



Università di Pisa

Corso di Ingegneria Informatica

Documentazione del progetto per il Corso di Basi di Dati

Anno accademico 2015 - 2016

Studenti: Del Castello Diego, Di Lascio Teresa

INDICE

1. PROGETTAZIONE CONCETTUALE	4
- <i>DESCRIZIONE DELLE ENTITÀ SCELTE</i>	<i>4</i>
- <i>ANALISI DELLE CARDINALITÀ.....</i>	<i>11</i>
2. INDIVIDUAZIONE DI OPERAZIONI SUI DATI	19
3. ANALISI DELLE PRESTAZIONI DELLE OPERAZIONI.....	21
4. INTRODUZIONE DELLE RIDONDANZE.....	26
5. PROGETTAZIONE LOGICA	40
- <i>VINCOLI DI INTEGRITÀ</i>	<i>42</i>
- <i>TRIGGER SUI VINCOLI DI INTEGRITÀ</i>	<i>47</i>
6. TRADUZIONE DELLE CARDINALITA'	49
7. ANALISI DELLE DIPENDENZE FUNZIONALI E NORMALIZZAZIONE	66
8. IMPLEMENTAZIONE SU DBMS ORACLE MYSQL	76
9. FUNZIONALITA' DI BACK-END E DOCUMENTAZIONE DELLE SCELTE DI IMPLEMENTAZIONE.....	126
10. GESTIONE DELL'AREA ANALYTICS.....	130

Specifiche di progetto

Si desidera progettare un database relazionale su DBMS Oracle MySQL che gestisca i dati relativi al sistema informativo di una grande azienda nel settore del giardinaggio. L'azienda coltiva e vende piante di ogni tipo. Inoltre, progetta e gestisce la manutenzione di giardini e aree verdi in genere. L'oggetto della progettazione include anche alcune funzionalità di back-end per l'analisi dei dati e la gestione intelligente.

Fasi di progettazione

1. PROGETTAZIONE CONCETTUALE

- *Descrizione delle entità scelte*

Area Coltivazione

PIANTA:

Si tratta dell'entità più importante di tutto il sistema e vi sono memorizzate tutte le caratteristiche necessarie alla catalogazione di un esemplare all'interno del database. Si è ritenuto performante considerare l'attributo *CodicePianta* come identificatore univoco della stessa.

LUCE:

Si tratta dell'entità atta a memorizzare tutto ciò che caratterizza una particolare illuminazione richiesta da una pianta. Si è ritenuto performante considerare l'attributo *CodiceLuce* come identificatore univoco della stessa.

TERRENO:

Si tratta dell'entità atta a memorizzare le caratteristiche di ogni particolare terreno presente nell'azienda che viene identificato univocamente tramite l'attributo *CodiceTerreno* all'interno del database.

ELEMENTO DISCIOLTO:

Si tratta dell'entità atta a memorizzare i dati concernenti ogni tipo di elemento rilasciato all'interno di un terreno, evidenziandone le caratteristiche principali. Si è ritenuto performante considerare l'attributo *CodiceElementoDisciolto* come identificatore univoco della stessa.

IRRIGAZIONE:

Si tratta dell'entità atta a memorizzare i dati relativi al fabbisogno idrico dei vari tipi di piante. Si è ritenuto performante considerare l'attributo *CodiceIrrigazione* come identificatore univoco della stessa.

CONCIMAZIONE:

Si tratta dell'entità atta a memorizzare i dati relativi agli interventi di concimazione necessari per lo sviluppo delle piante. Si è ritenuto performante considerare l'attributo *CodiceConcimazione* come identificatore univoco della stessa.

RINVASO:

Si tratta dell'entità atta a memorizzare i dati relativi alla procedura di rinvaso. Si è ritenuto performante considerare l'attributo *CodiceRinvaso* come identificatore univoco della stessa.

POTATURA:

Si tratta dell'entità atta a memorizzare i dati relativi al tipo di intervento di potatura e alla cadenza periodica in cui è necessario che venga effettuata. Si è ritenuto performante considerare l'attributo *CodicePotatura* come identificatore univoco della stessa.

PATOLOGIA:

Si tratta dell'entità atta a memorizzare tutte le possibili patologie che possono essere contratte dalle piante presenti nel database. Si è ritenuto performante considerare l'attributo *CodicePatologia* come identificatore univoco della stessa.

SINTOMO:

Si tratta dell'entità atta a memorizzare tutte le forme di manifestazione delle patologie che possono attaccare un esemplare. Si è ritenuto performante considerare l'attributo *CodiceSintomo* come identificatore univoco della stessa.

AGENTE_PATOGENO:

Si tratta dell'entità atta a memorizzare tutto quanto riguarda un agente patogeno che può cagionare problemi di salute ad un esemplare. Si è ritenuto performante considerare l'attributo *CodiceAgentePatogeno* come identificatore univoco della stessa.

IMMAGINE:

Si tratta dell'entità atta a memorizzare i dati delle foto identificative dello stato patologico di una pianta affetta da una qualche patologia, tramite manifestazione sintomatica di tipo fenotipico. Si è ritenuto performante considerare l'attributo *CodiceImmagine* come identificatore univoco della stessa.

CONTENITORE:

Si tratta dell'entità atta a memorizzare il vaso in cui è posta una pianta. Una pianta può anche essere posta in un terreno aperto, pertanto si è deciso di gestire questa possibilità introducendo l'attributo *TerrenoAperto* nella tabella *PIANTA* in cui si specifica il tipo di posizionamento della stessa. Si è ritenuto performante considerare l'attributo *CodiceContenitore* come identificatore univoco della stessa.

RIPIANO:

Si tratta dell'entità atta a memorizzare il ripiano in cui viene ubicato il vaso di una pianta. Si è ritenuto performante considerare l'attributo *CodiceRipiano* come identificatore univoco della stessa.

SEZIONE:

Si tratta dell'entità atta a memorizzare informazioni riguardo il posizionamento all'interno della serra. Si è ritenuto performante considerare l'attributo *CodiceRipiano* come identificatore univoco della stessa.

SERRA:

Si tratta dell'entità atta a memorizzare informazioni concernenti la serra in cui è ubicata la pianta. Si è ritenuto performante considerare l'attributo *CodiceSerra* come identificatore univoco della stessa.

SEDE:

Si tratta dell'entità atta a memorizzare tutto ciò che necessario a proposito della sede di appartenenza di tutte le piante, in particolar modo la locazione visto che possono essere dislocate su tutto il territorio nazionale. Si è ritenuto performante considerare l'attributo *CodiceSede* come identificatore univoco della stessa.

ACCRESIMENTO:

Si tratta dell'entità atta a memorizzare i dati tecnici necessari per creare un prospetto della crescita totale di un esemplare in un dato periodo di tempo. Si è ritenuto performante considerare l'attributo *CodiceAccrescimento* come identificatore univoco della stessa.

FARMACO:

Si tratta dell'entità atta a memorizzare tutte le informazioni necessarie alla catalogazione di un prodotto utile alla cura di una particolare patologia causata da un agente patogeno. Si è ritenuto performante considerare l'attributo *CodiceFarmaco* come identificatore univoco della stessa.

CURA:

Si tratta dell'entità che memorizza tutto ciò che riguarda la cura di un particolare problema di una pianta e viene creata in seguito alla formulazione di una diagnosi dopo la rivelazione di quest'ultima tramite appositi sensori capaci di monitorare la salute della pianta. Si è ritenuto performante considerare l'attributo *CodiceCura* come identificatore univoco della stessa.

SINTOMATOLOGIA:

Si tratta dell'entità che memorizza l'insieme di alcuni sintomi rilevati su una pianta che permettono il riscontro della sofferenza di una particolare patologia. Si è ritenuto performante considerare l'attributo *CodiceSintomatologia* come identificatore univoco della stessa.

INTERVENTO:

Si tratta dell'entità atta a memorizzare ogni tipo di atto manutentivo avvenuto e contemporaneamente a tenere fissata la prossima data consigliata, in base alle esigenze della pianta, per un intervento dello stesso tipo. Si è ritenuto performante considerare l'attributo *CodiceIntervento* come identificatore univoco della stessa.

FABBISOGNO:

Si tratta dell'entità atta a memorizzare tutti gli interventi di cui una pianta ha bisogno e la data consigliata in cui è necessario effettuarli. Si è ritenuto performante considerare l'attributo *CodiceFabbisogno* come identificatore univoco della stessa.

PERIODO:

Si tratta dell'entità usata per tenere conto dei periodi in cui si verificano o variano le varie caratteristiche ed esigenze di una pianta. Si è ritenuto performante considerare l'attributo *CodicePeriodo* come identificatore univoco della stessa.

Area Store e Social

ORDINE:

Si tratta dell'entità atta a memorizzare le informazioni concernenti ogni ordine avviato tramite il sito web dell'azienda da parte di un cliente. Si è ritenuto performante considerare l'attributo *CodiceOrdine* come identificatore univoco della stessa.

SCHEDA:

Si tratta dell'entità atta a memorizzare dati a proposito di ogni pianta quando la procedura d'acquisto da parte di un cliente è andata a buon fine, oppure per archiviare tutti i possibili tipi di richiesta di manutenzione che sono state necessarie per ogni esemplare posseduto dal cliente a cui si riferiscono. Si è ritenuto performante considerare l'attributo *CodiceScheda* come identificatore univoco della stessa.

CLIENTE:

Si tratta dell'entità atta a memorizzare le informazioni personali di ogni cliente dell'azienda in modo da avere un profilo per ognuno di essi. Si è ritenuto performante considerare l'attributo *CodiceCliente* come identificatore univoco della stessa.

POST:

Si tratta dell'entità che mantiene le informazioni di ogni post pubblicato nell'area social del sito web dell'azienda. Si è ritenuto performante considerare l'attributo *CodicePost* come identificatore univoco della stessa.

COMMENTO DI RISPOSTA:

Si tratta dell'entità che memorizza tutto ciò che riguarda ogni commento di risposta postato da un utente diverso come risposta ad un post precedentemente pubblicato. Si è ritenuto performante considerare l'attributo *CodiceCommentoRisposta* come identificatore univoco della stessa.

VALUTAZIONE:

Si tratta dell'entità che conserva le informazioni relative ad ogni valutazione assegnata da un utente ad un post. Si è ritenuto performante considerare l'attributo *CodiceValutazione* come identificatore univoco della stessa.

Area Progettazione e Garden Design

GIARDINO:

Si tratta dell'entità atta a memorizzare tutte le informazioni che i clienti forniscono circa i settori da progettare dei loro spazi verdi, in particolare si memorizzano, tramite coordinate, le disposizioni decise dal cliente. Si è ritenuto performante considerare l'attributo *CodiceGiardino* come identificatore univoco della stessa.

VASO:

Si tratta dell'entità atta a memorizzare ogni singolo vaso dell'area verde del cliente, tramite i dati che servono per la sua rappresentazione visiva. Si è ritenuto performante considerare l'attributo *CodiceVaso* come identificatore univoco della stessa.

SETTORE:

Si tratta dell'entità atta a memorizzare le informazioni concernenti ogni settore del giardino. Si è ritenuto performante considerare l'attributo *CodiceSettore* come identificatore univoco della stessa.

Area Manutenzione

MANUTENZIONE PRIVATA:

Si tratta dell'entità atta a memorizzare le informazioni riguardanti l'esecuzione di un atto manutentivo eseguito autonomamente da parte del cliente possessore della pianta. Si è ritenuto performante considerare l'attributo *CodiceManutenzionePrivata* come identificatore univoco della stessa.

MANUTENZIONE PROGRAMMATA:

Si tratta dell'entità atta a memorizzare le informazioni riguardanti il palesarsi di una necessità di un intervento di manutenzione in base alle esigenze espresse dal cliente. Si è ritenuto performante considerare l'attributo *CodiceManutenzioneProgrammata* come identificatore univoco della stessa.

MANUTENZIONE SU RICHIESTA:

Si tratta dell'entità atta a memorizzare le informazioni riguardanti la presenza di richieste di interventi di manutenzione straordinaria da parte del cliente. Si è ritenuto performante considerare l'attributo *CodiceManutenzioneSuRichiesta* come identificatore univoco della stessa.

- *Analisi delle cardinalità*

Nome Relazione	Entità	Cardinalità	Descrizione
Collocazione	PIANTA	(0 , 1)	Una pianta può essere posizionata al massimo in un contenitore altrimenti può trovarsi direttamente nel terreno e non essere associata a nessuno di essi.
Collocazione	CONTENITORE	(0 , 1)	Un contenitore o è vuoto (perché la pianta può essere spostata in quarantena o in seguito a un rinvaso) oppure è sempre associato a una sola pianta.
Dimora	CONTENITORE	(1 , 1)	Un contenitore è sempre associato a un solo ripiano.
Dimora	RIPIANO	(0 , N)	Un ripiano può non sostenere alcun contenitore, o un numero N di contenitori al massimo.
Ubicazione	SEZIONE	(1 , N)	Una sezione è composta da almeno un ripiano ma ne può avere un numero massimo dipendente dalla capienza della stessa (anche le piante poste in terreno aperto fanno parte di una sezione).
Ubicazione	RIPIANO	(1 , 1)	Un ripiano può essere presente in una sola sezione.
Disposizione	SERRA	(1 , N)	Una serra è composta da almeno una sezione e può averne un numero massimo che dipende dalla dimensione.
Disposizione	SEZIONE	(1 , 1)	Una sezione è presente in una sola serra.
Dislocazione	SEDE	(1 , N)	Una sede ha almeno una serra e può averne un numero massimo dipendente dalle dimensioni della stessa.

Dislocazione	SERRA	(1 , 1)	Ogni serra può essere ubicata in una sola sede.
Sistemazione	PIANTA	(0 , 1)	Ogni pianta viene sistemata in un'unica sezione, ma nel momento in cui viene acquistata non è sistemata in nessuna di esse.
Sistemazione	SEZIONE	(0 , N)	Ogni sezione può essere vuota o avere un numero qualsiasi di piante.
Manifestazione	PIANTA	(0 , N)	Ogni pianta può non aver mai manifestato alcuna sintomatologia, oppure averne riscontrata anche più di una.
Manifestazione	SINTOMATOLOGIA	(1 , N)	Ogni sintomatologia può essere riscontrata in un numero illimitato di piante, ma viene aggiunta al database appena si manifesta per un esemplare.
Appartenenza	SINTOMATOLOGIA	(1 , N)	Ogni sintomatologia è composta da almeno un sintomo ma può averne anche più di uno associato.
Appartenenza	SINTOMO	(1 , N)	Ogni sintomo è associato a una o più sintomatologie.
Descrizione	IMMAGINE	(1 , 1)	Ogni immagine si riferisce ad un solo sintomo.
Descrizione	SINTOMO	(1 , N)	Ogni sintomo ha almeno un'immagine ma può averne anche di più.
Esordio	SINTOMATOLOGIA	(1 , N)	Ogni sintomatologia può contraddistinguere più patologie, ma almeno di una.
Esordio	PATOLOGIA	(1 , N)	Ogni patologia può essere contraddistinta da una o più sintomatologie.
Trattamento	FARMACO	(1 , N)	Ogni farmaco può combattere più di un agente patogeno, ma ne combatte di sicuro almeno uno.
Trattamento	AGENTE PATOGENO	(1 , N)	Ogni agente patogeno può essere combattuto da più di un farmaco, ma è combattuto di sicuro da almeno uno.
Individuazione	PATOLOGIA	(1 , N)	Ogni patologia è causata da almeno un agente patogeno, ma può essere causata da più di uno.
Individuazione	AGENTE PATOGENO	(1 , N)	Ogni agente patogeno provoca almeno una patologia, ma può provocarne più di una.

Prescrizione	FARMACO	(1 , N)	Ogni farmaco può essere prescritto in più cure, ma di sicuro almeno in una.
Prescrizione	CURA	(1 , 1)	Ogni cura prevede la somministrazione di un solo farmaco.
Quarantena	PIANTA	(0 , N)	Ogni pianta può essere stata sottoposta a più cure ma, se non si è ancora ammalata, a nessuna.
Quarantena	CURA	(1 , N)	Per ogni pianta che si è ammalata è presente una cura ma è possibile che se ne trovino diverse associate allo stesso esemplare.
Periodo Fioritura	PERIODO	(1 , N)	Un periodo di fioritura può essere associato a più piante.
Periodo Fioritura	PIANTA	(1 , N)	Ogni pianta può avere più periodi di fioritura.
Periodo Fruttificazione	PERIODO	(1 , N)	Ogni periodo di fruttificazione può caratterizzare più piante.
Periodo Fruttificazione	PIANTA	(1 , N)	Ogni pianta può avere più periodi di fruttificazione.
Suolo	PIANTA	(1 , 1)	Una pianta può necessitare di un solo tipo di terreno.
Suolo	TERRENO	(1 , N)	Uno stesso tipo di terreno può essere necessario per più piante presenti.
Presenza	ELEMENTO DISCIOLTO	(1 , N)	Uno stesso elemento disciolto può essere presente in più terreni diversi.
Presenza	TERRENO	(1 , N)	Un terreno può presentare più elementi disciolti.
Bisogno	LUCE	(1 , N)	Uno stesso tipo di luce può essere necessario a più piante.
Bisogno	PIANTA	(1 , 1)	Una pianta può avere bisogno di un solo tipo di illuminazione.

Necessità	PIANTA	(1 , 1)	Ogni pianta ha un solo specifico piano idrico.
Necessità	IRRIGAZIONE	(1 , N)	Uno stesso tipo di irrigazione può essere necessario per più piante.
Periodicità Irrigatoria	IRRIGAZIONE	(1 , N)	Un'irrigazione è sempre caratterizzata dall'aver sempre almeno un periodo.
Periodicità Irrigatoria	PERIODO	(0 , N)	Un periodo potrebbe non riferirsi ad alcuna irrigazione ma caratterizzare altre necessità.
Occorrenza	INTERVENTO	(1 , N)	Uno stesso intervento può essere eseguito su più di una pianta.
Occorrenza	PIANTA	(0 , N)	La stessa pianta può avere subito più di un intervento o non averne subito affatto.
Esigenza	FABBISOGNO	(1 , N)	Lo stesso fabbisogno potrebbe essere comune a più piante.
Esigenza	PIANTA	(0 , N)	Una pianta potrebbe avere avuto più volte bisogno di un intervento oppure non averne mai avuto bisogno.
Esecuzione di concimazione	CONCIMAZIONE	(0 , N)	E' possibile che non sia ancora presente nessuna concimazione.
Esecuzione di concimazione	INTERVENTO	(0 , N)	Può verificarsi il caso in cui per una data pianta non è stato ancora effettuato nessun intervento di concimazione, oppure ne sono stati necessari diversi.
Esecuzione di potatura	POTATURA	(0 , N)	E' possibile che non sia ancora presente nessuna potatura.
Esecuzione di potatura	INTERVENTO	(0 , N)	Può verificarsi il caso in cui per una data pianta non è stato ancora effettuato nessun intervento di potatura, oppure ne sono stati necessari diversi.
Esecuzione di rinvaso	RINVASO	(0 , N)	E' possibile che non sia ancora presente nessun rinvaso.
Esecuzione di rinvaso	INTERVENTO	(0 , N)	Può verificarsi il caso in cui per una data pianta non è stato ancora effettuato nessun intervento di rinvaso, oppure ne sono stati necessari diversi.

