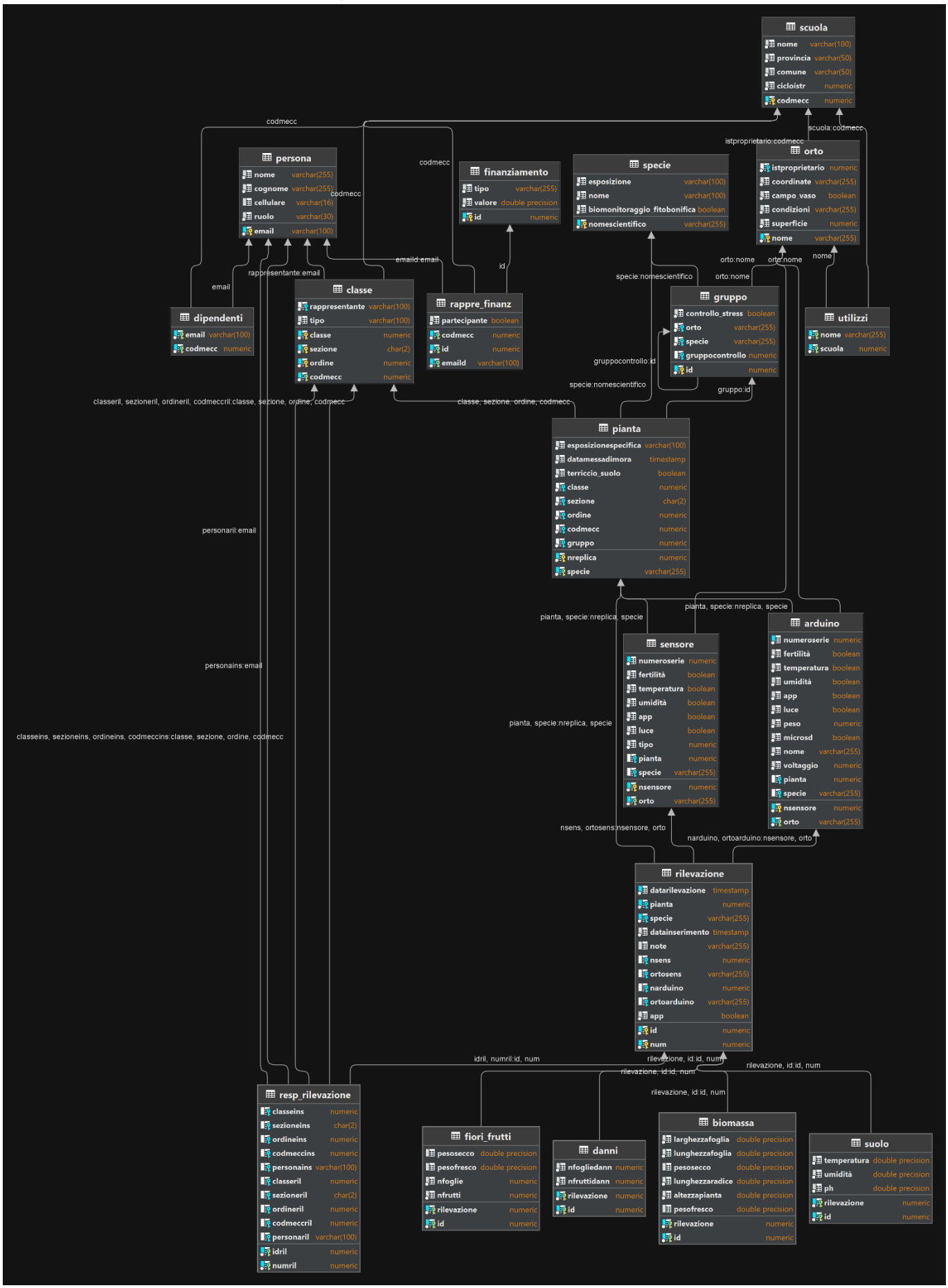
Script SQL per la creazione e popolazione dello schema

Creazione schema

L'implementazione dei vincoli individuati come *trigger* sarebbero implementabili come *check* ma richiederebbero una sottoquery e PostgreSQL non lo consente.