Necessità di concimazione	CONCIMAZIONE	(1 , N)	Una pianta può avere necessità di almeno un tipo di concimazione, ma quasi sempre è più di uno che varia in base al periodo dell'anno.
Necessità di concimazione	FABBISOGNO	(0 , N)	Può verificarsi il caso in cui per una data pianta non è stato ancora riscontrato un effettivo bisogno di concimazione, oppure ne sono stati già previsti diversi.
Necessità di potatura	POTATURA	(1 , N)	Una pianta può avere necessità di almeno un tipo di potatura, ma quasi sempre è più di uno che varia in base al periodo dell'anno.
Necessità di potatura	FABBISOGNO	(0 , N)	Può verificarsi il caso in cui per una data pianta non è stato ancora riscontrato un effettivo bisogno di una potatura, oppure ne sono stati già previsti diversi.
Necessità di rinvaso	RINVASO	(1 , N)	Una pianta può avere necessità di almeno un rinvaso, ma quasi sempre è più di uno che varia in base allo stato dell'esemplare.
Necessità di rinvaso	FABBISOGNO	(0 , N)	Può verificarsi il caso in cui per una data pianta non è stato ancora riscontrato un effettivo bisogno di un rinvaso, oppure ne sono stati già previsti diversi.
Dissoluzione	CONCIMAZIONE	(1 , N)	Un intervento di concimazione prevede sempre la somministrazione di almeno un micro o macro elemento, ma può richiederne un numero maggiore.
Dissoluzione	ELEMENTO DISCIOLTO	(0 , N)	Un elemento disciolto può essere utilizzato in più tipi di concimazione, ma anche in nessuna ed essere solo presente in un particolare terreno.
Sviluppo	PIANTA	(1 , 1)	Ogni pianta ha un unico indice di accrescimento.
Sviluppo	ACCRESIMENTO	(1 , N)	Almeno una pianta possiede un determinato indice di accrescimento perché esso viene creato in base a calcoli fatti sulle particolari caratteristiche di una pianta, ma può capitare anche che più di una pianta abbia lo stesso indice di accrescimento.
Acquisizione	CLIENTE	(0 , N)	Ogni cliente acquisisce una nuova scheda per ogni pianta acquistata, ma può anche non averne ancora.

Acquisizione	SCHEDA	(1 , 1)	Ogni scheda si crea nel momento in cui una pianta viene acquistata oppure quando viene richiesto un particolare tipo di intervento di manutenzione e, riferendosi ad un'unica pianta, è essa stessa unica.
Acquisto	PIANTA	(0 , N)	Perché possono esserci più schede di manutenzione e/o riferite ad ordini completamente evasi.
Acquisto	SCHEDA	(1 , 1)	Ogni scheda è collegata ad un unico ordine.
Produzione	SCHEDA	(0 , 1)	Ogni scheda può essere collegata al massimo ad un ordine.
Produzione	ORDINE	(0 , 1)	Ogni ordine può essere collegato solo ad una scheda, ma solo nel caso in cui risulti evaso.
Sottoposizione	PIANTA	(0 , 1)	Una pianta, se venduta è soggetta ad un ordine, altrimenti a nessuno.
Sottoposizione	ORDINE	(1 , 1)	Un ordine può essere riferito ad unica pianta.
Pubblicazione	CLIENTE	(0 , N)	Ogni utente può non aver scritto nessun post oppure tutti quelli che ha ritenuto opportuno, non ci sono limiti.
Pubblicazione	POST	(1 , 1)	Ogni post è scritto da un solo utente.
Scrittura	COMMENTO DI RISPOSTA	(1 , 1)	Ogni commento di risposta è riferito ad un solo post.
Scrittura	POST	(0 , N)	Ogni post può ricevere un numero illimitato di risposte, ma anche nessuna.

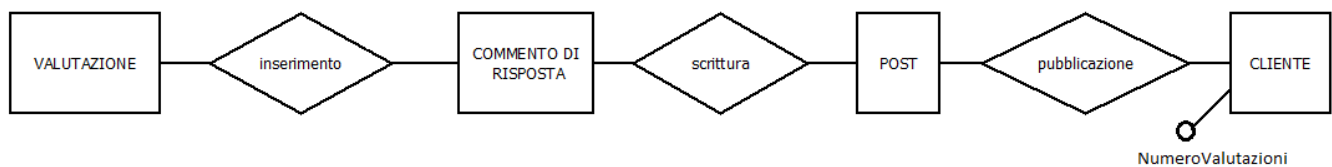
Ammette una	VALUTAZIONE	(1 , 1)	Avendo un codice univoco, ogni valutazione è associata ad un'unica risposta.
Ammette una	COMMENTO DI RISPOSTA	(0 , N)	Ogni commento di risposta può essere soggetto ad numero illimitato di valutazioni, ma anche nessuna.
Opinione	COMMENTO DI RISPOSTA	(1 , 1)	Un commento di risposta può essere aggiunto da un solo utente.
Opinione	CLIENTE	(0 , N)	Un cliente può aver risposto a più post o ancora a nessuno.
Schedatura Programmata	MANUTENZIONE PROGRAMMATA	(1 , 1)	Può esserci una sola, e sempre presente, programmazione da parte dell'utente di interventi manutentivi all'atto dell'acquisto.
Schedatura programmata	SCHEDA	(0 , 1)	Una scheda può riguardare un'unica richiesta di manutenzione programmata oppure nessuna nel caso in cui riguardi un altro tipo di richiesta.
Schedatura privata	MANUTENZIONE PRIVATA	(0 , 1)	Può esserci un solo intervento di manutenzione eseguito direttamente dall'utente oppure nessuno in caso non ne abbia effettuati in autonomia.
Schedatura privata	SCHEDA	(0 , 1)	Una scheda può riguardare un'unica richiesta di manutenzione privata oppure nessuna nel caso in cui riguardi un altro tipo di richiesta.
Schedatura richiesta	MANUTENZIONE SU RICHIESTA	(0 , 1)	Può esserci una sola richiesta di manutenzione straordinaria, nel caso il cliente lo ritenga necessario.
Schedatura richiesta	SCHEDA	(0 , 1)	Una scheda può riguardare un'unica richiesta di manutenzione su richiesta oppure nessuna nel caso in cui riguardi un altro tipo di richiesta.
Creazione	CLIENTE	(0 , N)	Ogni cliente può creare anche più di un giardino nell'area design, oppure nessuno.
Creazione	GIARDINO	(1 , 1)	Ogni giardino è creato da un solo cliente.
Posizionamento	GIARDINO	(1 , N)	Ogni giardino può essere formato da più settori, ma almeno da uno di sicuro.
Posizionamento	SETTORE	(1 , 1)	Ogni settore è presente in un solo giardino.

Posizione settore	SETTORE	(1 , 1)	Un settore è composto da un unico vertice superiore destro.
Posizione settore	VERTICE	(0 , 1)	Ogni vertice può riferirsi ad un solo settore, ma potrebbe riferirsi anche ad un vaso.
Posizione vaso	VASO	(1 , 1)	Un vaso è composto da un unico vertice superiore destro.
Posizione vaso	VERTICE	(0 , 1)	Ogni vertice può riferirsi ad un solo vaso, ma potrebbe riferirsi anche ad un settore.

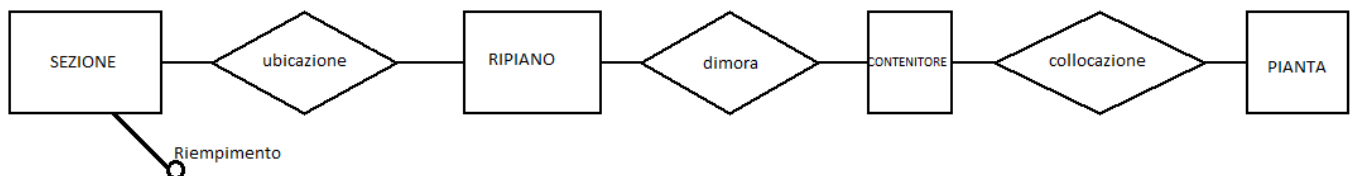
2. INDIVIDUAZIONE DI OPERAZIONI SUI DATI

RIDONDANZE SIGNIFICATIVE:

- i) L'attributo *NumeroValutazioni* nella tabella ACCOUNT può essere ricavato contando le valutazioni di ogni COMMENTO DI RISPOSTA eseguito dallo stesso CLIENTE ma è stato ritenuto opportuno inserire comunque l'attributo in modo ridondante.



- ii) L'attributo *Riempimento* nella tabella SEZIONE può essere ricavato contando le piante presenti in una determinata sezione ma è stato ritenuto opportuno inserire comunque l'attributo in modo ridondante.



Di seguito vengono riportate le otto operazioni ritenute significative per determinare le prestazioni del database (riservandone una di scrittura e una di lettura per ogni ridondanza, come richiesto nelle specifiche):

- ✿ Operazione 1:
 - Descrizione: Trovare quante valutazioni ha ricevuto un tale cliente tramite il *Nickname* (lettura sulla prima ridondanza)
 - Frequenza Giornaliera: 100 volte

- ✿ Operazione 2:
 - Descrizione: Inserire una nuova valutazione (scrittura sulla prima ridondanza)
 - Frequenza Giornaliera: 10 volte

- ✿ Operazione 3:
 - Descrizione: Trovare tutte le piante presenti in una sezione (lettura sulla seconda ridondanza)
 - Frequenza Giornaliera: 100 volte

- ✿ Operazione 4:
 - Descrizione: Inserire un nuova pianta (scrittura sulla seconda ridondanza)
 - Frequenza Giornaliera: 50 volte

- ✿ Operazione 5:
 - Descrizione: Aggiungere una pianta con un nuovo tipo di illuminazione (*INSERT*)
 - Frequenza Giornaliera: 1 volta

- ✿ Operazione 6:
 - Selezionare tutte le piante colpite da una certa patologia (*SELECT*)
 - Frequenza Giornaliera: 100 volte

- ✿ Operazione 7:
 - Eliminare un oggetto dal giardino (*DELETE*)
 - Frequenza Giornaliera: 50 volte

- ✿ Operazione 8:
 - Modificare il link presente in un post (*UPDATE*)
 - Frequenza Giornaliera: 2 volte

3. ANALISI DELLE PRESTAZIONI DELLE OPERAZIONI

Di seguito sono riportate le **Tavole dei Volumi** che permettono di stimare la mole di dati coinvolta in ciascuna entità o relazione, in termini di occorrenze.

Nome	E/R	Numero Istanze	Motivazione
SEDE	E	4	Ipotesi iniziale
SERRA	E	16	Ipotizziamo che in media ogni sede abbia 4 serre
SEZIONE	E	80	Ipotizziamo che in media ogni serra abbia 5 sezioni
RIPIANO	E	400	Ipotizziamo che in media ogni sezione abbia 5 ripiani
CONTENITORE	E	4000	Ipotizziamo che ogni ripiano abbia 10 contenitori
PIANTA	E	5000	Ipotizziamo che 1/5 delle piante sia a terreno aperto
SINTOMATOLOGIA	E	2000	Ipotizziamo che 2 piante su 5 manifestino una sintomatologia
SINTOMO	E	7	Come dettato da specifiche
IMMAGINE	E	14	Supponiamo che per ogni sintomo siano presenti 2 immagini
PATOLOGIA	E	2000	Ipotizziamo che ci siano tante patologie quante sintomatologie
AGENTEPATOGENO	E	6	Come dettato da specifiche
CURA	E	2500	Ipotizziamo che ci siano 5 diverse cure per ogni farmaco
FARMACO	E	600	Ipotizziamo che l'azienda disponga di 600 diversi farmaci
LUCE	E	50	Ipotizziamo che ci siano 50 diversi tipi di illuminazione
IRRIGAZIONE	E	50	Ipotizziamo che ci siano 50 diversi tipi di illuminazione
CONCIMAZIONE	E	200	Ipotizziamo che ci siano 200 diversi tipi di concimazione

RINVASO	E	75	Ipotizziamo che ci siano 75 diversi tipi di rinvaso
POTATURA	E	12	Ipotizziamo che ci siano 12 diversi tipi di potatura
ACCRESIMENTO	E	1000	Ipotizziamo che ci siano 1000 diversi indici di accrescimento
TERRENO	E	700	Ipotesi iniziale
ELEMENTO DISCIOLTO	E	2800	Supponiamo che in ogni terreno siano presenti in media 4 diversi elementi disciolti
FABBISOGNO	E	4000	Supponiamo che ogni pianta abbia avuto bisogno almeno una volta di manutenzione
INTERVENTO	E	4000	Supponiamo che in media gli interventi effettuati siano tanti quanti quelli richiesti
PERIODO	E	1200	Supponiamo che siano registrati 1200 periodi diversi
CLIENTE	E	750	Ipotesi iniziale
ORDINE	E	3000	Ipotizziamo che ogni utente compri in media 4 piante
SCHEDA	E	3500	Ipotizziamo che siano registrate 3500 schede
MANUTENZIONE SU RICHIESTA	E	200	Se ci sono 3500 schede e 3000 ordini 500 saranno schede riguardanti un intervento di manutenzione. Supponiamo che di queste, 200 siano fatte su richiesta
MANUTENZIONE PROGRAMMATA	E	150	Se ci sono 3500 schede e 3000 ordini 500 saranno schede riguardanti un intervento di manutenzione. Supponiamo che di queste, 150 siano programmate
MANUTENZIONE AUTOMATICA	E	100	Se ci sono 3500 schede e 3000 ordini 500 saranno schede riguardanti un intervento di manutenzione. Supponiamo che di queste, 100 siano automatiche
MANUTENZIONE PRIVATA	E	50	Se ci sono 3500 schede e 3000 ordini 500 saranno schede riguardanti un intervento di manutenzione. Supponiamo che di queste, 50 siano fatte in privato
GIARDINO	E	200	Ipotizziamo che solo 4/15 dei clienti crei un proprio modello di GIARDINO
SETTORE	E	1400	Ipotizziamo che ogni giardino abbia in media 7 settori
VASO	E	9100	Ipotizziamo che ogni settore abbia in media 7 vasi
POST	E	7500	Ipotizziamo che ogni cliente scriva in media 10 post nell'area social
COMMENTO DI RISPOSTA	E	75000	Ipotizziamo che ogni POST riceva in media 10 commenti di risposta
VALUTAZIONE	E	750000	Ipotizziamo che ogni COMMENTO DI RISPOSTA riceva in media 10 valutazioni

Collocazione	R	1,25	5000 piante / 4000 contenitori
Dimora	R	4000	Per cardinalità (1,1) con CONTENITORE
Ubicazione	R	400	Per cardinalità (1,1) con RIPIANO
Disposizione	R	80	Per cardinalità (1,1) con SEZIONE
Dislocazione	R	16	Per cardinalità (1,1) con SERRA
Sistemazione	R	62,5	5000 piante / 80 sezioni
Manifestazione	R	2,5	5000 piante / 2000 sintomatologia
Appartenenza	R	285	2000 sintomatologie / 7 sintomi
Descrizione	R	14	Per cardinalità (1,1) con IMMAGINE
Esordio	R	1	2000 sintomatologia / 2000 patologia
Individuazione	R	333	2000 patologia / 6 agente patogeno
Trattamento	R	100	600 farmaco / 6 agente patogeno
Prescrizione	R	48000	Per cardinalità (1,1) con CURA
Quarantena	R	2	5000 piante / 2500 cure
Bisogno	R	5000	Per cardinalità (1,1) con PIANTA
Sviluppo	R	5000	Per cardinalità (1,1) con PIANTA
Necessità	R	5000	Per cardinalità (1,1) con PIANTA
Esigenza	R	1,25	5000 piante / 4000 fabbisogno
Necessità concimazione	R	20	4000 fabbisogno / 200 concimazione
Necessità Potatura	R	333	4000 fabbisogno / 12 potature
Necessità rinvaso	R	53	4000 fabbisogno / 75 rinvasi
Occorrenza	R	1,25	5000 piante / 4000 intervento

Esecuzione concimazione	R	20	4000 interventi / 200 concimazione
Esecuzione potatura	R	333	4000 interventi / 12 potature
Esecuzione rinvaso	R	53	4000 interventi / 75 rinvasi
Suolo	R	5000	Per cardinalità (1,1) con PIANTA
Presenza	R	4	2800 elemento disciolto / 700 terreno
Dissoluzione	R	14	2800 elemento disciolto / 200 concimazione
Sottoposizione	R	3000	Per cardinalità (1,1) con ORDINE
Produzione	R	1,17	3500 scheda / 3000 ordine
Scheda Richiesta	R	17,5	3500 scheda / 200 manutenzione su richiesta
Scheda Programmata	R	23,3	3500 scheda / 150 manutenzione programmata
Scheda Automatica	R	35	3500 scheda / 100 manutenzione automatica
Scheda Privata	R	70	3500 scheda / 50 manutenzione privata
Acquisizione	R	3500	Per cardinalità (1,1) con SCHEDA
Creazione	R	3,75	750 cliente / 200 giardino
Posizionamento	R	3000	Per cardinalità (1,1) con SETTORE
Contenimento	R	6,5	9100 vaso / 1400 settore
Pubblicazione	R	7500	Per cardinalità (1,1) con POST
Scrittura	R	75000	Per cardinalità (1,1) con COMMENTO DI RISPOSTA
Inserimento	R	750000	Per cardinalità (1,1) con VALUTAZIONE
Individuazione	R	200	In media 1000 diverse patologie sono causate dallo stesso agente patogeno
Opinione	R	75000	Per cardinalità (1,1) con COMMENTO DI RISPOSTA
Vegetazione	R	5000	Per cardinalità (1,1) con PIANTA

Periodo Fioritura	R	2	Ipotizziamo che in media ogni pianta abbia 2 periodi di fioritura
Periodo Fruttificazione	R	2	Ipotizziamo che in media ogni pianta abbia 2 periodi di fruttificazione
Fabbisogno periodico	R	3,33	4000 fabbisogno / 1200 periodo
Periodo programmato	R	150	Per cardinalità (1,1) con MANUTENZIONE PROGRAMMATA
Periodo di manifestazione	R	2000	Per cardinalità (1,1) con PATOLOGIA
Periodo di non somministrazione	R	600	Per cardinalità (1,1) con FARMACO

4. INTRODUZIONE DELLE RIDONDANZE

Di seguito è riportata la **Tavola degli Accessi** che fornisce una stima delle operazioni necessarie all'esecuzione.

Operazione 1

Descrizione: Trovare quante valutazioni ha ricevuto un cliente tramite il *Nickname*

Frequenza giornaliera: 100 volte

La tavola degli accessi di questa operazione cambia a seconda che sia presente o meno la *Ridondanza 1*, cioè a seconda che sia più o meno presente l'attributo *NumeroValutazioni* all'interno dell'entità CLIENTE. Le due tavole degli accessi saranno quindi rispettivamente:

Diagramma E-R coinvolto (senza ridondanza):



Tavola dei volumi coinvolta (senza ridondanza):

CLIENTE	E	750	Ipotesi iniziale
Opinione	R	75000	Per cardinalità (1,1) con COMMENTO DI RISPOSTA
COMMENTO DI RISPOSTA	E	75000	Ipotizziamo che ogni POST riceva in media 10 commenti di risposta
Inserimento	R	750000	Per cardinalità (1,1) con VALUTAZIONE
VALUTAZIONE	E	750000	Ipotizziamo che ogni COMMENTO DI RISPOSTA riceva in media 10 valutazioni

Tavola degli accessi (senza ridondanza):

Senza ridondanza 1		
Num.Operazioni elementari	Tipo Operazioni	Motivazione
2	Lettura	Compio 2 operazioni di lettura nell'entità COMMENTO DI RISPOSTA: una per confrontare il <i>Nickname</i> con quello ottenuto e l'altra per recuperare il <i>CodiceCommentoRisposta</i>
2 x 20 = 40	Lettura	Per ognuno di essi (dalla tavola dei volumi si evince che ogni cliente scrive in media 20 commenti) compio un'operazione di lettura nell'entità VALUTAZIONE per confrontare il <i>CommentoDiRisposta</i> con il <i>CodiceCommentoDiRisposta</i> ottenuto e una per ricavare il <i>CodiceValutazione</i>
1	Lettura	Compio 1 operazione di lettura per contare le valutazioni ottenute
43	Totale operazioni elementari per singola operazione	
43 x 100 = 4300	Totale operazioni elementari al giorno	

Diagramma E-R coinvolto (con ridondanza):

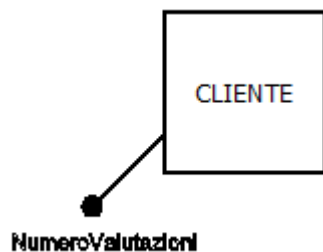


Tavola dei volumi coinvolta (con ridondanza):

CLIENTE	E	750	Ipotesi iniziale
---------	---	-----	------------------

Con ridondanza 1		
Num.Operazioni elementari	Tipo Operazioni	Motivazione
1	Lettura	1 operazione di lettura nell'entità CLIENTE per recuperare il <i>NumeroValutazioni</i>
1	Totale operazioni elementari per singola operazione	
1 x 100 = 100	Totale operazioni elementari al giorno	

Operazione 2

Descrizione: Inserire una nuova valutazione

Frequenza giornaliera: 10 volte

La tavola degli accessi di questa operazione cambia a seconda che sia presente o meno la *Ridondanza 1*, cioè a seconda che sia più o meno presente l'attributo *NumeroValutazioni* all'interno dell'entità CLIENTE. Le due tavole degli accessi saranno quindi rispettivamente:

Diagramma E-R coinvolto (senza ridondanza):



Tavola dei volumi coinvolta (senza ridondanza):

VALUTAZIONE	E	750000	Ipotizziamo che ogni COMMENTO DI RISPOSTA riceva in media 10 valutazioni
-------------	---	--------	--

Senza ridondanza 1		
Num.Operazioni elementari	Tipo Operazioni	Motivazione
1	Scrittura	Compio 1 operazione di scrittura nell'entità VALUTAZIONE per inserire il <i>CodiceValutazione</i>
$1 \times 2 = 2$	Totale operazioni elementari per singola operazione	
$2 \times 10 = 20$	Totale operazioni elementari al giorno	

Diagramma E-R coinvolto (con ridondanza):

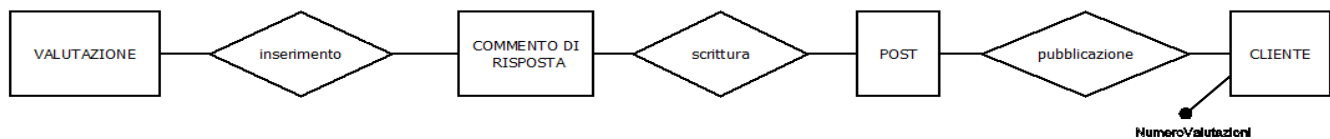


Tavola dei volumi coinvolta (con ridondanza):

CLIENTE	E	750	Ipotesi iniziale
Opinione	R	75000	Per cardinalità (1,1) con COMMENTO DI RISPOSTA
COMMENTO DI RISPOSTA	E	75000	Ipotizziamo che ogni POST riceva in media 10 commenti di risposta
Inserimento	R	750000	Per cardinalità (1,1) con VALUTAZIONE
VALUTAZIONE	E	750000	Ipotizziamo che ogni COMMENTO DI RISPOSTA riceva in media 10 valutazioni

Con ridondanza 1		
Num.Operazioni elementari	Tipo Operazioni	Motivazione
1	Scrittura	Compio 1 operazione di scrittura nell'entità VALUTAZIONE per inserire il <i>CodiceValutazione</i>
2	Lettura	Compio 2 operazioni di lettura nell'entità COMMENTO DI RISPOSTA: una per confrontare l'attributo <i>CommentoDiRisposta</i> con il <i>CodiceCommentoDiRisposta</i> e l'altra per recuperare il <i>Nickname</i>
1	Lettura	Compio 1 operazione di lettura nell'entità CLIENTE per confrontare il <i>Nickname</i> con quello precedentemente ottenuto
1	Scrittura	Compio 1 operazione di scrittura nell'entità CLIENTE per modificare l'attributo <i>NumeroValutazioni</i>
$3 + 2 \times 2 = 7$	Totale operazioni elementari per singola operazione	
$7 \times 10 = 70$	Totale operazioni elementari al giorno	

Operazione 3

Descrizione: Trovare quante piante sono presenti all'interno di una *Sezione* dato il *CodiceSezione*.

Frequenza giornaliera: 100 volte

La tavola degli accessi di questa operazione cambia a seconda che sia presente o meno la *Ridondanza 2*, cioè a seconda che sia più o meno presente l'attributo *Riempimento* all'interno dell'entità SEZIONE. Le due tavole degli accessi saranno quindi rispettivamente:

Diagramma E-R coinvolto (senza ridondanza):



Tavola dei volumi coinvolta (senza ridondanza):

PIANTA	E	5000	Ipotizziamo che 1/5 delle piante sia a terreno aperto
Sistemazione	R	62,5	5000 piante / 80 sezioni
SEZIONE	E	80	Ipotizziamo che in media ogni serra abbia 5 sezioni

Senza ridondanza 2		
Num.Operazioni elementari	Tipo Operazioni	Motivazione
2	Lettura	Compio 2 operazioni di lettura nell'entità PIANTA: una per confrontare l'attributo <i>Sezione</i> con il <i>CodiceSezione</i> e l'altra per recuperare il <i>CodicePianta</i>
1	Lettura	Compio 1 operazione di lettura per contare le piante
3	Totale operazioni elementari per singola operazione	
3 x 100 = 300	Totale operazioni elementari al giorno	

Diagramma E-R coinvolto (con ridondanza):



Tavola dei volumi coinvolta (con ridondanza):

SEZIONE	E	80	Ipotizziamo che in media ogni serra abbia 5 sezioni
---------	---	----	---

Con ridondanza 2		
Num.Operazioni elementari	Tipo Operazioni	Motivazione
1	Lettura	1 operazione di lettura nell'entità SEZIONE per recuperare il <i>Riempimento</i>
1	Totale operazioni elementari per singola operazione	
1 x 100 = 100	Totale operazioni elementari al giorno	

Operazione 4

Descrizione: Inserire una nuova pianta

Frequenza giornaliera: 50 volte

La tavola degli accessi di questa operazione cambia a seconda che sia presente o meno la *Ridondanza 2*, cioè a seconda che sia più o meno presente l'attributo *Acquirente* all'interno dell'entità ORDINE. Le due tavole degli accessi saranno quindi rispettivamente:

Diagramma E-R coinvolto (senza ridondanza):



Tavola dei volumi coinvolta (senza ridondanza):

PIANTA	E	5000	Ipotizziamo che 1/5 delle piante sia a terreno aperto
--------	---	------	---

Senza ridondanza 2		
Num.Operazioni elementari	Tipo Operazioni	Motivazione
1	Scrittura	Compio 1 operazione di scrittura nell'entità PIANTA per inserire il <i>CodicePianta</i>
$1 \times 2 = 2$	Totale operazioni elementari per singola operazione	
$2 \times 50 = 100$	Totale operazioni elementari al giorno	

Diagramma E-R coinvolto (con ridondanza):



Tavola dei volumi coinvolta (con ridondanza):

PIANTA	E	5000	Ipotizziamo che 1/5 delle piante sia a terreno aperto
Sistemazione	R	62,5	5000 piante / 80 sezioni
SEZIONE	E	80	Ipotizziamo che in media ogni serra abbia 5 sezioni

Con ridondanza 1		
Num.Operazioni elementari	Tipo Operazioni	Motivazione
1	Scrittura	Compio 1 operazione di scrittura nell'entità PIANTA per inserire il <i>CodicePianta</i>
1	Lettura	Compio 1 operazione di lettura nell'entità SEZIONE per confrontare l'attributo <i>Sezione</i> con il <i>CodiceSezione</i>
1	Scrittura	Compio 1 operazione di scrittura nell'entità SEZIONE per modificare l'attributo <i>Riempimento</i>
$1 + 2 \times 2 = 5$	Totale operazioni elementari per singola operazione	
$5 \times 50 = 250$	Totale operazioni elementari al giorno	

Operazione 5

Descrizione: Aggiungere una pianta con un nuovo tipo di illuminazione

Frequenza giornaliera: 1 volta

Diagramma E-R coinvolto:

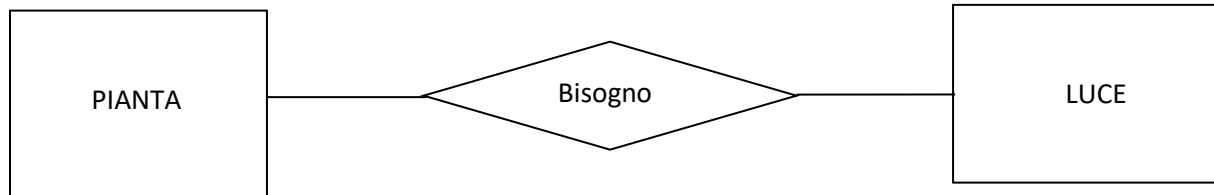


Tavola dei volumi coinvolta:

PIANTA	E	5000	Ipotizziamo che 1/5 delle piante sia a terreno aperto
LUCE	E	50	Ipotizziamo che ci siano 50 diversi tipi di illuminazione
Bisogno	R	5000	Per cardinalità (1,1) con PIANTA

Tavola degli accessi:

Num.Operazioni elementari	Tipo Operazioni	Motivazione
1	Scrittura	Compio 1 operazione di scrittura nell'entità LUCE per inserire il nuovo intervento dato
1	Scrittura	Compio 1 operazione di scrittura nell'entità PIANTA per inserire la pianta data
$2 \times 2 = 4$	Totale operazioni elementari per singola operazione	
$4 \times 1 = 4$	Totale operazioni elementari al giorno	

La tavola degli accessi di questa operazione non cambia in base alla presenza o meno delle ridondanze individuate.

Operazione 6

Descrizione: Selezionare tutte le piante colpite da una certa patologia

Frequenza giornaliera: 100 volte

Diagramma E-R coinvolto:

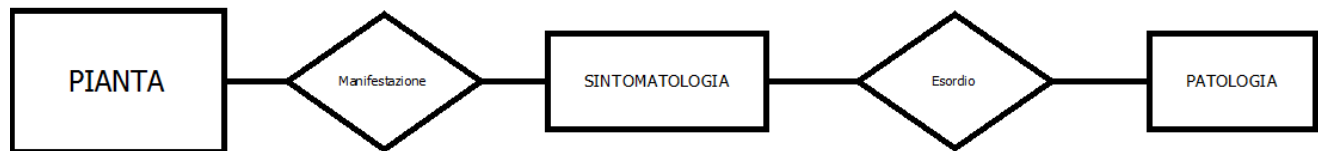


Tavola dei volumi coinvolta:

PIANTA	E	5000	Ipotizziamo che 1/5 delle piante sia a terreno aperto
SINTOMATOLOGIA	E	2000	Ipotizziamo che 2 piante su 5 manifestino una sintomatologia
PATOLOGIA	E	2000	Ipotizziamo che ci siano tante patologie quante sintomatologie
Manifestazione	R	2,5	5000 piante / 2000 sintomatologia
Esordio	R	1	2000 sintomatologia / 2000 patologia

Tavola degli accessi:

Num.Operazioni elementari	Tipo Operazioni	Motivazione
1	Lettura	Compio 1 operazione di lettura nell'entità PATOLOGIA per recuperare l'attributo <i>Sintomatologia</i>
2	Lettura	Compio 2 operazioni di lettura nell'entità SINTOMATOLOGIA: una per confrontare il <i>CodiceSintomatologia</i> con l'attributo ottenuto precedentemente e l'altra per recuperare l'attributo <i>PiantaColpita</i>
1	Lettura	Compio 1 operazione di lettura nell'entità PIANTA per confrontare l'attributo <i>PiantaColpita</i> con il <i>CodicePianta</i>
$1 + 2 + 1 = 4$	Totale operazioni elementari per singola operazione	
$4 \times 100 = 400$	Totale operazioni elementari al giorno	

La tavola degli accessi di questa operazione non cambia in base alla presenza o meno delle ridondanze individuate.

Operazione 7

Descrizione: Eliminare un settore di cui si conosce il codice dal giardino a cui appartiene

Frequenza giornaliera: 50 volte

Diagramma E-R coinvolto:



Tavola dei volumi coinvolta:

GIARDINO	E	200	Ipotizziamo che solo 4/15 dei clienti crei un proprio modello di GIARDINO
Posizionamento	R	3000	Per cardinalità (1,1) con SETTORE
SETTORE	E	1400	In base agli attributi dell'entità SETTORE sarebbero disponibili un'infinità di diversi codici distintivi, ma noi ipotizziamo che attualmente nel database ne siano memorizzati 1400

Tavola degli accessi:

Num.Operazioni elementari	Tipo Operazioni	Motivazione
1	Lettura	Compio 1 operazione di lettura nell'entità SETTORE per ricavare l'attributo <i>Giardino</i>
2	Lettura	Compio 2 operazioni di lettura nell'entità GIARDINO: una per confrontare il <i>CodiceGiardino</i> con l'attributo ottenuto precedentemente e l'altra per recuperare l'attributo <i>NumeroSettori</i>
1	Scrittura	Compio 1 operazione di scrittura nell'entità GIARDINO per modificare l'attributo <i>NumeroSettori</i>
1	Scrittura	Compio 1 operazione di scrittura nell'entità SETTORE per eliminare il settore indicato
$3 + 2 \times 2 = 7$	Totale operazioni elementari per singola operazione	
$7 \times 50 = 350$	Totale operazioni elementari al giorno	

La tavola degli accessi di questa operazione non cambia in base alla presenza o meno delle ridondanze individuate.

Operazione 8

Descrizione: modificare il link di un post dato

Frequenza giornaliera: 2 volte

Diagramma E-R coinvolto:



Tavola dei volumi coinvolta:

POST	E	7500	Ipotizziamo che ogni ACCOUNT scriva in media 10 post nell'area social
------	---	------	---

Tavola degli accessi:

Num.Operazioni elementari	Tipo Operazioni	Motivazione
1	Lettura	Compio 1 operazione di lettura nell'entità POST per ricavare l'attributo <i>Link</i>
1	Scrittura	Compio 1 operazione di scrittura nell'entità POST per modificare l'attributo <i>Link</i>
$1 + 1 \times 2 = 3$	Totale operazioni elementari per singola operazione	
$3 \times 2 = 6$	Totale operazioni elementari al giorno	

La tavola degli accessi di questa operazione non cambia in base alla presenza o meno delle ridondanze individuate.

Adesso si può fare il punto della situazione, per decidere se lasciare o meno ciascuna delle due ridondanze.

Ridondanza 1:

Senza ridondanza	
Operazione	Numero operazioni elementari al giorno
Operazione 1	4300
Operazione 2	20
Totale	4320

Con ridondanza	
Operazione	Numero operazioni elementari al giorno
Operazione 1	100
Operazione 2	70
Totale	170

Analizzando i due schemi, si può dedurre che la presenza della *Ridondanza 1* riduce, in media, i tempi di esecuzione delle operazioni e quindi si ritiene opportuno lasciare l'attributo *NumeroValutazioni* nella tabella CLIENTE.

Ridondanza 2:

Senza ridondanza	
Operazione	Numero operazioni elementari al giorno
Operazione 3	300
Operazione 4	100
Totale	400

Con ridondanza	
Operazione	Numero operazioni elementari al giorno
Operazione 3	100
Operazione 4	250
Totale	350

Analizzando i due schemi, si può dedurre che la presenza della *Ridondanza 2* riduce, in media, i tempi di esecuzione delle operazioni e quindi si ritiene opportuno lasciare l'attributo *Riempimento* nella tabella SEZIONE.