Popolazione schema

Diagramma dello script SQL



Interrogazione del database

Vista SQL

Si richiede la definizione di una vista che fornisca alcune informazioni riassuntive per ogni attività di biomonitoraggio: per ogni gruppo e per il corrispondente gruppo di controllo mostrare il numero di piante, la specie, l'orto in cui è posizionato il gruppo e, su base mensile, il valore medio dei parametri ambientali e di crescita delle piante (selezionare almeno tre parametri, quelli che si ritengono più significativi).

```
--"per ogni gruppo e per il corrispondente gruppo di controllo": per questo vengono
stampati solo i gruppi che hanno anche il gruppo di controllo (ovvero gruppo di stress)
CREATE VIEW orti.AttivitaBiomonitoraggio AS
    SELECT  GroupBio.Id AS "G", GroupBio.GruppoControllo AS "GC", GroupControl.Specie,
            GroupBio.N AS "N piante G", GroupControl.N AS "N piante GC",
            GroupBio.Orto as "Orto G", GroupControl.Orto as "Orto GC",
            GroupBio.ValLR AS "Lung Radice G", GroupControl.ValLR AS "Lung Radice GC",
            GroupBio.ValT AS "Temp G", GroupControl.ValT AS "Temp GC",
            GroupBio.ValPh AS "Ph G", GroupControl.ValPh AS "Ph GC"
    FROM (
        --INFO GRUPPO
        SELECT G.Id, G.GruppoControllo, G.Orto, T.N, T.ValLR, T.ValT, T.ValPh
        FROM orti.gruppo AS G
        JOIN (
            SELECT T1.Gruppo, N, ValLR, ValT, ValPh FROM (
                --per ogni gruppo quante piante
                SELECT Gruppo, COUNT(*) as N
                FROM orti.Pianta
                GROUP BY Gruppo
            ) AS T1 JOIN (
                --per ogni gruppo valori medi
                SELECT Gruppo, AVG(LunghezzaRadice) as ValLR, AVG(Temperatura) as ValT,
                 AVG(Ph) as ValPh
                FROM orti.Pianta as P
                JOIN orti.Rilevazione AS R ON R.Pianta = P.Nreplica AND R.Specie =
                 P.Specie --ignora se non ha mai subito rilevazioni
                JOIN orti.Suolo ON R.Id = Suolo.Rilevazione
                JOIN orti.Biomassa ON R.Id = Biomassa.Rilevazione
                GROUP BY Gruppo
            ) AS T2 ON T1.Gruppo = T2.Gruppo
        ) AS T ON G.Id = T.Gruppo   --prendo le info del gruppo di interesse
    ) AS GroupBio JOIN (
        --INFO GRUPPO DI CONTROLLO
        SELECT G.Id, G.GruppoControllo, G.Orto, G.Specie, T.N, T.ValLR, T.ValT, T.ValPh
        FROM orti.gruppo AS G
        JOIN (
            SELECT T1.Gruppo, N, ValLR, ValT, ValPh FROM (
                --per ogni gruppo quante piante
                SELECT Gruppo, COUNT(*) as N
                FROM orti.Pianta
                GROUP BY Gruppo
            ) AS T1 JOIN (
                --per ogni gruppo valori medi
                SELECT Gruppo, AVG(LunghezzaRadice) as ValLR, AVG(Temperatura) as ValT,
                 AVG(Ph) as ValPh
                FROM orti.Pianta as P
```

```
                JOIN orti.Rilevazione AS R ON R.Pianta = P.Nreplica AND R.Specie =
                 P.Specie --ignora se non ha mai subito rilevazioni
                JOIN orti.Suolo ON R.Id = Suolo.Rilevazione
                JOIN orti.Biomassa ON R.Id = Biomassa.Rilevazione
                GROUP BY Gruppo
            ) AS T2 ON T1.Gruppo = T2.Gruppo
        ) AS T ON G.GruppoControllo = T.Gruppo --prendo le info del gruppo di controllo
    ) AS GroupControl ON GroupBio.Id = GroupControl.Id  --unisco le due tabelle
    JOIN orti.specie AS S ON GroupControl.Specie = S.NomeScientifico
    WHERE Biomonitoraggio_Fitobonifica = true;  --biomonitoraggio
```

Test

```
select * from orti.AttivitaBiomonitoraggio;
```

G	GC	specie	N piante G	N piante GC	Orto G	Orto GC	Lung Radice G	Lung Radice GC	Temp G	Temp GC	Ph G	Ph GC
13	12	Mentha sp. pl.	3	3	Orto2	Orto2	13.79	14.03	22.25	24.56	6.35	6.77
20	19	Perlargonium sp. pl.	3	2	Orto2	Orto2	15	15	24.025	24.025	6.55	6.55
22	21	Helianthus annuus L.	4	3	Orto2	Orto2	15	15	24.95	24.95	6.55	6.55

Interrogazioni SQL

a. *Determinare le scuole che, pur avendo un finanziamento per il progetto, non hanno inserito rilevazioni in questo anno scolastico.*

```
SELECT CodMecc, Nome
FROM Orti.Scuola as S
WHERE EXISTS (
    SELECT CodMecc FROM Orti.Rappre_Finanz
    WHERE Rappre_Finanz.CodMecc = S.CodMecc)
AND NOT EXISTS (
    SELECT CodMeccRil FROM Orti.Resp_Rilevazione
    JOIN Orti.Rilevazione ON Rilevazione.Id = Resp_Rilevazione.IdRil
    WHERE S.CodMecc = Resp_Rilevazione.CodMeccRil AND
        EXTRACT (YEAR FROM (Rilevazione.DataRilevazione)) = EXTRACT (YEAR FROM
         (CURRENT_DATE))
)
```

Test

codmecc	nome
222	Scuola Secondaria ABC
333	Liceo Scientifico DEF
555	Scuola Primaria LMN

b. *Determinare le specie utilizzate in tutti i comuni in cui ci sono scuole aderenti al progetto.*

```
--solo scuole partecipanti al progetto sono state inserite nel db
SELECT Specie
FROM orti.Pianta
JOIN orti.Scuola ON Pianta.CodMecc = Scuola.CodMecc
GROUP BY Specie HAVING COUNT(DISTINCT Comune) >= ALL(
    SELECT COUNT(DISTINCT Comune)   --comuni partecipanti
    FROM orti.Scuola)
```

Test

specie
Calendula officinalis L.

c. *Determinare per ogni scuola l'individuo/la classe della scuola che ha effettuato più rilevazioni.*

```
--per come abbiamo organizzato noi le tabelle, è solo possibile fare le due richieste
  singolarmente, poiché la classe ha 4 campi mentre persona 1
SELECT *
FROM (  --CLASSE
    SELECT CodMeccRil AS Scuola, ClasseRil AS Classe, SezioneRil AS Sezione, OrdineRil
AS Ordine, COUNT(*) AS "N Ril Classe"
    FROM orti.Resp_Rilevazione AS X
```

```
    GROUP BY CodMeccRil, ClasseRil, SezioneRil, OrdineRil
    HAVING COUNT(*) >= (
        SELECT COUNT(*) as C
        FROM orti.Resp_Rilevazione
        WHERE CodMeccRil IS NOT NULL AND X.CodMeccRil = CodMeccRil
        GROUP BY CodMeccRil, ClasseRil, SezioneRil, OrdineRil
        ORDER BY C DESC
        LIMIT 1)
) AS T1 NATURAL JOIN ( --PERSONA
    SELECT CodMecc AS Scuola, PersonaRil, COUNT(*) AS "N Ril Persona"
    FROM orti.Resp_Rilevazione
    JOIN orti.Dipendenti AS X ON Email = PersonaRil
    GROUP BY CodMecc, PersonaRil
    HAVING COUNT(*) >=
        (SELECT COUNT(*) AS C
        FROM orti.Resp_Rilevazione
        JOIN orti.Dipendenti ON Email = PersonaRil
        WHERE PersonaRil IS NOT NULL AND X.CodMecc = CodMecc
        GROUP BY CodMecc, PersonaRil
        ORDER BY C DESC
        LIMIT 1)
    ) AS T2
ORDER BY Scuola;
```

Test

****

Procedure/funzioni SQL

a. *Funzione che realizza l'abbinamento tra gruppo e gruppo di controllo nel caso di operazioni di biomonitoraggio.*

```
CREATE OR REPLACE FUNCTION orti.changeGC (IN G NUMERIC, IN GC NUMERIC)
RETURNS VOID AS $$
BEGIN
    -- Controllo se esiste un record con gli Id specificati
    IF (EXISTS (SELECT 1 FROM orti.Gruppo WHERE Id = G) AND
        EXISTS (SELECT 1 FROM orti.Gruppo WHERE Id = GC)) THEN
        --Controllo siano della stessa specie
        IF ((SELECT Specie FROM orti.Gruppo WHERE Id = G) = (SELECT Specie FROM
         orti.Gruppo WHERE Id = GC)) THEN
            UPDATE orti.Gruppo
            SET GruppoControllo = GC
            WHERE Id = G;
        END IF;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Test

****

b. *Funzione che corrisponde alla seguente query parametrica: data una replica con finalità di fitobonifica e due date, determina i valori medi dei parametri rilevati per tale replica nel periodo compreso tra le due date.*