5. PROGETTAZIONE LOGICA

- PIANTA (CodicePianta, Nome, Genere, Cultivar, DimensioneMassima, Accrescimento, DistanzaMin, Infestante, NumeroFioritureAnnue, NumeroFruttificazioniAnnue, Luce, Terreno, Irrigazione, Sempreverde, Dioica, Sintomatologia, IndiceManutenzione)
- CONTENITORE (CodiceContentitore, Ripiano, SuperficieOccupata, LivelloIrradiazione, Pianta)
- RIPIANO (CodiceRipiano, Sezione, NumeroContentitori)
- SEZIONE (NomeSezione, Serra, Capienza, Riempimento, Irrigazione, Illuminazione, Temperatura, Umidita)
- SERRA (NomeSerra, Indirizzo, Larghezza, Lunghezza, Altezza, PiantaOspitabili, PiantaOspitate)
- SEDE (NomeSede, Indirizzo, NumeroDipendenti)
- SINTOMATOLOGIA (CodiceSintomatologia, Descrizione)
- SINTOMO (CodiceSintomo, Immagine, SintomatologiaDiAppartenenza)
- IMMAGINE (CodiceImmagine, Pixel, SintomoRiscontrato, Dimensione)
- PATOLOGIA (CodicePatologia, AgentePatogeno, Sintomatologia, ContraccettivoChimico, PeriodoManifestazioneInizio, PeriodoManifestazioneFine)
- FARMACO (CodiceFarmaco, PrincipioAttivo, Somministrazione, AgentePatogenoCombattuto, AmpioSpettro, DosaggioConsigliato, Cura, TempoMinPreConsumazione, PeriodoNonSomministrazioneInizio, PeriodoNonSomministrazioneFine)
- CURA (CodiceCura, Data, DosaggioEffettivo, Farmaco)
- ACCRESCIMENTO (CodiceAccrescimento, IndiceAccrescimento, AccrescimentoTop, AccrescimentoRoot, Pianta)
- LUCE (CodiceLuce, EsposizioneVegetativaMin, EsposizioneRiposoMin, Tipo, LuceDiretta)
- IRRIGAZIONE (CodiceIrrigazione, Fabbisognoldrico, PeriodicitaVegetativa, PeriodicitaRiposo)
- RINVASO (CodiceRinvaso, DimensioneVasoNuovo)
- CONCIMAZIONE (CodiceConcimazione, DataSomministrazione, NumeroSomministrazioni, Quantita, ElementiDisciolti, Periodicit , SomministrazioneRadicale)
- POTATURA (CodicePotatura, Tipo, Numero, PeriodoInizio, PeriodoFine)
- TERRENO (CodiceTerreno, PH, Consistenza, Permeabilit , ElementoDisciolto)

- ELEMENTO_DISCIOLTO (CodiceElementoDisciolto, Nome, PercentualePresenza, Terreno, Microelemento, Concimazione)
- ORDINE (CodiceOrdine, Data, Acquirente, Pianta, Stato, Scheda)
- SCHEDA (CodiceScheda, Pianta, DataAcquisto, Cliente, DimensioneAcquisto, DimensioneContenitore, TerrenoAperto, Tipo, Manutenzione)
- CLIENTE (Nickname, Email, Password, DomandaSicurezza, RispostaDomandaSicurezza, Nome, Cognome, CittàResidenza, NumeroValutazioni)
- GIARDINO (CodiceGiardino, Cliente, NumeroOggetti, Lunghezza, Larghezza)
- SETTORE (CodiceSettore, Giardino)
- VASO (CodiceVaso, Giardino, Settore)
- POST (Timestamp, Thread, Testo, Link, Nickname, Tema)
- COMMENTO_DI_RISPOSTA (Timestamp, PostDiRiferimento, Nickname, Testo, Valutazione)
- AGENTE_PATOGENO (Nome, Patologia, Farmaco)
- VALUTAZIONE (CodiceValutazione, Commento, Punteggio, Nickname)
- QUARANTENA (CodiceQuarantena, DataInizio, DataFine, Esito, Pianta, Cura)
- PERIODO (CodicePeriodo, Tipo, MeseInizio, MeseFine)
- FIORITURA (CodiceFioritura, Pianta)
- FRUTTIFICAZIONE (CodiceFruttificazione, Pianta)
- FABBISOGNO (CodiceFabbisogno, Tipo, CodiceTipo, Effettuata, PeriodoConsigliato)
- INTERVENTO (CodiceIntervento, Tipo, CodiceTipo, Data, StimaProssimaData)
- MANUTENZIONE_AUTOMATICA (CodiceManutenzioneAutomatica,)
- MANUTENZIONE_PRIVATA (CodiceManutenzionePrivata,)
- MANUTENZIONE_SU_RICHIESTA (CodiceManutenzioneSuRichiesta,)
- MANUTENZIONE_PROGRAMMATA (CodiceManutenzioneProgrammata)

- *Vincoli di integrità sulle ridondanze introdotte:*

Sono stati implementati i trigger che consentono la corretta gestione dei vincoli di integrità generati dalla presenza delle due ridondanze:

```
delimiter $$
CREATE TRIGGER `PrimaRidondanzaInsert` AFTER INSERT ON `VALUTAZIONE` FOR EACH
ROW BEGIN
SET @ClientInteressato = (SELECT Nickname
                           FROM COMMENTODIRISPOSTA
                           WHERE NEW.CommentoDiRisposta =
CodiceCommentoDiRisposta
                           );
UPDATE `CLIENT`
SET `NumeroValutazioni` = `NumeroValutazioni` + 1
WHERE `Nickname` = @ClientInteressato;
END $$
delimiter ;
```

```
delimiter $$
CREATE TRIGGER `PrimaRidondanzaDelete` AFTER DELETE ON `VALUTAZIONE` FOR EACH
ROW BEGIN
SET @ClientInteressato = (SELECT Nickname
                           FROM COMMENTODIRISPOSTA
                           WHERE OLD.CommentoDiRisposta =
CodiceCommentoDiRisposta
                           );
UPDATE `CLIENT`
SET `NumeroValutazioni` = `NumeroValutazioni` - 1
WHERE `Nickname` = @ClientInteressato;
END $$
delimiter ;
```

















Si è provveduto alla gestione della modifica sia un update che in delete di una valutazione in modo che l'attributo ridonante presente nell'account di ogni cliente venga sempre aggiornato.


















```
delimiter $$
CREATE TRIGGER `SecondaRidondanzaInsert` BEFORE INSERT ON `PIANTA` FOR EACH ROW
BEGIN
SET @riempimento = (SELECT Riempimento
                     FROM SEZIONE
                     WHERE NEW.Sezione = CodiceSezione
                     );
SET @capienza = (SELECT Capienza
                 FROM SEZIONE
                 WHERE NEW.Sezione = CodiceSezione
                 );
IF @capienza = @riempimento THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Impossibile eseguire procedura: SEZIONE GIA` PIENA';
ELSE
UPDATE `SEZIONE`
SET `Riempimento` = `Riempimento` + 1
WHERE `CodiceSezione` = NEW.Sezione;
END IF;
END $$
delimiter ;
```











```
delimiter $$
CREATE TRIGGER `SecondaRidondanzaDelete` AFTER DELETE ON `PIANTA` FOR EACH ROW
BEGIN
UPDATE `SEZIONE`
SET `Riempimento` = `Riempimento` - 1
WHERE `CodiceSezione` = OLD.Sezione;
END $$
delimiter ;
```

Si è provveduto alla gestione della modifica sia un update che in delete di una pianta all'interno di una sezione in modo che l'attributo ridonante presente in SEZIONE venga sempre aggiornato.

Saranno di seguito riportati tutti i **Vincoli di Integrità Referenziale** necessari al corretto impiego del database:

-  Esiste un vincolo di integrità referenziale tra l'attributo *Contenitore* della tabella PIANTA e l'attributo *CodiceContenitore* della tabella CONTENITORE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Sezione* della tabella PIANTA e l'attributo *CodiceSezione* della tabella SEZIONE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Ripiano* della tabella CONTENITORE e l'attributo *CodiceRipiano* della tabella RIPIANO.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Sezione* della tabella RIPIANO e l'attributo *CodiceSezione* della tabella SEZIONE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Serra* della tabella SEZIONE e l'attributo *CodiceSerra* della tabella SERRA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Sede* della tabella SERRA e l'attributo *CodiceSede* della tabella SEDE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Sintomatologia* della tabella PIANTA e l'attributo *CodiceSintomatologia* della tabella SINTOMATOLOGIA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Sintomatologia* della tabella PATOLOGIA e l'attributo *CodiceSintomatologia* della tabella SINTOMATOLOGIA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *SintomatologiaAppartenenza* della tabella SINTOMO e l'attributo *CodiceSintomatologia* della tabella SINTOMATOLOGIA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *SintomoRiscontrato* della tabella IMMAGINE e l'attributo *Codicesintomo* della tabella SINTOMO.
-  Esiste un vincolo di integrità referenziale tra l'attributo *CodiceFarmaco* della tabella FARMACO e l'attributo *FarmacoSomministrato* della tabella CURA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Accrescimento* della tabella PIANTA e l'attributo *CodiceAccrescimento* della tabella ACCRESCIMENTO.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Cura* della tabella QUARANTENA e l'attributo *CodiceCura* della tabella CURA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Pianta* della tabella QUARANTENA e l'attributo *CodicePianta* della tabella PIANTA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Luce* della tabella PIANTA e l'attributo *CodiceLuce* della tabella LUCE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Irrigazione* della tabella PIANTA e l'attributo *CodiceIrrigazione* della tabella IRRIGAZIONE.

-  Esiste un vincolo di integrità referenziale tra l'attributo *Fabbisogno* della tabella PIANTA e l'attributo *CodiceFabbisogno* della tabella FABBISOGNO.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Intervento* della tabella PIANTA e l'attributo *CodiceIntervento* della tabella INTERVENTO.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Tipo* della tabella INTERVENTO e l'attributo *CodicePotatura* della tabella POTATURA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *CodicePianta* della tabella PIANTA e l'attributo *Pianta* della tabella ORDINE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Scheda* della tabella ORDINE e l'attributo *CodiceScheda* della tabella SCHEDA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *PeriodoVegetativo* della tabella PIANTA e l'attributo *CodicePeriodo* della tabella PERIODO.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Cliente* della tabella SCHEDA e l'attributo *Nickname* della tabella CLIENTE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Cliente* della tabella GIARDINO e l'attributo *Nickname* della tabella CLIENTE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Giardino* della tabella SETTORE e l'attributo *CodiceGiardino* della tabella GIARDINO.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Nickname* della tabella CLIENTE e l'attributo *Cliente* della tabella POST.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Nickname* della tabella CLIENTE e l'attributo *Nickname* della tabella VALUTAZIONE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Nickname* della tabella CLIENTE e l'attributo *Nickname* della tabella COMMENTO_DI_RISPOSTA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *PostRiferimento* della tabella COMMENTO_RISPOSTA e l'attributo *CodicePost* della tabella POST.
-  Esiste un vincolo di integrità referenziale tra l'attributo *CommentoRiferimento* della tabella VALUTAZIONE e l'attributo *CodiceValutazione* della tabella VALUTAZIONE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *CodiceElementoDisciolto* della tabella ELEMENTO_DISCIOLTO e l'attributo *ElementoDisciolto* della tabella TERRENO.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Concimazione* della tabella ELEMENTO_DISCIOLTO e l'attributo *CodiceConcimazione* della tabella CONCIMAZIONE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Settore* della tabella VASO e l'attributo *CodiceSettore* della tabella SETTORE.

-  Esiste un vincolo di integrità referenziale tra l'attributo *Patologia* nella tabella AGENTE_PATOGENO e l'attributo *CodicePatologia* della tabella PATOLOGIA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Farmaco* della tabella AGENTE_PATOGENO e l'attributo *CodiceFarmaco* della tabella FARMACO.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Sintomo* della tabella IMMAGINE e l'attributo *CodiceSintomo* della tabella SINTOMO.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Sintomatologia* della tabella PATOLOGIA e l'attributo *CodiceSintomatologia* della tabella SINTOMATOLOGIA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Tipo* della tabella PERIODO e l'attributo *CodiceManutenzioneProgrammata* della tabella MANUTENZIONE_PROGRAMMATA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Tipo* della tabella INTERVENTO e l'attributo *CodicePotatura* della tabella POTATURA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Tipo* della tabella INTERVENTO e l'attributo *CodiceRinvaso* della tabella RINVASO.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Tipo* della tabella FABBISOGNO e l'attributo *CodicePotatura* della tabella POTATURA.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Tipo* della tabella FABBISOGNO e l'attributo *CodiceConcimazione* della tabella CONCIMAZIONE.
-  Esiste un vincolo di integrità referenziale tra l'attributo *Tipo* della tabella FABBISOGNO e l'attributo *CodiceRinvaso* della tabella POTATURA.

- *Trigger sui Vincoli di Integrità*

```
delimiter $$
CREATE TRIGGER `ValiditaScheda` BEFORE INSERT ON `SCHEDA` FOR EACH ROW BEGIN
IF (NEW.DimensioneContenitore IS NOT NULL AND NEW.TerrenoAperto = 1) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da inserimento! La pianta o è in terreno aperto
                                o ha contenitore di dimensione definita';
ELSEIF NEW.Tipo = 'Ordine' AND NEW.CodicePianta IN (SELECT CodicePianta
                                                    FROM SCHEDA
                                                    WHERE Tipo = 'Ordine') THEN

SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore! Pianta già ordinata!';
END IF;
END $$
delimiter ;
```

Questo trigger ha il compito per verificare la validità di una scheda e agisce impedendo che si crei una scheda in cui risulta che la pianta sia posta in terreno aperto ma che contemporaneamente abbia tra i suoi attributi la dimensione di un contenitore. Controlla inoltre che, nel caso in cui la scheda registri un ordine, la pianta interessata non sia già stata acquistata in precedenza.

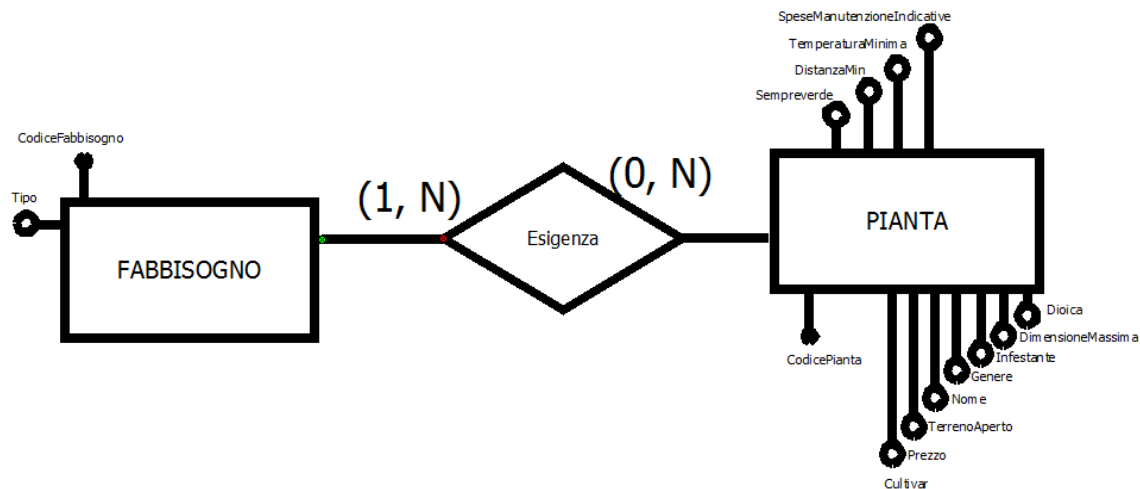
```
CREATE TRIGGER `ValiditaElemento` BEFORE INSERT ON `ELEMENTODISCIOLTO` FOR EACH ROW
BEGIN
SET @controllopercentuale = (SELECT SUM(PercentualePresenza)
                              FROM ELEMENTODISCIOLTO
                              WHERE Terreno = NEW.Terreno
                              );
IF NEW.`Nome` IN ( SELECT Nome
                  FROM ELEMENTODISCIOLTO
                  WHERE Terreno = NEW.Terreno) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore! Elemento già inserito in questo terreno';
ELSEIF NEW.`Nome` IN ( SELECT Nome
                      FROM ELEMENTODISCIOLTO
                      WHERE Concimazione = NEW.Concimazione) THEN

SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore! Elemento già inserito in questo intervento di
concimazione';
ELSEIF (NEW.`PercentualePresenza` + @controllopercentuale > 100) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore! Percentuale complessiva superiore al 100%';
END IF;
END $$
delimiter ;
```

Questo trigger ha il compito di verificare la validità dell'inserimento di un elemento disciolto ai fini della corretta espressione della percentuale con cui esso è presente nel terreno. In particolare il trigger ``ValiditaElemento`` impedisce che si inserisca lo stesso elemento nello stesso terreno e che la percentuale di presenza dei singoli elementi di un terreno superi il 100%.

6. TRADUZIONE DELLE CARDINALITA'

TRADUZIONE CARDINALITA' (1, N) – (0, N):



In questo caso è necessario predisporre tre tabelle legate da più vincoli di integrità referenziale:

FABBISOGNO (CodiceFabbisogno, Tipo)

PIANTA (CodicePianta, DistanzaMin, TemperaturaMinima, Sempreverde, SpeseManutenzioneIndicative, Dioica, DimensioneMassima, Infestante, Genere, Nome, TerrenoAperto, Prezzo, Cultivar, CodiceContenitore)

ESIGENZA (CodiceFabbisogno, CodicePianta)

TRADUZIONE CARDINALITA' (0, N) – (0, N):



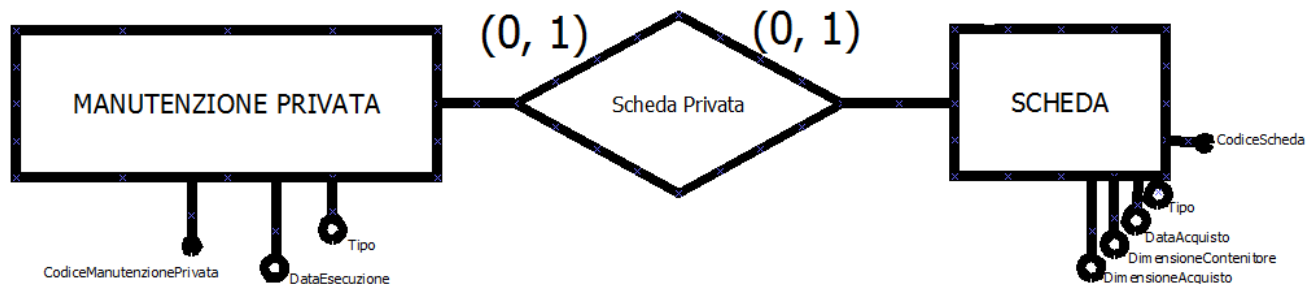
In questo caso è necessario predisporre tre tabelle legate da più vincoli di integrità referenziale:

RINVASO (CodiceRinvaso)

INTERVENTO (CodiceIntervento, ProssimaDataConsigliata, Data, Tipo)

ESECUZIONE RINVASO (CodiceRinvaso, CodiceIntervento)

TRADUZIONE CARDINALITA' (0, 1) – (0, 1):

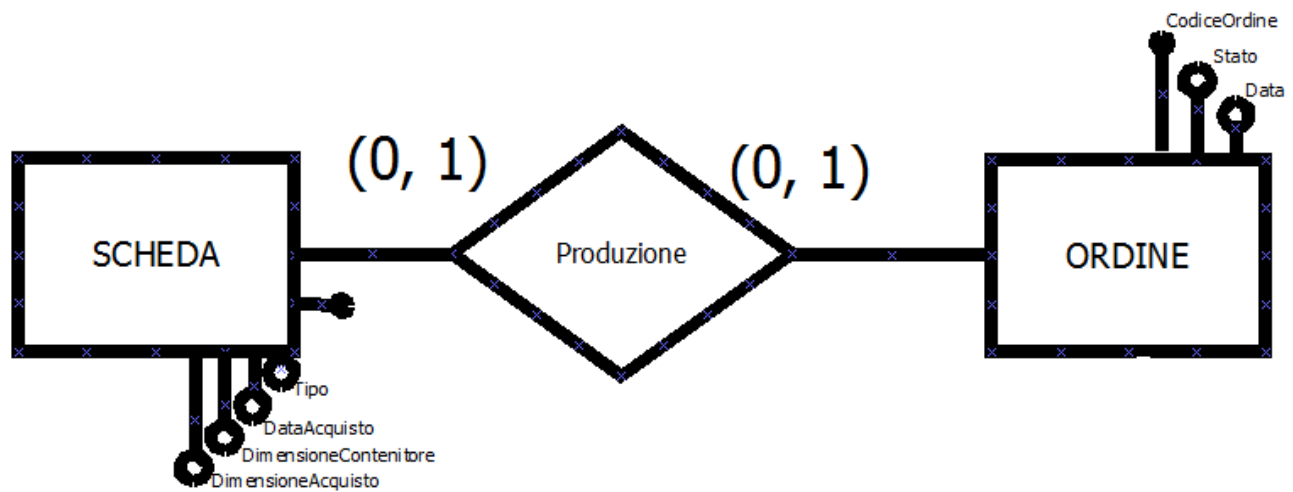


In questo caso si è deciso di procedere utilizzando sono le due tabelle provenienti dalle entità MANUTENZIONE PRIVATA e SCHEDA e di sdoppiare la relazione come segue:

MANUTENZIONE_PRIVATA (CodiceManutenzionePrivata, DataEsecuzione, Tipo, CodiceScheda)

SCHEDA (CodiceScheda, Tipo, DataAcquisto, DimensioneContenitore, DimensioneAcquisto)

In questo modo si introduce solo un vincolo di integrità referenziale che lega le due tabelle.