```
CREATE OR REPLACE FUNCTION orti.printFitoAvg (
    IN N NUMERIC,
```

```
    IN S VARCHAR(255),
    IN Data1 TIMESTAMP,
    IN Data2 TIMESTAMP
)
RETURNS TABLE (
    NFoglieDann NUMERIC,
    NFruttiDann NUMERIC,
    PesoSeccoFiori FLOAT,
    PesoFrescoFiori FLOAT,
    NfoglieFiori NUMERIC,
    NfruttiFiori NUMERIC,
    TemperaturaSuolo FLOAT,
    UmiditàSuolo FLOAT,
    PhSuolo FLOAT,
    LarghezzaFoglia FLOAT,
    LunghezzaFoglia FLOAT,
    PesoSeccoBiomassa FLOAT,
    LunghezzaRadice FLOAT,
    AltezzaPianta FLOAT,
    PesoFrescoBiomassa FLOAT
) AS $$
DECLARE
    C TIMESTAMP;
BEGIN
    -- Controllo validità input
    IF (
        EXISTS (SELECT 1 FROM orti.Pianta WHERE Nreplica = N AND Specie = S) -- pianta
                                                                              valida
        AND
        EXISTS (SELECT 1 FROM orti.Specie WHERE NomeScientifico = S AND
         Biomonitoraggio_Fitobonifica = false) -- specie di fitobonifica
    ) THEN
        IF (Data1 > Data2) THEN --riordino eventualmente le date
            C := Data1;
            Data1 := Data2;
            Data2 := C;
        END IF;

        IF((SELECT COUNT(*) FROM orti.Rilevazione WHERE Pianta = N AND Specie = S) <> 0)
THEN
            RETURN QUERY
            SELECT  AVG(D.NfoglieDann), AVG(D.NfruttiDann),
                    AVG(F.PesoSecco), AVG(F.PesoFresco), AVG(F.Nfoglie), AVG(F.Nfrutti),
                    AVG(Su.Temperatura), AVG(Su.Umidità), AVG(Su.Ph),
                    AVG(B.LarghezzaFoglia), AVG(B.LunghezzaFoglia), AVG(B.PesoSecco) ,
                    AVG(B.LunghezzaRadice), AVG(B.AltezzaPianta), AVG(B.PesoFresco)
            FROM orti.Danni AS D
            JOIN orti.Rilevazione AS R ON D.Rilevazione = R.Id AND D.Id = R.Num
            JOIN orti.Fiori_Frutti AS F ON F.Rilevazione = R.Id AND F.Id = R.Num
            JOIN orti.Suolo AS Su ON Su.Rilevazione = R.Id AND Su.Id = R.Num
            JOIN orti.Biomassa AS B ON B.Rilevazione = R.Id AND B.Id = R.Num
            WHERE Pianta = N AND Specie = S AND DataRilevazione BETWEEN Data1 AND Data2;
        END IF;
    END IF;
```

```
END;
$$ LANGUAGE plpgsql;
```

Test

```
SELECT * FROM orti.printFitoAvg(1, 'Cucumis satius L.', '2023-07-02 00:00:00', '2023-07-03 00:00:00');
SELECT * FROM orti.printFitoAvg(4, 'Helianthus annuus L.', '2023-07-02 00:00:00', '2023-07-03 00:00:00'); --vuota perché non fito
```

****

Trigger

a. *Verifica del vincolo che ogni scuola dovrebbe concentrarsi su tre specie e ogni gruppo dovrebbe contenere 20 repliche*.

```
--TRIGGER 1.a: Trigger per il controllo del vincolo "3 specie per scuola" (massimo 3)
CREATE FUNCTION Orti.verifica_numero_specie()
    RETURNS TRIGGER AS $$
DECLARE
    num_specie INTEGER;
BEGIN
    -- Conta il numero di specie associate alla scuola
    SELECT COUNT(DISTINCT Specie)
    INTO num_specie
    FROM Orti.Pianta
    WHERE CodMecc = NEW.CodMecc;

    IF num_specie > 3 THEN
        RAISE EXCEPTION 'Il numero di specie per la scuola supera il limite consentito
         (3)';
    END IF;

    RETURN NEW;
EXCEPTION
    WHEN OTHERS THEN
        -- Handle the exception here if needed
        RAISE;
END;
$$ LANGUAGE plpgsql;
-- Trigger per il controllo del vincolo sul numero di specie per scuola
CREATE TRIGGER check_numero_specie
    AFTER INSERT OR UPDATE ON Orti.Pianta
    FOR EACH ROW
    EXECUTE PROCEDURE Orti.verifica_numero_specie();
```

Test

****

```
--TRIGGER 1.b
-- Trigger per il controllo del vincolo "20 repliche per gruppo"
CREATE FUNCTION Orti.verifica_numero_repliche()
    RETURNS TRIGGER AS $$
    DECLARE
        num_repliche INTEGER;
    BEGIN
        -- Conta il numero di repliche associate al gruppo
        SELECT COUNT(*)
        INTO num_repliche
```

```
        FROM Orti.Pianta
        WHERE Gruppo = NEW.Gruppo;

        IF num_repliche > 20 THEN
            RAISE EXCEPTION 'Il numero di repliche per il gruppo supera il limite
consentito (20)';
            ROLLBACK;
        END IF;

        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;
-- Trigger per il controllo del vincolo sul numero di repliche per gruppo
CREATE TRIGGER check_numero_repliche
    AFTER INSERT OR UPDATE ON Orti.Pianta
    FOR EACH ROW
    EXECUTE PROCEDURE Orti.verifica_numero_repliche();
```

Test

****

b. *Generazione di un messaggio (o inserimento di una informazione di warning in qualche tabella) quando viene rilevato un valore decrescente per un parametro di biomassa.*

```
CREATE FUNCTION Orti.check_biomassa()
    RETURNS TRIGGER AS $$
DECLARE
    old_LarghezzaFoglia FLOAT;
    old_LunghezzaFoglia FLOAT;
    old_PesoSecco FLOAT;
    old_LunghezzaRadice FLOAT;
    old_AltezzaPianta FLOAT;
    old_PesoFresco FLOAT;
    replica_rilevata NUMERIC;
    specie_rilevata VARCHAR;
    dataora_rilevata TIMESTAMP;
BEGIN
    SELECT Pianta, Specie, DataRilevazione  INTO replica_rilevata, specie_rilevata,
dataora_rilevata
    FROM Orti.Rilevazione WHERE NEW.rilevazione = Rilevazione.id AND NEW.id =
Rilevazione.num;

    SELECT LarghezzaFoglia, LunghezzaFoglia, PesoSecco, LunghezzaRadice, AltezzaPianta,
PesoFresco
    INTO old_LarghezzaFoglia, old_LunghezzaFoglia, old_PesoSecco, old_LunghezzaRadice,
old_AltezzaPianta, old_PesoFresco
    FROM Orti.Biomassa JOIN Orti.Rilevazione ON Orti.Biomassa.Rilevazione =
Orti.Rilevazione.Id
    AND Orti.Biomassa.Id = Orti.Rilevazione.Num
    WHERE Orti.Rilevazione.DataRilevazione = (
        SELECT Datarilevazione FROM Orti.Rilevazione WHERE Datarilevazione <
dataora_rilevata
        ORDER BY (Datarilevazione) DESC LIMIT 1
    );

    IF NEW.PesoFresco < old_PesoFresco THEN
```

```
        RAISE EXCEPTION 'Attenzione: Valore decrescente di "Peso fresco" rilevato';
    END IF;

    IF NEW.AltezzaPianta < old_AltezzaPianta THEN
        RAISE EXCEPTION 'Attenzione: Valore decrescente di "Altezza pianta" rilevato';
    END IF;

    IF NEW.LunghezzaRadice < old_LunghezzaRadice THEN
        RAISE EXCEPTION 'Attenzione: Valore decrescente di "Lunghezza radice" rilevato';
    END IF;

    IF NEW.LunghezzaFoglia < old_LunghezzaFoglia THEN
        RAISE EXCEPTION 'Attenzione: Valore decrescente di "Lunghezza foglia" rilevato';
    END IF;

    IF NEW.LarghezzaFoglia < old_LarghezzaFoglia THEN
        RAISE EXCEPTION 'Attenzione: Valore decrescente di "Larghezza foglia" rilevato';
    END IF;

    IF NEW.PesoSecco < old_PesoSecco THEN
        RAISE EXCEPTION 'Attenzione: Valore decrescente di "Peso secco" rilevato';
    END IF;

    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER check_biomassa_trigger
BEFORE INSERT ON Orti.Biomassa
FOR EACH ROW
EXECUTE PROCEDURE Orti.check_biomassa();
```

Test

****

PROGETTO ORTI BD

Parte III – Progettazione fisica, elaborazione delle interrogazioni e controllo dell'accesso

Progetto fisico e sua validazione

Interrogazioni del carico di lavoro

1) Valori del suolo messi a confronto

```
SELECT  P.Rilevazione, T1 AS "Temp Pianta", T2 AS "Temp Confronto",
        U1 AS "Umidità Pianta", U2 AS "Umidità Confronto",
        P1 AS "Ph Pianta", P2 AS "Ph Confronto"
FROM (SELECT Rilevazione, Temperatura AS T1, Umidità AS U1, Ph AS P1
    FROM orti.Suolo
    WHERE Id = 1 AND Rilevazione IN (
        SELECT Rilevazione
        FROM orti.Suolo
        GROUP BY Rilevazione HAVING COUNT(*) = 2
    )
) AS P JOIN (
    SELECT Rilevazione, Temperatura AS T2, Umidità AS U2, Ph AS P2
    FROM orti.Suolo
    WHERE Id = 2
) AS PC ON P.Rilevazione = PC.Rilevazione
ORDER BY P.Rilevazione
```

2) Determinare il rappresentante della classe 1 A 1 111

```
SELECT Rappresentante
FROM orti.Classe
WHERE Classe = 1 AND Sezione = 'A' AND Ordine = 1 AND CodMecc = 111;
```

3) Numero di rilevazioni inserite per scuola