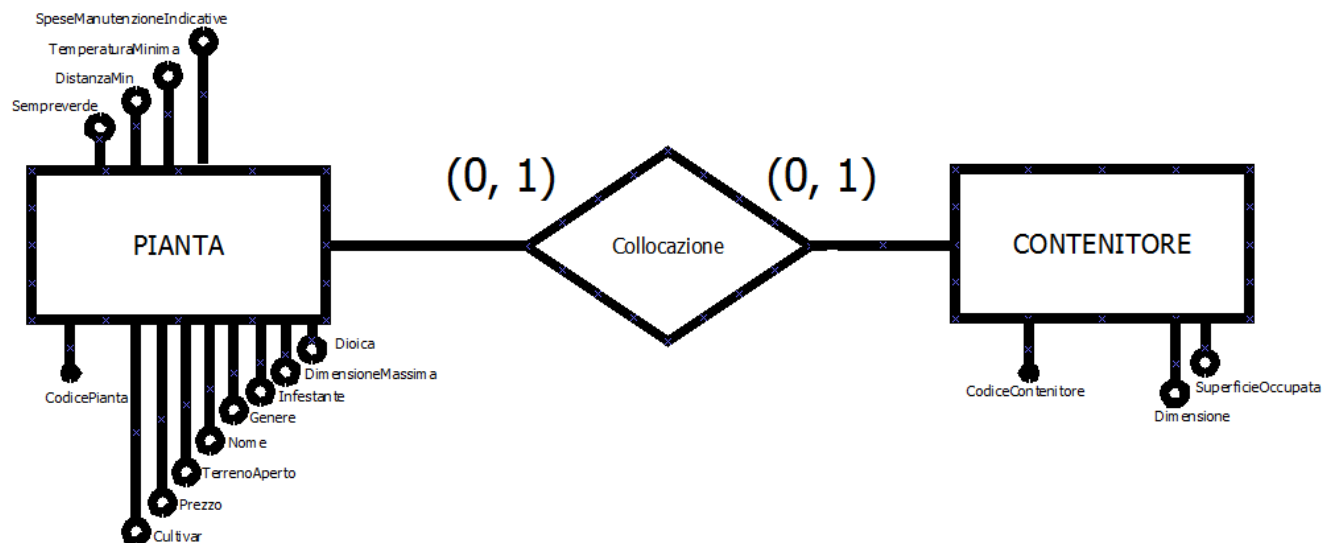


In questo caso si è deciso di procedere utilizzando sono le due tabelle provenienti dalle entità ORDINE e SCHEDA e di sdoppiare la relazione come segue:

ORDINE (CodiceOrdine, Stato, Data, CodiceScheda)

SCHEDA (CodiceScheda, Tipo, DataAcquisto, DimensioneContenitore, DimensioneAcquisto)

In questo modo si introduce solo un vincolo di integrità referenziale che lega le due tabelle.

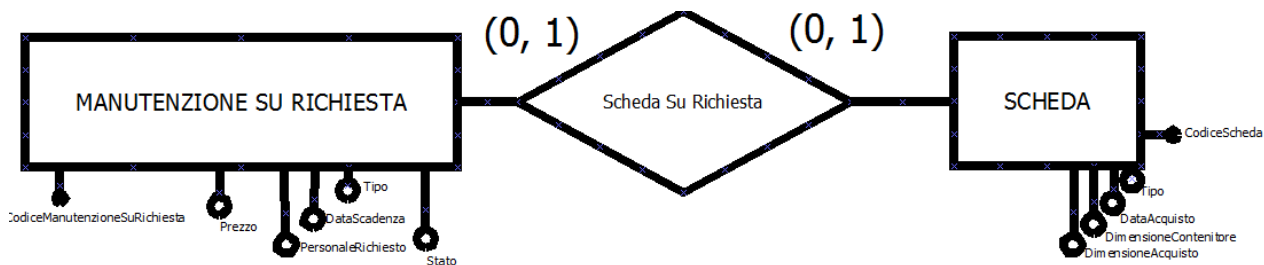


In questo caso si è deciso di procedere utilizzando sono le due tabelle provenienti dalle entità PIANTA e CONTENITORE e di sdoppiare la relazione come segue:

PIANTA (CodicePianta, DistanzaMin, TemperaturaMinima, Sempreverde, SpeseManutenzioneIndicative, Dioica, DimensioneMassima, Infestante, Genere, Nome, TerrenoAperto, Prezzo, Cultivar, CodiceContenitore)

CONTENITORE (CodiceContenitore, SuperficieOccupata, Dimensione)

In questo modo si introduce solo un vincolo di integrità referenziale che lega le due tabelle.



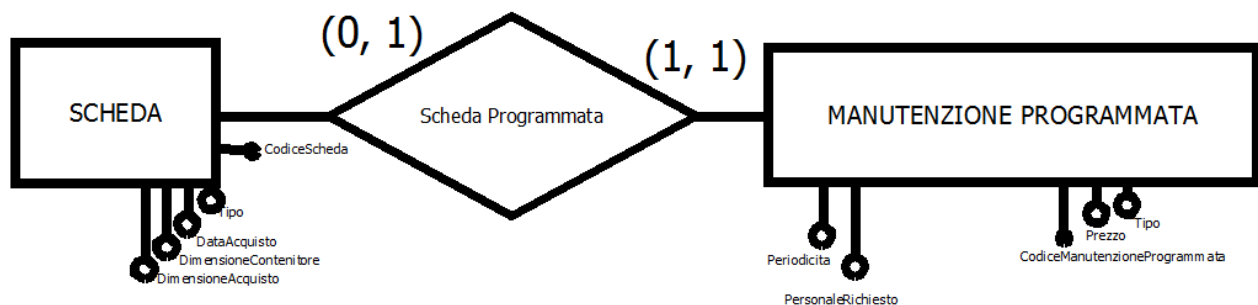
In questo caso si è deciso di procedere utilizzando sono le due tabelle provenienti dalle entità MANUTENZIONE_SU_RICHIESTA e SCHEDA e di sdoppiare la relazione come segue:

MANUTENZIONE_PRIVATA (CodiceManutenzioneSuRichiesta, DataScadenza, Tipo, Prezzo, PersonaleRichiesto, Stato, CodiceScheda)

SCHEDA (CodiceScheda, Tipo, DataAcquisto, DimensioneContenitore, DimensioneAcquisto)

In questo modo si introduce solo un vincolo di integrità referenziale che lega le due tabelle.

TRADUZIONE CARDINALITA' (0, 1) – (1, 1):

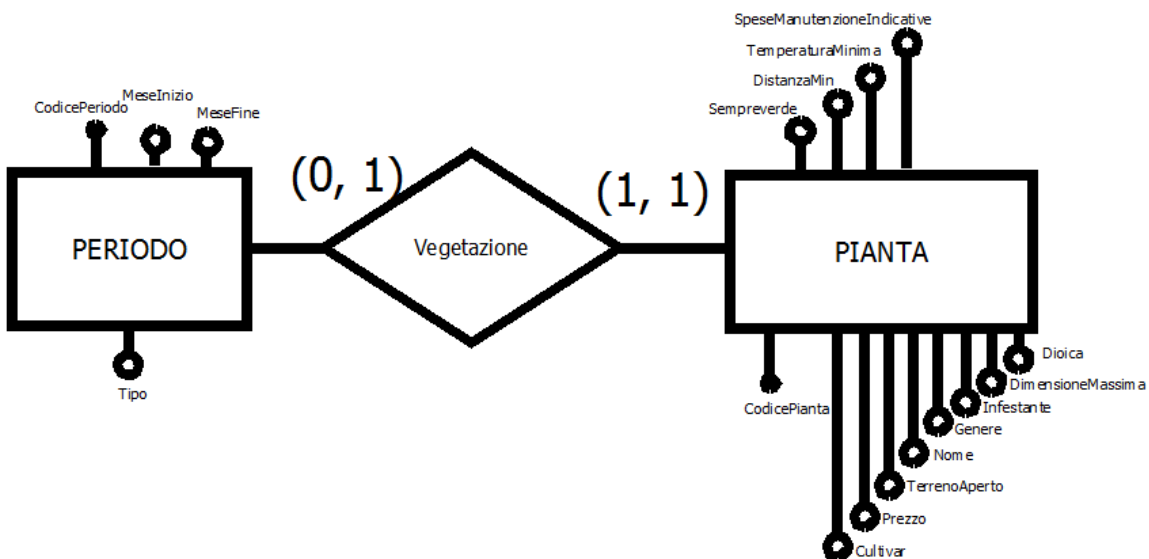


In questo caso si è deciso di procedere utilizzando sono le due tabelle provenienti dalle entità MANUTENZIONE_PROGRAMMATA e SCHEDA e di sdoppiare la relazione come segue:

MANUTENZIONE_PRIVATA (CodiceManutenzioneProgrammata, Periodicità, Tipo, Prezzo, PersonaleRichiesto, Stato, CodiceScheda)

SCHEDA (CodiceScheda, Tipo, DataAcquisto, DimensioneContenitore, DimensioneAcquisto)

In questo modo si introduce solo un vincolo di integrità referenziale che lega le due tabelle.



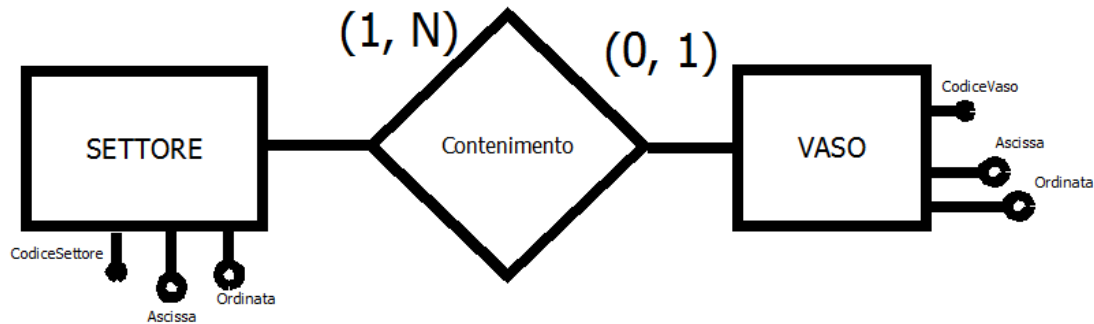
In questo caso si è deciso di procedere utilizzando sono le due tabelle provenienti dalle entità PERIODO e PIANTA e di sdoppiare la relazione come segue:

PERIODO (CodicePeriodo, Tipo, MeseFine, MeseInizio, CodicePianta)

PIANTA (CodicePianta, DistanzaMin, TemperaturaMinima, Sempreverde, SpeseManutenzioneIndicative, Dioica, DimensioneMassima, Infestante, Genere, Nome, TerrenoAperto, Prezzo, Cultivar, CodiceContenitore)

In questo modo si introduce solo un vincolo di integrità referenziale che lega le due tabelle.

TRADUZIONE CARDINALITA' (0, 1) – (1, N):

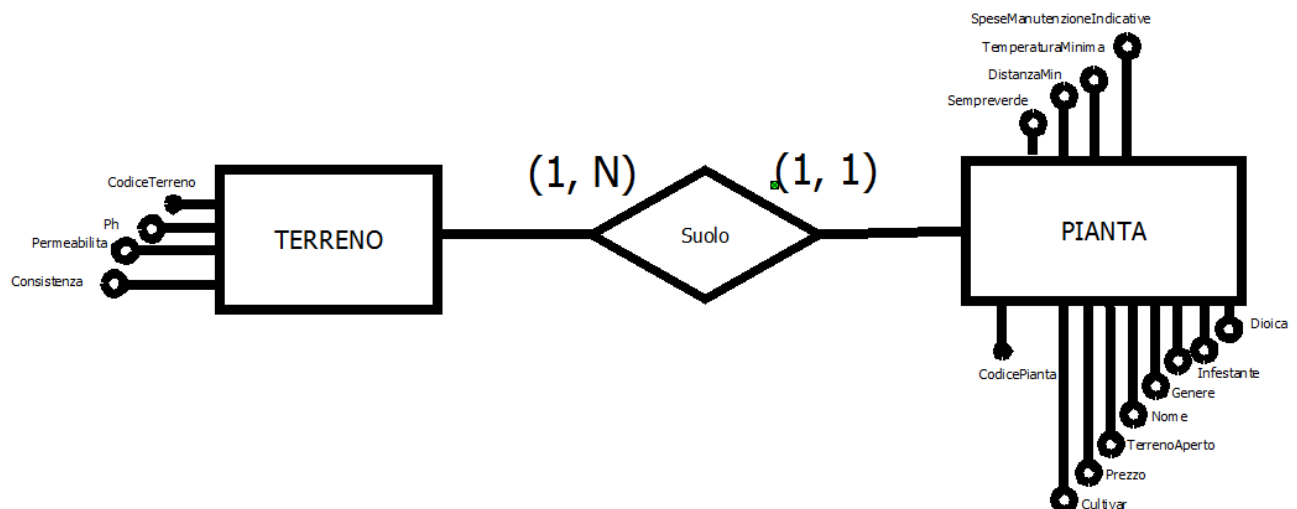


In questo caso si è deciso di procedere utilizzando sono le due tabelle provenienti dalle entità SETTORE e VASO e di sdoppiare la relazione come segue:

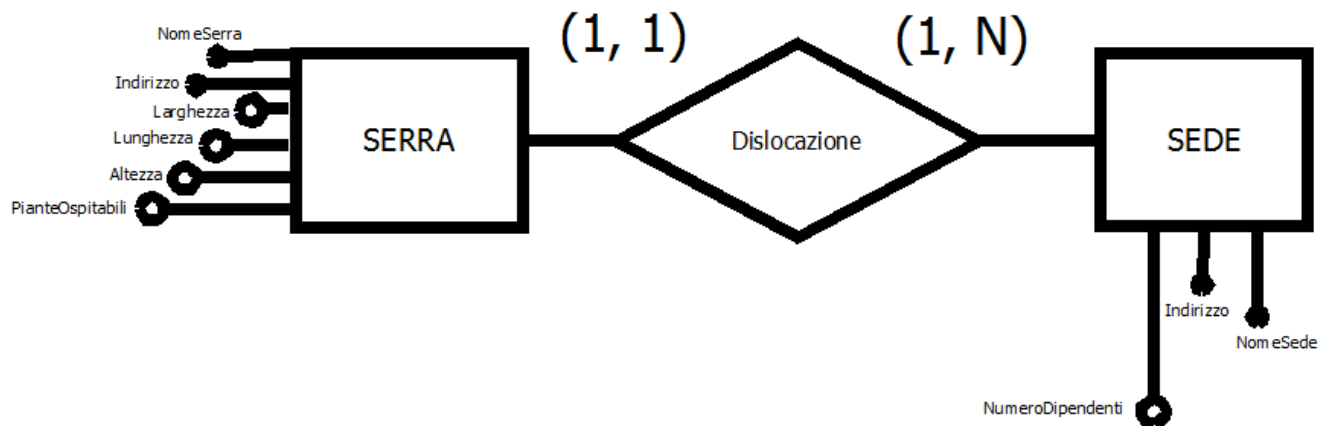
SETTORE (CodiceSettore, Ascissa, Ordinata)

VASO (CodiceVaso, DistanzaMin, Ascissa, Ordinata, CodiceSettore)

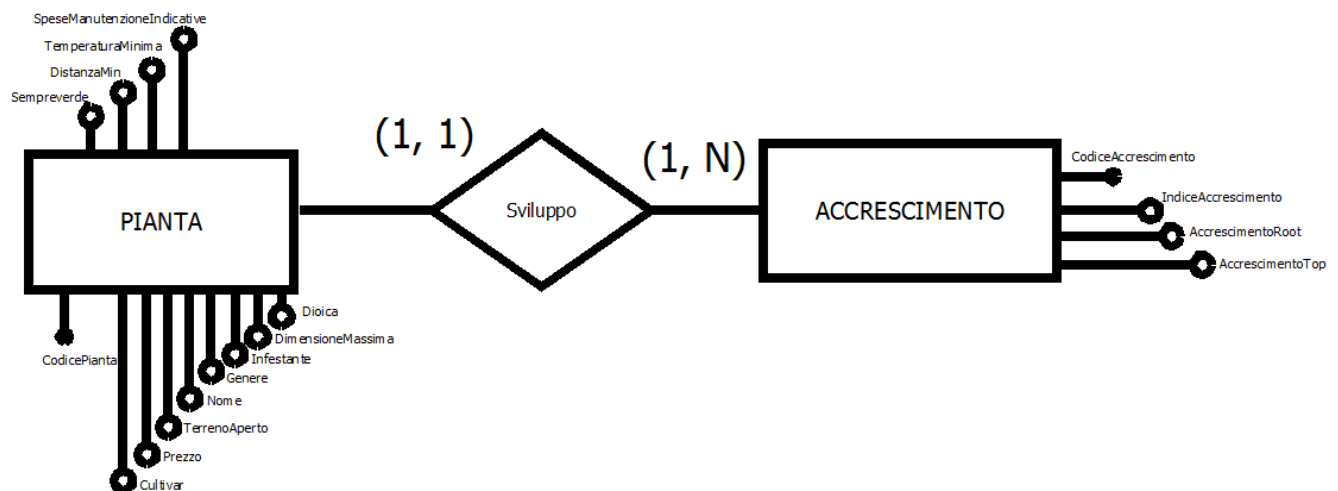
TRADUZIONE CARDINALITA' (1, 1) – (1, N):



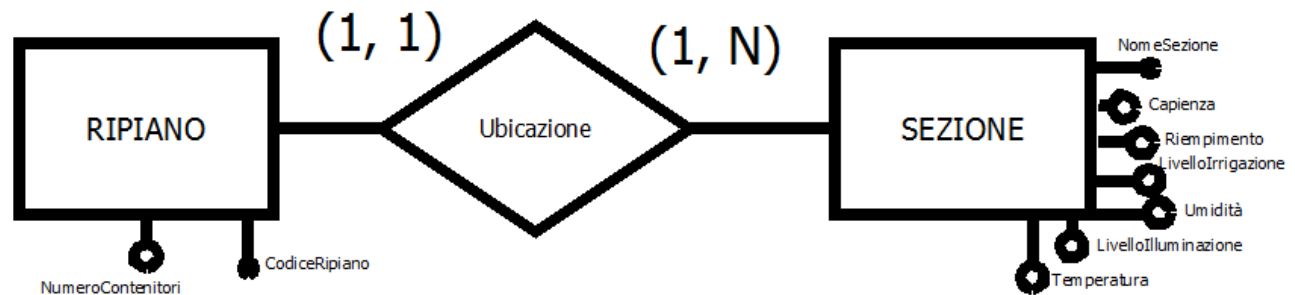
In questo caso risulta più conveniente accorpare la relazione nella tabella PIANTA, aggiungendo semplicemente un vincolo di integrità referenziale tra le due tabelle.



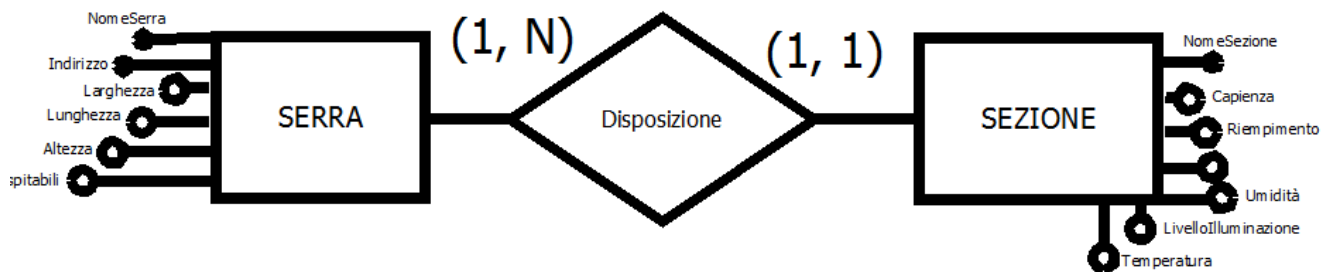
In questo caso risulta più conveniente accorpare la relazione nella tabella SERRA, aggiungendo semplicemente un vincolo di integrità referenziale tra le due tabelle.