```
SELECT CodMeccIns, COUNT(*)
FROM orti.Resp_Rilevazione
WHERE CodMeccIns IS NOT NULL
GROUP BY CodMeccIns
ORDER BY CodMeccIns
```

Progetto fisico

1) Indice in Suolo su (Rilevazione, Id)
   Tipo: indice hash (non clusterizzato).
   Motivazione: questo indice è stato scelto perché le interrogazioni sembrano richiedere un confronto tra due diverse misurazioni di suolo (Id 1 e Id 2) effettuate sulla stessa rilevazione. L'indice hash su (Suolo.Rilevazione, Suolo.Id) aiuterà a velocizzare le operazioni di ricerca quando si accede ai dati relativi a una specifica rilevazione con entrambi gli Id 1 e Id 2. L'uso dell'indice hash permette di effettuare ricerche molto efficienti, poiché si basa su una funzione hash che mappa direttamente la combinazione di Rilevazione e Id ai blocchi di memoria contenenti i dati corrispondenti.
2) Indice in Classe su (CodMecc, Classe, Sezione, Ordine)
   Tipo: indice ordinato (clusterizzato).
   Motivazione: l'indice ordinato (clusterizzato) su queste colonne migliorerà le prestazioni delle query che coinvolgono la ricerca di classi specifiche in base a CodMecc, Classe,

Sezione e Ordine. Poiché queste colonne sono utilizzate nella condizione di ricerca nella query sopra, creando un indice ordinato su di esse, il motore di database sarà in grado di eseguire un'operazione di accesso diretto, riducendo il tempo di ricerca dei dati desiderati. Inoltre, poiché l'indice è clusterizzato, le righe corrispondenti ai valori indicizzati saranno fisicamente vicine l'una all'altra nel disco, migliorando ulteriormente le prestazioni del database durante la ricerca di dati correlati.

3) Indice in Resp_Rilevazione su (CodMeccIns)
Tipo: indice ad albero non clusterizzato.
Motivazione: l'indice ordinato ad albero su "CodMeccIns" aiuterà a migliorare le prestazioni della query, in quanto la query effettua un'operazione di raggruppamento (GROUP BY) e conta il numero di occorrenze per ciascun valore unico di "CodMeccIns". Quando si utilizza l'indice ordinato, il motore di database può accedere ai valori di "CodMeccIns" in modo più efficiente, evitando di scorrere l'intera tabella per eseguire l'aggregazione.
L'indice clusterizzato ordinerà fisicamente i dati della tabella sulla base del valore di "CodMeccIns", consentendo di accedere ai dati correlati in modo sequenziale, migliorando ulteriormente le prestazioni.
Tuttavia, è importante notare che l'utilizzo di un indice clusterizzato può influenzare le prestazioni delle operazioni di inserimento, aggiornamento e cancellazione dei dati, poiché i dati vengono organizzati fisicamente sulla base di "CodMeccIns". Quindi, è necessario bilanciare le esigenze delle interrogazioni con le operazioni di manipolazione dei dati per ottenere le migliori prestazioni complessive del database. Se le operazioni di manipolazione dei dati sono molto frequenti, è possibile valutare l'utilizzo di un indice ordinato non clusterizzato per "CodMeccIns" per evitare impatti negativi sulle prestazioni durante le operazioni di modifica dei dati.

Occupazione spaziale

Numero di tuple per tabella

relname	n_live_tup
dipendenti	78
danni	34
classe	100
resp_rilevazione	34
finanziamento	5
biomassa	34
suolo	34
rilevazione	40
orto	5
sensore	40
gruppo	22
pianta	68
utilizzi	10
scuola	5
rappre_finanz	9
fiori_frutti	34
specie	10
arduino	21
persona	38

Numero di blocchi per tabella

relname	table_size_blocks
dipendenti	32768
danni	32768
classe	65536
resp_rilevazione	32768
finanziamento	32768
biomassa	32768
suolo	32768
rilevazione	49152
orto	32768
sensore	49152
gruppo	32768
pianta	32768
utilizzi	32768
scuola	32768
rappre_finanz	32768
fiori_frutti	32768
specie	24576
arduino	49152
persona	32768

Piani di esecuzione

1) Query complete 00:00:00.801 → Query complete 00:00:00.430
QUERY PLAN PRIMA DELL'INDICE

```
Sort  (cost=40.58..40.59 rows=1 width=80)
  Sort Key: suolo.rilevazione
  ->  Nested Loop  (cost=20.80..40.57 rows=1 width=80)
        Join Filter: (suolo_2.rilevazione = suolo.rilevazione)
```