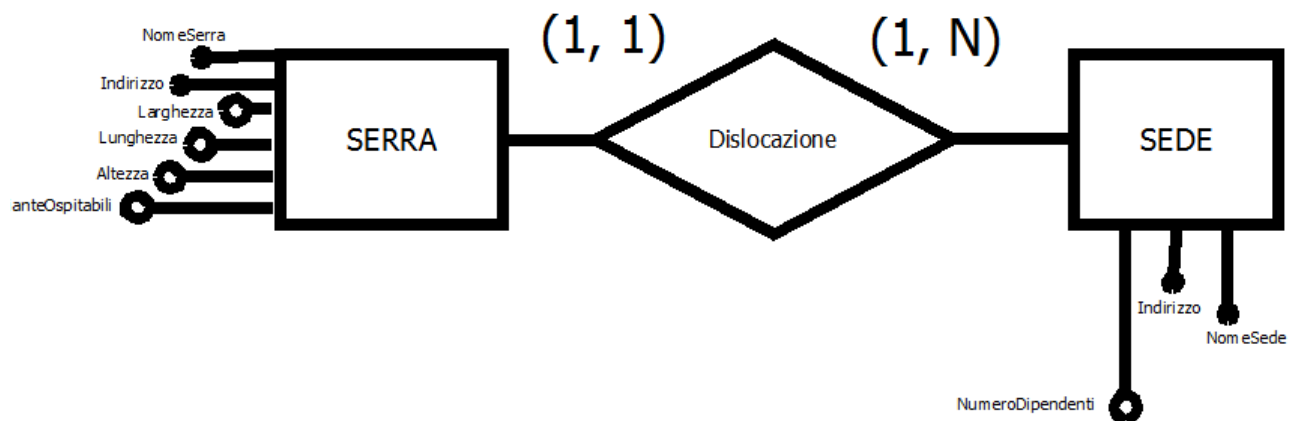
In questo caso risulta più conveniente accorpare la relazione nella tabella PIANTA, aggiungendo semplicemente un vincolo di integrità referenziale tra le due tabelle.



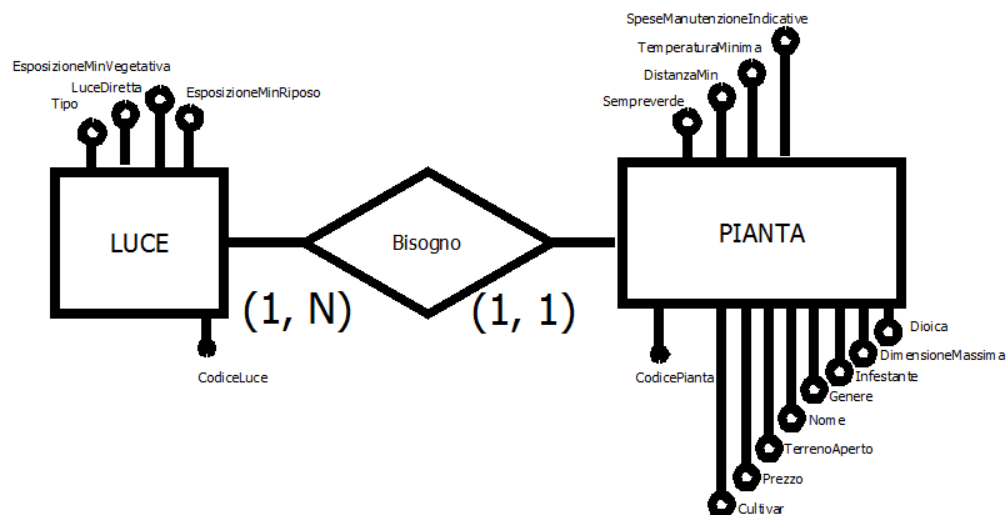
In questo caso risulta più conveniente accorpare la relazione nella tabella RIPIANO, aggiungendo semplicemente un vincolo di integrità referenziale tra le due tabelle.



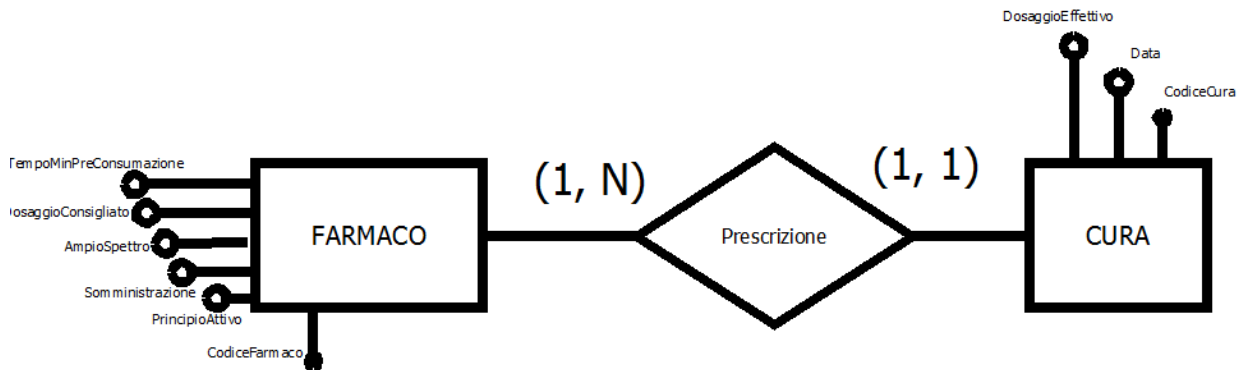
In questo caso risulta più conveniente accorpare la relazione nella tabella SEZIONE, aggiungendo semplicemente un vincolo di integrità referenziale tra le due tabelle.



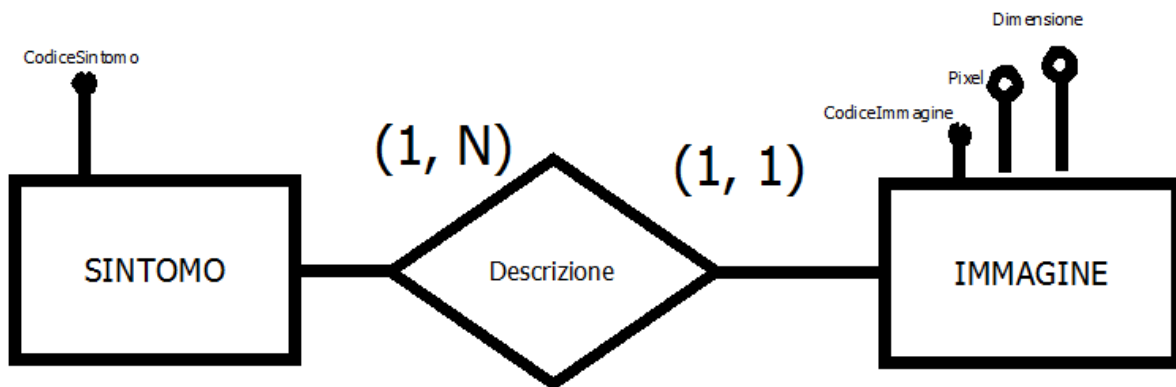
In questo caso risulta più conveniente accorpare la relazione nella tabella SERRA, aggiungendo semplicemente un vincolo di integrità referenziale tra le due tabelle.



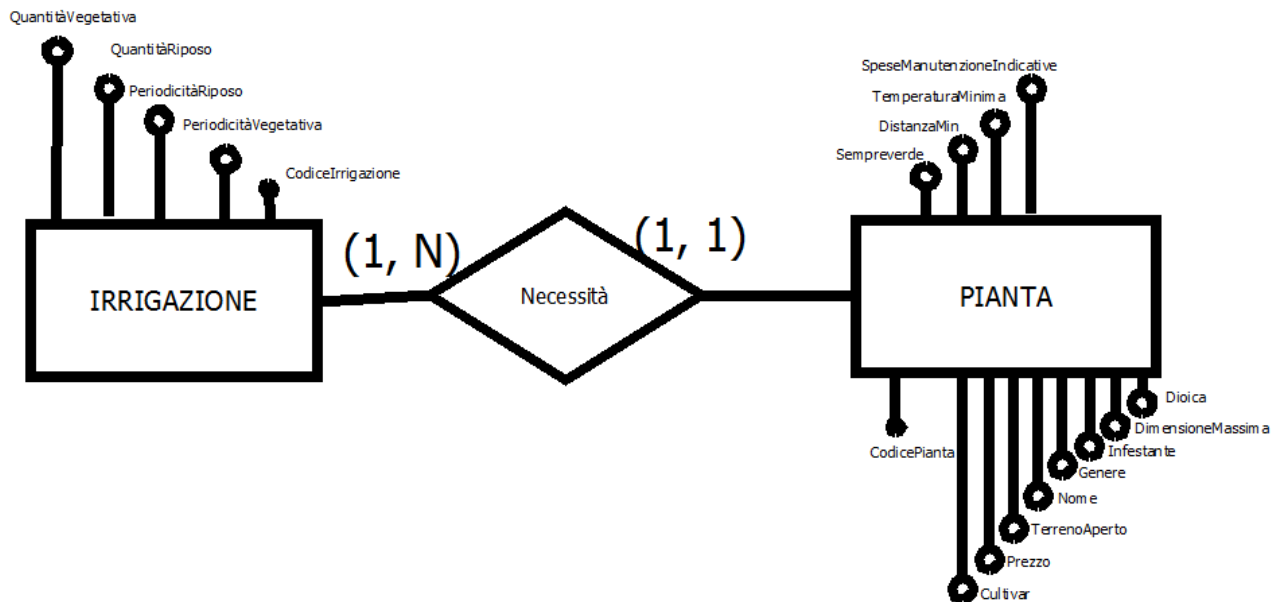
In questo caso risulta più conveniente accorpare la relazione nella tabella PIANTA, aggiungendo semplicemente un vincolo di integrità referenziale tra le due tabelle.



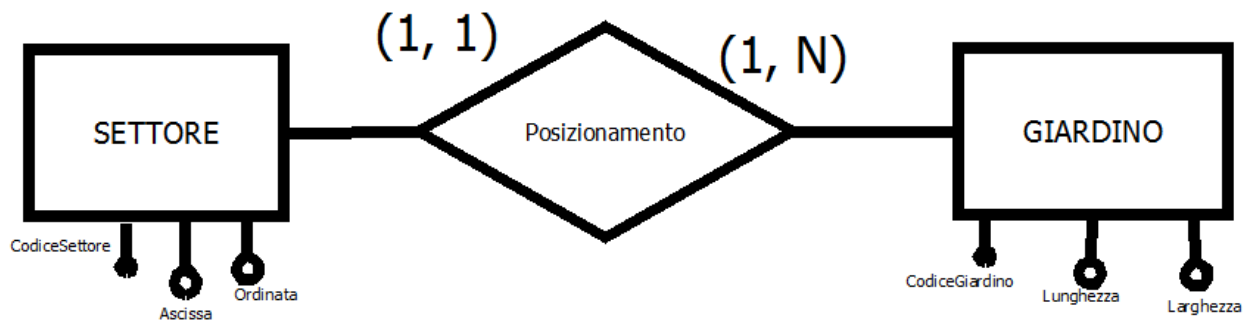
In questo caso risulta più conveniente accorpare la relazione nella tabella CURA, aggiungendo semplicemente un vincolo di integrità referenziale tra le due tabelle.



In questo caso risulta più conveniente accorpare la relazione nella tabella IMMAGINE, aggiungendo semplicemente un vincolo di integrità referenziale tra le due tabelle.

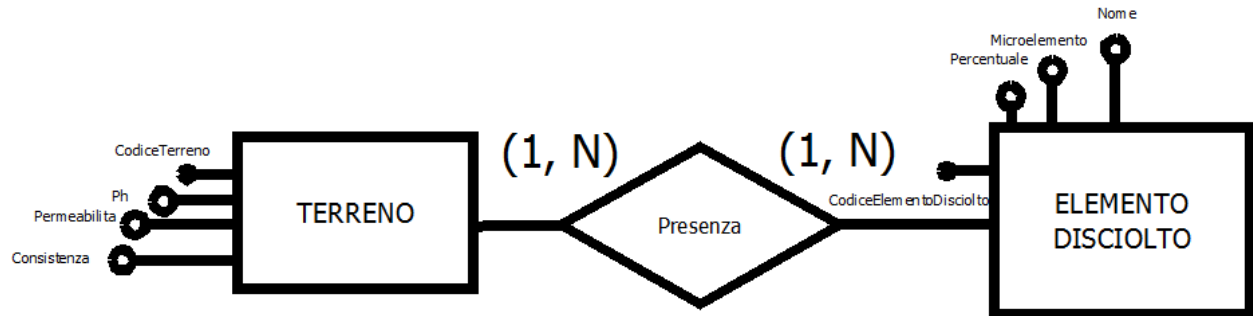


In questo caso risulta più conveniente accorpare la relazione nella tabella **PIANTA**, aggiungendo semplicemente un vincolo di integrità referenziale tra le due tabelle.



In questo caso risulta più conveniente accorpare la relazione nella tabella **SETTORE**, aggiungendo semplicemente un vincolo di integrità referenziale tra le due tabelle.

TRADUZIONE CARDINALITA' (1, N) – (1, N):

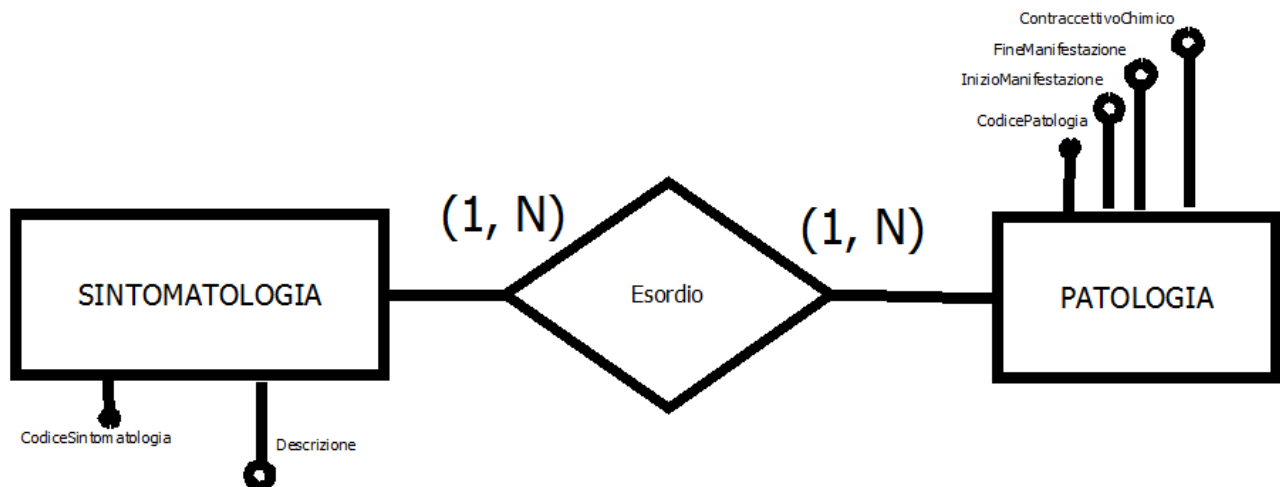


In questo caso è necessario predisporre tre tabelle legate da più vincoli di integrità referenziale:

ELEMENTO_DISCIOLTO (CodiceElementoDisciolto, Nome, Percentuale, Microelemento)

TERRENO (CodiceTerreno, Ph, Permeabilità, Consistenza)

PRESENZA (CodiceElementoDisciolto, CodiceTerreno)

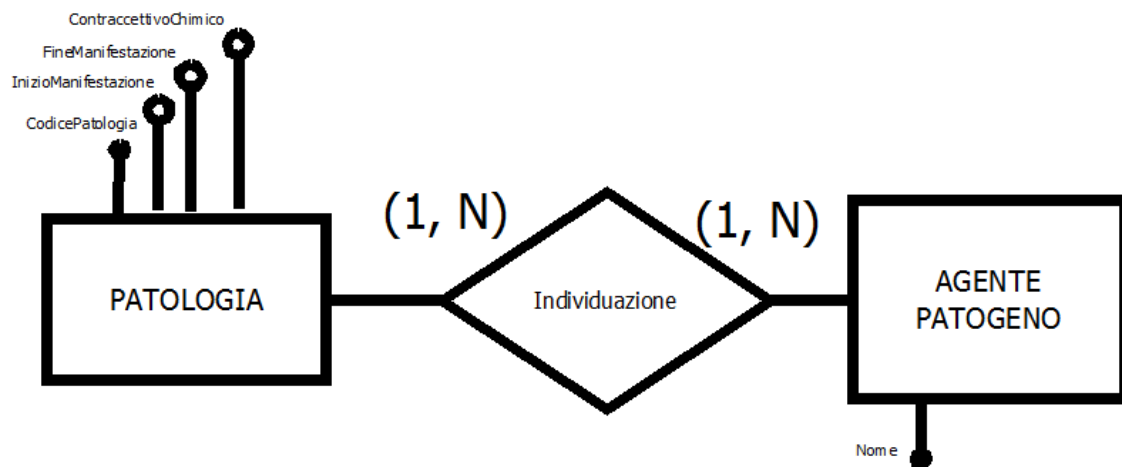


In questo caso è necessario predisporre tre tabelle legate da più vincoli di integrità referenziale:

SINTOMATOLOGIA (CodiceSintomatologia, Descrizione)

PATOLOGIA (CodicePatologia, InizioManifestazione, FineManifestazione, ContraccettivoChimico)

ESORDIO (CodiceSintomatologia, CodicePatologia)

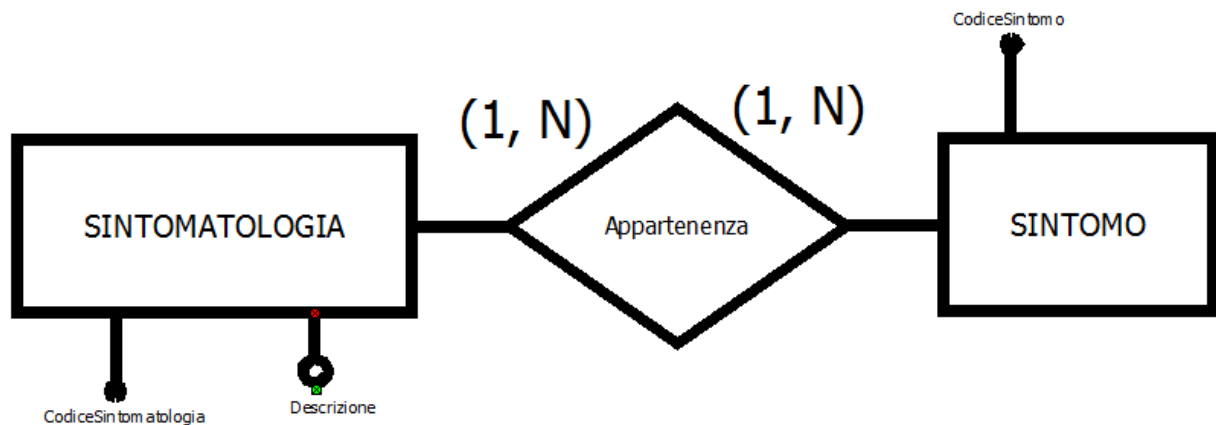


In questo caso è necessario predisporre tre tabelle legate da più vincoli di integrità referenziale:

AGENTE_PATOGENO (Nome)

PATOLOGIA (CodicePatologia, InizioManifestazione, FineManifestazione, ContraccettivoChimico)

INDIVIDUAZIONE (Nome, CodicePatologia)

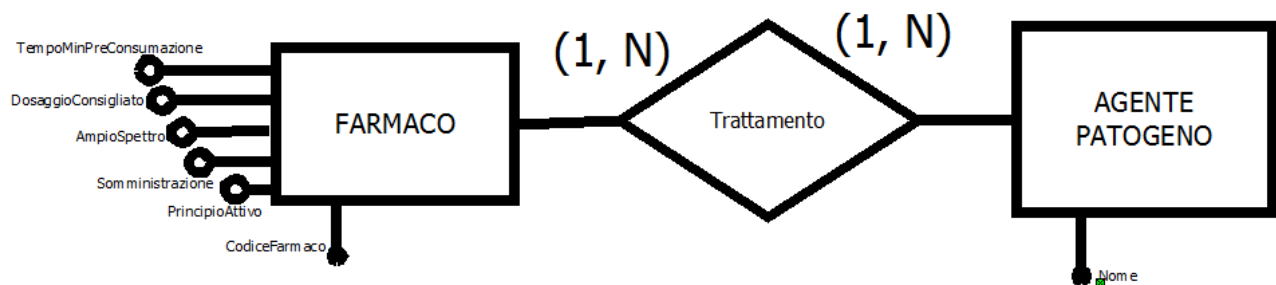


In questo caso è necessario predisporre tre tabelle legate da più vincoli di integrità referenziale:

SINTOMATOLOGIA (CodiceSintomatologia, Descrizione)

SINTOMO (CodiceSintomo)

APPARTENENZA (CodiceSintomo, CodiceSintomatologia)



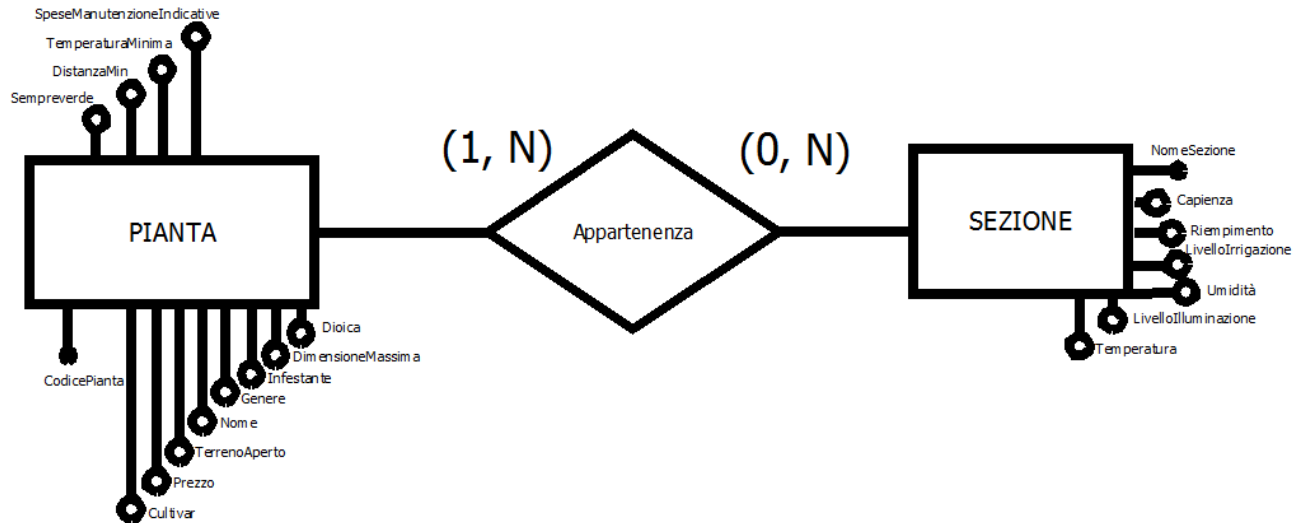
In questo caso è necessario predisporre tre tabelle legate da più vincoli di integrità referenziale:

AGENTE_PATOGENO (Nome)

FARMACO (CodiceFarmaco, TempoMinPreConsumazione, DosaggioConsigliato, AmpioSpettro, Somministrazione, PrincipioAttivo)

TRATTAMENTO (Nome, CodiceFarmaco)

TRADUZIONE CARDINALITA' (0, N) – (1, N):



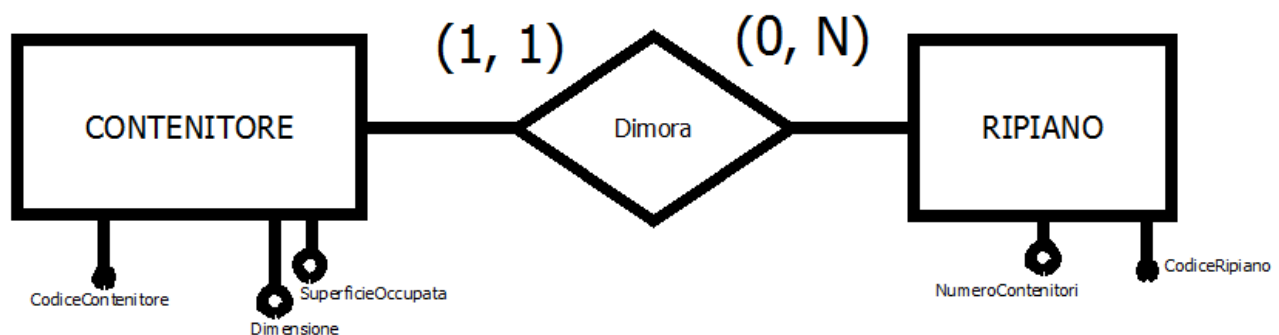
In questo caso è necessario predisporre tre tabelle legate da più vincoli di integrità referenziale:

PIANTA (CodicePianta, DistanzaMin, TemperaturaMinima, Sempreverde, SpeseManutenzioneIndicative, Dioica, DimensioneMassima, Infestante, Genere, Nome, TerrenoAperto, Prezzo, Cultivar, CodiceContenitore)

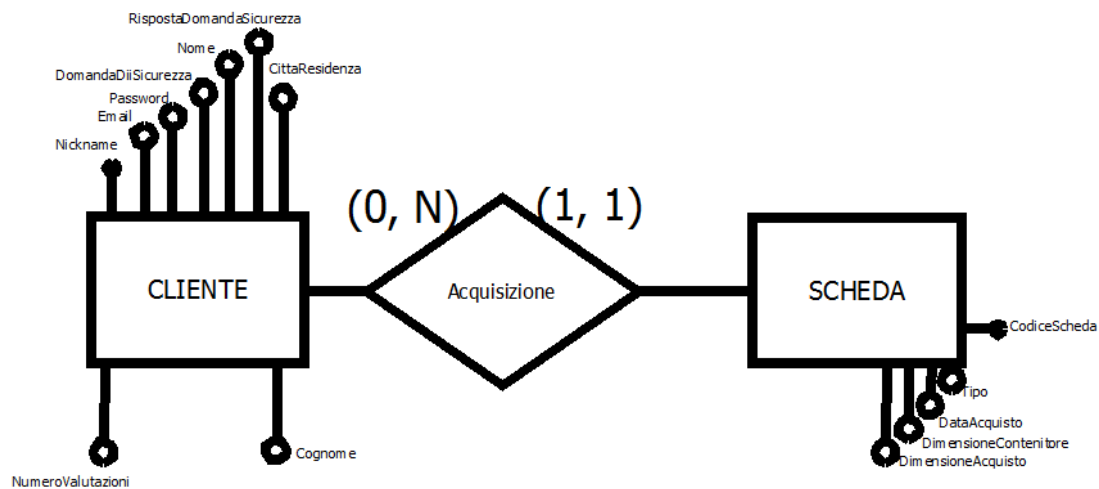
PATOLOGIA (CodicePatologia, InizioManifestazione, FineManifestazione, ContraccettivoChimico)

APPARTENENZA (CodicePatologia, CodicePianta)

TRADUZIONE DELLE CARDINALITA' (1, 1) – (0, N):



TRADUZIONE CARDINALITA' (0, N) – (1, 1):



In questo caso si è deciso di procedere utilizzando sono le due tabelle provenienti dalle entità SETTORE e VASO e di sdoppiare la relazione come segue:

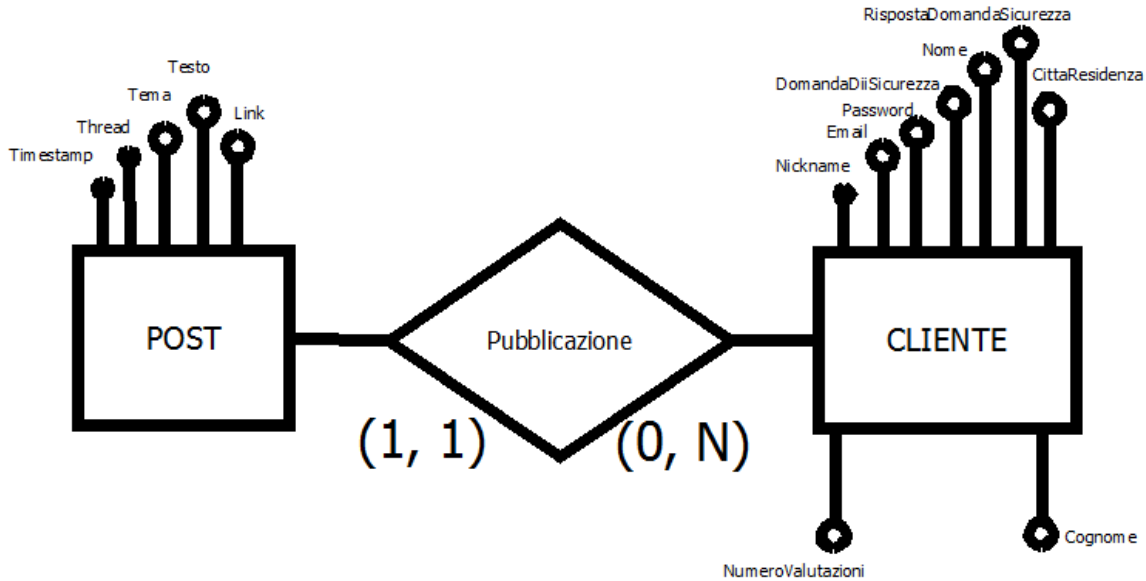
CLIENTE (Nickname, NumeroValutazioni, DomandaDiSicurezza, Nome, Email, Password, CittaResidenza, RispostaDomandaSicurezza, Cognome)

SCHEDA (CodiceScheda, Nickname, Tipo, DataAcquisto, DimensioneAcquisto, DimensioneContenitore)

In questo caso si è deciso di procedere utilizzando sono le due tabelle provenienti dalle entità SETTORE e VASO e di sdoppiare la relazione come segue:

CLIENTE (Nickname, NumeroValutazioni, DomandaDiSicurezza, Nome, Email, Password, CittaResidenza, RispostaDomandaSicurezza, Cognome)

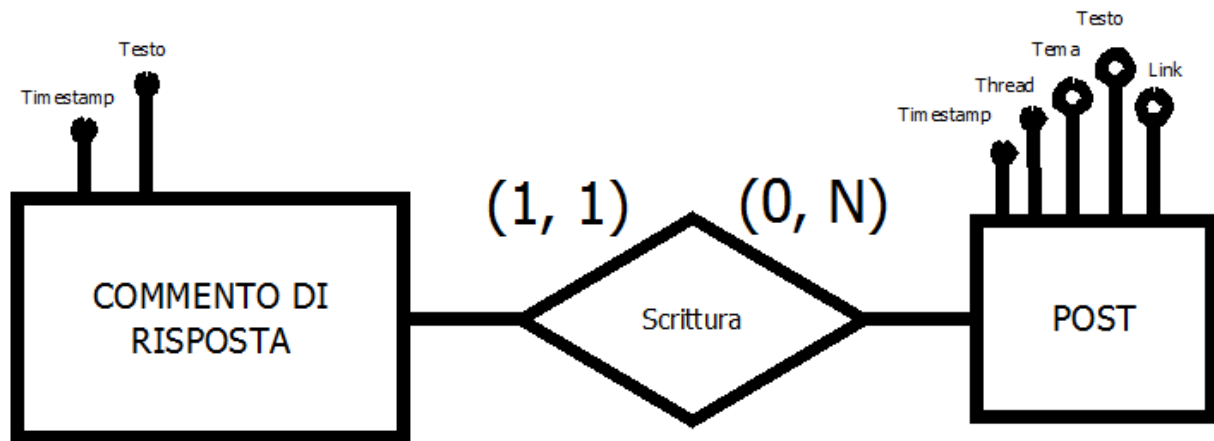
SCHEDA (CodiceScheda, Nickname, Tipo, DataAcquisto, DimensioneAcquisto, DimensioneContenitore)



In questo caso si è deciso di procedere utilizzando sono le due tabelle provenienti dalle entità SETTORE e VASO e di sdoppiare la relazione come segue:

CLIENTE (Nickname, NumeroValutazioni, DomandaDiSicurezza, Nome, Email, Password, CittaResidenza, RispostaDomandaSicurezza, Cognome)

POST (Timestamp, Thread, Tema, Testo, Link, Nickname)



In questo caso si è deciso di procedere utilizzando sono le due tabelle provenienti dalle entità SETTORE e VASO e di sdoppiare la relazione come segue:


COMMENTO_DI_RISPOSTA (Timestamp, Testo, TimestampPost, ThreadPost)

POST (Timestamp, Thread, Tema, Testo, Link, Nickname)

7. ANALISI DELLE DIPENDENZE FUNZIONALI E NORMALIZZAZIONE

Tabella PIANTA:

PIANTA (CodicePianta, Nome, Genere, Cultivar, DimensioneMassima, Accrescimento, DistanzaMin, Infestante, Luce, Terreno, Irrigazione, Sempreverde, Dioica, IndiceManutenzione, TerrenoAperto, Contenitore, Sezione, TemperaturaMinima, SpeseManutenzioneIndicative, Quarantena, Prezzo, Intervento, Fabbisogno, Fruttificazione, Fioritura)

 CodicePianta → Nome, Genere, Cultivar, DimensioneMassima, Accrescimento, FioritureAnnue, FruttificazioneAnnue, Luce, Terreno, Irrigazione, Concimazione, Sempreverde, Dioica, IndiceManutenzione, Intervento, TerrenoAperto, Contenitore, Sezione, TemperaturaMinima, SpeseManutenzioneIndicative, Distanza, Quarantena, Prezzo, Infestante, Fruttificazione, Fioritura

Poiché la parte sinistra è superchiave, PIANTA è in BCNF.

Tabella CONTENITORE:

CONTENITORE (CodiceContenitore, Ripiano, Dimensione, SuperficieOccupata)

 CodiceContenitore → Ripiano, Dimensione, SuperficieOccupata

Poiché la parte sinistra è superchiave, CONTENITORE è BCNF.

Tabella RIPIANO:


RIPIANO (CodiceRipiano, Sezione, NumeroContenitori)

 CodiceRipiano → Sezione, NumeroContenitori

Poiché la parte sinistra è superchiave, RIPIANO è BCNF.

Tabella SEZIONE:

SEZIONE (NomeSezione, Serra, Capienza, Riempimento, LivelloIrrigazione, LivelloIlluminazione, Temperatura, Umidita)

 NomeSezione → Serra, Capienza, Riempimento, LivelloIrrigazione, LivelloIlluminazione, Temperatura, Umidita

Poiché la parte sinistra è superchiave, SEZIONE è BCNF.

Tabella SERRA:

SERRA (NomeSerra, Indirizzo, Larghezza, Sede, Lunghezza, Altezza, PianteOspitabili)

 NomeSerra, Indirizzo → Sede, Larghezza, Lunghezza, Altezza, PianteOspitabili

Poiché la parte sinistra è superchiave, SERRA è BCNF.

Tabella SEDE:

SEDE (NomeSede, Indirizzo, NumeroDipendenti)

 NomeSede, Indirizzo → NumeroDipendenti

Poiché la parte sinistra è superchiave, SEDE è BCNF.

Tabella SINTOMATOLOGIA:

SINTOMATOLOGIA (CodiceSintomatologia, Descrizione)

 CodiceSintomatologia → Descrizione

Poiché la parte sinistra è superchiave, SINTOMATOLOGIA è BCNF.

Tabella SINTOMO:

SINTOMO (CodiceSintomo, Immagine, SintomatologiaDiAppartenenza)

 CodiceSintomo → Immagine, SintomatologiaDiAppartenenza

Poiché la parte sinistra è superchiave, SINTOMO è BCNF.

Tabella IMMAGINE:


IMMAGINE (CodiceImmagine, Pixel, SintomoRiscontrato, Dimensione)

 CodiceImmagine → Pixel, SintomoRiscontrato, Dimensione

Poiché la parte sinistra è superchiave, IMMAGINE è BCNF.

Tabella PATOLOGIA:


PATOLOGIA (CodicePatologia, AgentePatogeno, Sintomatologia, ContraccettivoChimico, PeriodoManifestazione)

 CodicePatologia → AgentePatogeno, Sintomatologia, ContraccettivoChimico, PeriodoManifestazione

Poiché la parte sinistra è superchiave, PATOLOGIA è BCNF.

Tabella FARMACO:


FARMACO (CodiceFarmaco, PrincipioAttivo, Somministrazione, AgentePatogenoCombattuto, AmpioSpettro, DosaggioConsigliato, Cura, TempoMinPreConsumazione, PeriodoNonSomministrazione)

 CodiceFarmaco → PrincipioAttivo, Somministrazione, AgentePatogenoCombattuto, AmpioSpettro, DosaggioConsigliato, Cura, TempoMinPreConsumazione, PeriodoNonSomministrazione

Poiché la parte sinistra è superchiave, FARMACO è BCNF.

Tabella CURA:


CURA (CodiceCura, Data, DosaggioEffettivo, Farmaco)

 CodiceCura → DosaggioEffettivo, Farmaco, Data

Poiché la parte sinistra è superchiave, CURA è BCNF.

Tabella ACCRESCIMENTO:


ACCRESCIMENTO (CodiceAccrescimento, IndiceAccrescimento, AccrescimentoTop, AccrescimentoRoot, Pianta)

 CodiceAccrescimento → IndiceAccrescimento, AccrescimentoTop, AccrescimentoRoot, Pianta

Poiché la parte sinistra è superchiave, ACCRESCIMENTO è BCNF.

Tabella LUCE:


LUCE (CodiceLuce, EsposizioneVegetativaMin, EsposizioneRiposoMin, Tipo, LuceDiretta)

 CodiceLuce → EsposizioneVegetativaMin, EsposizioneRiposoMin, Tipo, LuceDiretta

Poiché la parte sinistra è superchiave, Luce è BCNF.