```
        ->  Nested Loop  (cost=20.65..34.39 rows=1 width=88)
              ->  HashAggregate  (cost=20.50..23.00 rows=1 width=32)
                    Group Key: suolo_2.rilevazione
                    Filter: (count(*) = 2)
                    ->  Seq Scan on suolo suolo_2  (cost=0.00..17.00 rows=700 width=32)
            ->  Index Scan using suolo_pkey on suolo suolo_1  (cost=0.15..8.17 rows=1 width=56)
                    Index Cond: ((rilevazione = suolo_2.rilevazione) AND (id = '2'::numeric))
        ->  Index Scan using suolo_pkey on suolo  (cost=0.15..6.17 rows=1 width=56)
              Index Cond: ((rilevazione = suolo_1.rilevazione) AND (id = '1'::numeric))
```

QUERY PLAN DOPO INSERIMENTO INDICE

```
Nested Loop  (cost=3.39..4.85 rows=1 width=80)
  Join Filter: (suolo.rilevazione = suolo_1.rilevazione)
  ->  Merge Join  (cost=3.39..3.42 rows=1 width=88)
        Merge Cond: (suolo.rilevazione = suolo_2.rilevazione)
        ->  Sort  (cost=1.44..1.44 rows=1 width=56)
              Sort Key: suolo.rilevazione
              ->  Seq Scan on suolo  (cost=0.00..1.43 rows=1 width=56)
                    Filter: (id = '1'::numeric)
        ->  Sort  (cost=1.96..1.96 rows=1 width=32)
              Sort Key: suolo_2.rilevazione
              ->  HashAggregate  (cost=1.51..1.94 rows=1 width=32)
                    Group Key: suolo_2.rilevazione
                    Filter: (count(*) = 2)
                    ->  Seq Scan on suolo suolo_2  (cost=0.00..1.34 rows=34 width=32)
  ->  Seq Scan on suolo suolo_1  (cost=0.00..1.43 rows=1 width=56)
        Filter: (id = '2'::numeric)
```

2) Query complete 00:00:00.173 → Query complete 00:00:00.123

QUERY PLAN PRIMA DELL'INSERIMENTO DELL'INDICE

```
Seq Scan on classe  (cost=0.00..4.00 rows=1 width=25)
  Filter: ((classe = '1'::numeric) AND (ordine = '1'::numeric) AND (sezione = 'A'::bpchar) AND (codmecc = '111'::numeric))
```

QUERY PLAN DOPO INSERIMENTO DELL'INDICE

```
Seq Scan on classe  (cost=0.00..4.00 rows=1 width=25)
  Filter: ((classe = '1'::numeric) AND (ordine = '1'::numeric) AND (sezione = 'A'::bpchar) AND (codmecc = '111'::numeric))
```

Si può notare come il piano di esecuzione non sia cambiato con la creazione dell'indice, poiché è comunque avvenuta una scansione sequenziale.

3) Query complete 00:00:00.313 → Query complete 00:00:00.166

QUERY PLAN PRIMA DELL'INSERIMENTO DELL'INDICE

```
GroupAggregate  (cost=14.32..16.07 rows=100 width=40)
  Group Key: codmeccins
  ->  Sort  (cost=14.32..14.57 rows=100 width=32)
        Sort Key: codmeccins
        ->  Seq Scan on resp_rilevazione  (cost=0.00..11.00 rows=100 width=32)
              Filter: (codmeccins IS NOT NULL)
```

QUERY PLAN DOPO INSERIMENTO INDICE

```
GroupAggregate  (cost=2.20..2.80 rows=34 width=40)
  Group Key: codmeccins
  ->  Sort  (cost=2.20..2.29 rows=34 width=32)
        Sort Key: codmeccins
        ->  Seq Scan on resp_rilevazione  (cost=0.00..1.34 rows=34 width=32)
              Filter: (codmeccins IS NOT NULL)
```

Implementazione SQL

```
1) CREATE INDEX idx_Suolo_Rilevazione_Id ON orti.Suolo(Rilevazione, Id);
2) CREATE INDEX idx_Classe_CodMecc_Classe_Sezione_Ordine ON orti.Classe(CodMecc,
   Classe, Sezione, Ordine);
   CLUSTER orti.Classe USING idx_Classe_CodMecc_Classe_Sezione_Ordine;
3) CREATE INDEX idx_Resp_Rilevazione_CodMeccIns ON orti.Resp_Rilevazione(CodMeccIns);
```

Controllo dell'accesso

Si richiede di definire i ruoli insegnante, gestore globale del progetto, referente di istituto, referente della scuola e studente (di una delle classi coinvolte nel progetto), individuando una opportuna gerarchia tra di essi.

Descrizione delle scelte implementative

Studente ≤ Insegnante ≤ RappreScuola ≤ RappreProgetto.
Ogni ruolo alla destra del '≤' ha tutti i permessi del ruolo alla sinistra, più eventualmente altri.

Si è scelto di garantire a Studente i permessi di solo inserimento e visualizzazione sulle tabelle delle rilevazioni poiché si vuole evitare che uno studente possa modificare/eliminare rilevazioni non inerenti alla propria classe. In caso di errori, può essere aiutato da un Insegnante che ha tutti i permessi su tali tabelle e per questo può eventualmente correggere le sviste.
Inoltre, Studente può visualizzare quale dei sensori si trova su quale pianta.

A Insegnante non sono stati permessi accessi alle entità "burocratiche" del progetto. Può modificare le Piante piantate e fare inserimenti e selezioni sui Gruppi, ma non può eliminare (per quello serve almeno il Rappresentante della Scuola).

RappreScuola ha tutti i privilegi su quasi tutto il DataBase, tranne sulle Entità che si riferiscono direttamente alla gestione del progetto.

Infine, il RappreProg ha tutti i privilegi sull'intera base di dati.

Implementazione SQL

```
CREATE ROLE Studente;
GRANT insert, select ON orti.Rilevazione, orti.Resp_Rilevazione, orti.Danni,
orti.Biomassa, orti.Suolo, orti.Fiori_Frutti TO Studente;
GRANT select ON orti.Arduino, orti.Sensore, orti.Pianta, orti.Specie TO Studente;

CREATE ROLE Insegnante;
GRANT Studente TO Insegnante;
GRANT ALL PRIVILEGES ON orti.Rilevazione, orti.Resp_Rilevazione, orti.Danni,
orti.Biomassa, orti.Suolo, orti.Fiori_Frutti TO Insegnante;
GRANT ALL PRIVILEGES ON orti.Pianta TO Insegnante;
GRANT insert, select ON orti.Gruppo TO Insegnante;
GRANT select ON orti.Classe TO Insegnante;

CREATE ROLE RappreScuola;
GRANT Insegnante TO RappreScuola;
GRANT ALL PRIVILEGES ON orti.Arduino, orti.Sensore, orti.Gruppo, orti.Orto,
orti.Utilizzi, orti.Classe, orti.Persona, orti.Dipendenti TO RappreScuola;
GRANT insert ON orti.Scuola, orti.Rappre_Finanz TO RappreScuola;

CREATE ROLE RappreProg;
GRANT RappreScuola TO RappreProg;
GRANT ALL PRIVILEGES ON SCHEMA orti TO RappreProg WITH GRANT OPTION;
```

```
CREATE USER mario;  GRANT Studente TO mario;
CREATE USER giulio; GRANT Insegnante TO giulio;
CREATE USER francesca;  GRANT RappreScuola TO francesca;
CREATE USER sara;   GRANT RappreProg TO sara;
CREATE USER marta;  GRANT RappreScuola TO marta;
```

Permessi riassunti

	Studente	Insegnante	RappreScuola	RappreProg
Persona	\	\	ALL PRIVILEGES	ALL PRIVILEGES
Scuola	\	\	Insert	ALL PRIVILEGES
Dipendenti	\	\	ALL PRIVILEGES	ALL PRIVILEGES
Finanziamento	\	\	\	ALL PRIVILEGES
Rappre_Finanz	\	\	insert	ALL PRIVILEGES
Classe	\	select	ALL PRIVILEGES	ALL PRIVILEGES
Specie	select	select	select	ALL PRIVILEGES
Orto	\	\	ALL PRIVILEGES	ALL PRIVILEGES
Gruppo	\	insert, select	ALL PRIVILEGES	ALL PRIVILEGES
Utilizzi	\	\	ALL PRIVILEGES	ALL PRIVILEGES
Pianta	select	ALL PRIVILEGES	ALL PRIVILEGES	ALL PRIVILEGES
Sensore	select	select	ALL PRIVILEGES	ALL PRIVILEGES
Arduino	select	select	ALL PRIVILEGES	ALL PRIVILEGES
Rilevazione	insert, select	ALL PRIVILEGES	ALL PRIVILEGES	ALL PRIVILEGES
Resp_Rilevazione	insert, select	ALL PRIVILEGES	ALL PRIVILEGES	ALL PRIVILEGES
Danni	insert, select	ALL PRIVILEGES	ALL PRIVILEGES	ALL PRIVILEGES
Fiori_Frutti	insert, select	ALL PRIVILEGES	ALL PRIVILEGES	ALL PRIVILEGES
Suolo	insert, select	ALL PRIVILEGES	ALL PRIVILEGES	ALL PRIVILEGES
Biomassa	insert, select	ALL PRIVILEGES	ALL PRIVILEGES	ALL PRIVILEGES