Tabella IRRIGAZIONE:

IRRIGAZIONE (CodiceIrrigazione, QuantitaVegetativo, QuantitaRiposo, PeriodicitaVegetativa, PeriodicitaRiposo)

 CodiceIrrigazione → QuantitaVegetativo, QuantitaRiposo,, PeriodicitaVegetativa, PeriodicitaRiposo

Poiché la parte sinistra è superchiave, IRRIGAZIONE è BCNF.

Tabella RIVASO:


RINVASO (CodiceRinvaso, DimensioneVasoNuovo)

 CodiceRinvaso → DimensioneVasoNuovo

Poiché la parte sinistra è superchiave, RINVASO è BCNF.

Tabella CONCIMAZIONE:

CONCIMAZIONE (CodiceConcimazione, DataSomministrazione, NumeroSomministrazioni, Quantita, Periodicità, SomministrazioneRadicale)

 CodiceConcimazione → NumeroSomministrazioni, Quantita, Periodicità, SomministrazioneRadicale

Poiché la parte sinistra è superchiave, CONCIMAZIONE è BCNF

Tabella POTATURA:


POTATURA (CodicePotatura, Tipo, Periodo)

 CodicePotatura → Tipo, Periodo

Poiché la parte sinistra è superchiave, POTATURA è BCNF.

Tabella TERRENO:

TERRENO (CodiceTerreno, PH, Consistenza, Permeabilità)

 CodiceTerreno → PH, Consistenza, Permeabilità

Poiché la parte sinistra è superchiave, TERRENO è BCNF.

Tabella ELEMENTO_DISCIOLTO:


ELEMENTO_DISCIOLTO (CodiceElementoDisciolto, Nome, PercentualePresenza, Terreno, Microelemento, Concimazione)

 CodiceElementoDisciolto → Nome, Microelemento, Terreno, Concimazione, PercentualePresenza

Poiché la parte sinistra è superchiave, ELEMENTO_DISCIOLTO è BNFC.

Tabella ORDINE:


ORDINE (CodiceOrdine, Data, Acquirente, Pianta, Stato, Scheda)

 CodiceOrdine → Pianta, Stato, Scheda, Data, Acquirente

Poiché la parte sinistra è superchiave, ORDINE è BCNF.

Tabella SCHEDA:


SCHEDA (CodiceScheda, Pianta, DataAcquisto, Cliente, DimensioneAcquisto, DimensioneContenitore, Tipo, Prezzo, Ordine)

 CodiceScheda → DimensioneAcquisto, DimensioneContenitore, Cliente, DataAcquisto, Tipo, Prezzo, Pianta, Ordine

Poiché la parte sinistra è superchiave, SCHEDA è BCNF.

Tabella CLIENTE:


CLIENTE (Nickname, Email, Password, DomandaSicurezza, RispostaDomandaSicurezza, Nome, Cognome, CittàResidenza, NumeroValutazioni)

 Nickname → Password, DomandaSicurezza, RispostaDomandaSicurezza, Nome, Email, Cognome, CittàResidenza, NumeroValutazioni

Poiché la parte sinistra è superchiave, CLIENTE è BCNF.

Tabella GIARDINO:

GIARDINO (CodiceGiardino, Cliente, NumeroOggetti, NumeroSettori, Lunghezza, Larghezza)

 CodiceGiardino → NumeroOggetti, NumeroSettori, Lunghezza, Larghezza, Cliente

Poiché la parte sinistra è superchiave, GIARDINO è BCNF.

Tabella SETTORE:

SETTORE (CodiceSettore, Giardino, Ascissa, Ordinata)

 CodiceSettore → Giardino, Ascissa, Ordinata

Poiché la parte sinistra è superchiave, OGGETTO è BCNF.

Tabella VASO:


VASO (CodiceVaso, Settore, Ascissa, Ordinata)

 CodiceVaso → Settore, Ascissa, Ordinata

Poiché la parte sinistra è superchiave, OGGETTO è BCNF.

Tabella POST:


POST (Timestamp, Thread, Testo, Link, Nickname, Tema)

 TimeStamp, Thread → Testo, Link, Nickname, Tema

Poiché la parte sinistra è superchiave, POST è BCNF.

Tabella COMMENTO_DI_RISPOSTA:


COMMENTO_DI_RISPOSTA (Timestamp, Testo, PostDiRiferimento, Nickname, Valutazione)

 Timestamp, Testo → Nickname, Testo, Valutazione, PostDiRiferimento

Poiché la parte sinistra è superchiave, COMMENTO_DI_RISPOSTA è BCNF.

Tabella AGENTE_PATOGENO:


AGENTE_PATOGENO (Nome, Patologia, Farmaco)

 Nome → Patologia, Farmaco

Poiché la parte sinistra è superchiave, COMMENTO_DI_RISPOSTA è BCNF.

Tabella VALUTAZIONE:


VALUTAZIONE (CodiceValutazione, Commento, Punteggio, Nickname)

 CodiceValutazione → Commento, Punteggio, Nickname

Poiché la parte sinistra è superchiave, VALUTAZIONE è BCNF.

Tabella QUARANTENA:

QUARANTENA (DataInizio, Pianta, Cura, DataFine, Esito)

 Pianta, Cura, DataInizio → DataFine, Esito

Poiché la parte sinistra è superchiave, VALUTAZIONE è BCNF.

Tabella PERIODO:

PERIODO (CodicePeriodo, Tipo, MeseInizio, MeseFine)

 CodicePeriodo → Tipo, MeseInizio, MeseFine

Poiché la parte sinistra è superchiave, OGGETTO è BCNF.

Tabella FIORITURA:

VASO (CodiceFioritura, Pianta)

 CodiceFioritura → Pianta

Poiché la parte sinistra è superchiave, OGGETTO è BCNF.

Tabella FRUTTIFICAZIONE:

VASO (CodiceFruttificazione, Pianta)

 CodiceFruttificazione → Pianta

Poiché la parte sinistra è superchiave, OGGETTO è BCNF.

Tabella FABBISOGNO:


FABBISOGNO (CodiceFabbisogno, Tipo, CodiceTipo, PeriodoConsigliato)

 CodiceFabbisogno → Tipo, CodiceTipo, PeriodoConsigliato

Poiché la parte sinistra è superchiave, OGGETTO è BCNF.

Tabella INTERVENTO:


INTERVENTO (CodiceIntervento, Tipo, CodiceTipo, Data, StimaProssimaData)

 CodiceIntervento → Tipo, CodiceTipo, Data, StimaProssimaData

Poiché la parte sinistra è superchiave, OGGETTO è BCNF.

Tabella MANUTENZIONE_AUTOMATICA:


MANUTENZIONE_AUTOMATICA (CodiceManutenzioneAutomatica, Stato, PersonaleRichiesto, Scheda, Tipo, Prezzo)

 CodiceManutenzioneAutomatica → Stato, PersonaleRichiesto, Scheda, Tipo, Prezzo

Poiché la parte sinistra è superchiave, OGGETTO è BCNF.

Tabella MANUTENZIONE_PRIVATA:


MANUTENZIONE_PRIVATA (CodiceManutenzionePrivata, DataEsecuzione, Tipo, Scheda)

 CodiceManutenzionePrivata → DataEsecuzione, Tipo, Scheda

Poiché la parte sinistra è superchiave, OGGETTO è BCNF.

Tabella MANUTENZIONE_SU_RICHIESTA:


MANUTENZIONE_SU_RICHIESTA (CodiceManutenzioneSuRichiesta, Tipo, DataScadenza, Scheda, Stato, PersonaleRichiesto, Prezzo)

 CodiceManutenzioneSuRichiesta → Tipo, DataScadenza, Scheda, Stato, PersonaleRichiesto, Prezzo

Poiché la parte sinistra è superchiave, OGGETTO è BCNF.

Tabella MANUTENZIONE_PROGRAMMATA:

MANUTENZIONE_PROGRAMMATA (CodiceManutenzioneProgrammata, Scheda, Periodo, Tipo, PersonaleRichiesto, Prezzo, Periodicità)

 CodiceManutenzioneProgrammata → Scheda, Periodo, Tipo, PersonaleRichiesto, Prezzo, Periodicità

Poiché la parte sinistra è superchiave, OGGETTO è BCNF.

8. IMPLEMENTAZIONE SU DBMS ORACLE MYSQL

```

SET NAMES latin1;
SET FOREIGN_KEY_CHECKS = 0;

BEGIN;
CREATE DATABASE IF NOT EXISTS `PolliceVerde`;
COMMIT;

USE `PolliceVerde`;

-- -----
-- Table structure for `PERIODO`
-- -----
DROP TABLE IF EXISTS `PERIODO`;
CREATE TABLE `PERIODO` (
  `CodicePeriodo` INT(11) NOT NULL,
  `Tipo` ENUM('Fioritura', 'Fruttificazione', 'Fabbisogno', 'Manutenzione
programmata', 'Non Somministrazione', 'Patologia', 'Vegetazione', 'Riposo') NOT NULL,
  `MeseInizio` INT(2) NOT NULL,
  `MeseFine` INT(2) NOT NULL,
  PRIMARY KEY (`CodicePeriodo`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- -----
-- Records of `PERIODO`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`periodo` (`CodicePeriodo`, `Tipo`, `MeseInizio`,
`MeseFine`) VALUES ('1', '1', '1', '1');
INSERT INTO `polliceverde`.`periodo` (`CodicePeriodo`, `Tipo`, `MeseInizio`,
`MeseFine`) VALUES ('2', '2', '1', '2');
INSERT INTO `polliceverde`.`periodo` (`CodicePeriodo`, `Tipo`, `MeseInizio`,
`MeseFine`) VALUES ('3', '3', '1', '3');
INSERT INTO `polliceverde`.`periodo` (`CodicePeriodo`, `Tipo`, `MeseInizio`,
`MeseFine`) VALUES ('4', '3', '1', '4');
INSERT INTO `polliceverde`.`periodo` (`CodicePeriodo`, `Tipo`, `MeseInizio`,
`MeseFine`) VALUES ('5', '4', '1', '5');
INSERT INTO `polliceverde`.`periodo` (`CodicePeriodo`, `Tipo`, `MeseInizio`,
`MeseFine`) VALUES ('6', '5', '5', '6');
INSERT INTO `polliceverde`.`periodo` (`CodicePeriodo`, `Tipo`, `MeseInizio`,
`MeseFine`) VALUES ('7', '6', '1', '7');
INSERT INTO `polliceverde`.`periodo` (`CodicePeriodo`, `Tipo`, `MeseInizio`,
`MeseFine`) VALUES ('8', '7', '12', '12');
INSERT INTO `polliceverde`.`periodo` (`CodicePeriodo`, `Tipo`, `MeseInizio`,
`MeseFine`) VALUES ('9', '5', '2', '9');

```

```
INSERT INTO `polliceverde`.`periodo` (`CodicePeriodo`, `Tipo`, `MeseInizio`,
`MeseFine`) VALUES ('10', '8', '1', '10');
COMMIT;
```

```
-- -----
-- Table structure for `SEZIONE`
-- -----
DROP TABLE IF EXISTS `SEZIONE`;
CREATE TABLE `SEZIONE` (
  `CodiceSezione` INT(11) NOT NULL,
  `Serra` INT(11) NOT NULL,
    FOREIGN KEY (`Serra`) REFERENCES `PolliceVerde`.`SERRA` (`CodiceSerra`)
ON DELETE NO ACTION ON UPDATE CASCADE,
  `Nome` VARCHAR(45) NOT NULL,
  `Capienza` INT(3) NOT NULL,
  `Riempimento` INT(3) NOT NULL DEFAULT 0, /*quante piante contiene*/
  `Irrigazione` INT(11) NOT NULL,
    FOREIGN KEY (`Irrigazione`) REFERENCES `PolliceVerde`.`IRRIGAZIONE`
(`CodiceIrrigazione`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Illuminazione` INT(11) NOT NULL,
    FOREIGN KEY (`Illuminazione`) REFERENCES `PolliceVerde`.`LUCE`
(`CodiceLuce`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Temperatura` INT(3) NOT NULL,
  `Umidita` INT(3) NOT NULL,
  PRIMARY KEY (`CodiceSezione`),
  UNIQUE INDEX `Nome_UNIQUE` (`Nome` ASC)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `SEZIONE`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`sezione` (`CodiceSezione`, `Serra`, `Nome`, `Capienza`,
`Irrigazione`, `Illuminazione`, `Temperatura`, `Umidita`) VALUES ('1', '1', 'Io',
'50', '1', '1', '18', '30'), ('2', '1', 'Non', '50', '2', '1', '18', '32'), ('3', '1',
'Capisco', '50', '3', '1', '18', '34'), ('4', '1', 'Che', '50', '4', '1', '18', '36'),
('5', '1', 'Nome', '50', '5', '1', '18', '40'), ('6', '2', 'Si', '50', '1', '2', '24',
'20'), ('7', '2', 'Possa', '50', '2', '2', '24', '23'), ('8', '2', 'Dare', '50', '3',
'2', '24', '25'), ('9', '2', 'A una', '50', '4', '2', '24', '30'), ('10', '2', 'SEZIONE',
'50', '5', '2', '24', '15');
COMMIT;
```

```
-- -----
-- Table structure for `PIANTA`
-- -----
DROP TABLE IF EXISTS `PIANTA`;
CREATE TABLE `PIANTA` (
  `CodicePianta` INT(11) NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Genere` VARCHAR(45) NOT NULL,
  `Cultivar` TINYINT NOT NULL DEFAULT 0,
  `DimensioneMassima` INT(3) NOT NULL,
```

```

`Accrescimento` INT(5) NOT NULL
    REFERENCES `PolliceVerde`.`ACCRESIMENTO` (`CodiceAccrescimento`),
`Infestante` TINYINT NOT NULL DEFAULT 0,
`NumeroFioritureAnnue` INT(1) NOT NULL DEFAULT 0,
`NumeroFruttificazioniAnnue` INT(1) NOT NULL DEFAULT 0,
`Luce` INT(11) NOT NULL,
    FOREIGN KEY (`Luce`) REFERENCES `PolliceVerde`.`LUCE` (`CodiceLuce`) ON
DELETE NO ACTION ON UPDATE CASCADE,
`Terreno` INT(11) NOT NULL,
    FOREIGN KEY (`Terreno`) REFERENCES `PolliceVerde`.`TERRENO`
(`CodiceTerreno`) ON DELETE NO ACTION ON UPDATE CASCADE,
`Irrigazione` INT(11) NOT NULL,
    FOREIGN KEY (`Irrigazione`) REFERENCES `PolliceVerde`.`IRRIGAZIONE`
(`CodiceIrrigazione`) ON DELETE NO ACTION ON UPDATE CASCADE,
`Concimazione` INT(11) NOT NULL,
    FOREIGN KEY (`Concimazione`) REFERENCES `PolliceVerde`.`CONCIMAZIONE`
(`CodiceConcimazione`) ON DELETE NO ACTION ON UPDATE CASCADE,
`Manutenzione` INT(11) NOT NULL,
    FOREIGN KEY (`Manutenzione`) REFERENCES `PolliceVerde`.`MANUTENZIONE`
(`CodiceManutenzione`) ON DELETE NO ACTION ON UPDATE CASCADE,
`PeriodoVegetazione` INT(11) NOT NULL,
    FOREIGN KEY (`PeriodoVegetazione`) REFERENCES `PolliceVerde`.`PERIODO`
(`CodicePeriodo`) ON DELETE NO ACTION ON UPDATE CASCADE,
`PeriodoRiposo` INT(11) NOT NULL,
    FOREIGN KEY (`PeriodoRiposo`) REFERENCES `PolliceVerde`.`PERIODO`
(`CodicePeriodo`) ON DELETE NO ACTION ON UPDATE CASCADE,
`PeriodoFioritura` INT(11) NOT NULL,
    FOREIGN KEY (`PeriodoFioritura`) REFERENCES `PolliceVerde`.`PERIODO`
(`CodicePeriodo`) ON DELETE NO ACTION ON UPDATE CASCADE,
`PeriodoFruttificazione` INT(11) NOT NULL,
    FOREIGN KEY (`PeriodoFruttificazione`) REFERENCES
`PolliceVerde`.`PERIODO` (`CodicePeriodo`) ON DELETE NO ACTION ON UPDATE CASCADE,
`Sempreverde` TINYINT NOT NULL DEFAULT 0,
`Dioica` TINYINT NOT NULL DEFAULT 0,
`IndiceManutenzione` ENUM('Basso', 'Medio', 'Alto') NOT NULL,
`Contenitore` INT(11) NULL,
    FOREIGN KEY (`Contenitore`) REFERENCES `PolliceVerde`.`CONTENITORE`
(`CodiceContenitore`) ON DELETE NO ACTION ON UPDATE CASCADE,
`Sezione` INT(11) NOT NULL,
    FOREIGN KEY (`Sezione`) REFERENCES `PolliceVerde`.`SEZIONE`
(`CodiceSezione`) ON DELETE CASCADE ON UPDATE CASCADE,
`TemperaturaMinima` INT(11) NOT NULL,
`SpeseManutenzioneIndicative` INT(11) NOT NULL,
`Prezzo` FLOAT NOT NULL,
PRIMARY KEY (`CodicePianta`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

delimiter \$\$

```

CREATE TRIGGER `ValiditaPeriodoVegetazione` BEFORE INSERT ON `PIANTA` FOR EACH ROW
BEGIN
IF (SELECT Tipo
    FROM PERIODO

```

```

        WHERE CodicePeriodo = NEW.PeriodoVegetazione) <> 'Vegetazione' THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da vincolo di integrita referenziale con la tabella PERIODO';
END IF;
END $$
delimiter ;

delimiter $$
CREATE TRIGGER `ValiditaPeriodoRiposo` BEFORE INSERT ON `PIANTA` FOR EACH ROW BEGIN
IF (SELECT Tipo
    FROM PERIODO
    WHERE CodicePeriodo = NEW.PeriodoRiposo) <> 'Riposo' THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da vincolo di integrita referenziale con la tabella PERIODO';
END IF;
END $$
delimiter ;

delimiter $$
CREATE TRIGGER `ValiditaPeriodoFioritura` BEFORE INSERT ON `PIANTA` FOR EACH ROW BEGIN
IF (SELECT Tipo
    FROM PERIODO
    WHERE CodicePeriodo = NEW.PeriodoFioritura) <> 'Fioritura' THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da vincolo di integrita referenziale con la tabella PERIODO';
END IF;
END $$
delimiter ;

delimiter $$
CREATE TRIGGER `ValiditaPeriodoFruttificazione` BEFORE INSERT ON `PIANTA` FOR EACH ROW
BEGIN
IF (SELECT Tipo
    FROM PERIODO
    WHERE CodicePeriodo = NEW.PeriodoFruttificazione) <> 'Fruttificazione' THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da vincolo di integrita referenziale con la tabella PERIODO';
END IF;
END $$
delimiter ;

delimiter $$
CREATE TRIGGER `SecondaRidondanzaInsert` BEFORE INSERT ON `PIANTA` FOR EACH ROW BEGIN
SET @riempimento = (SELECT Riempimento
                    FROM SEZIONE
                    WHERE NEW.Sezione = CodiceSezione
                    );
SET @capienza = (SELECT Capienza
                FROM SEZIONE
                WHERE NEW.Sezione = CodiceSezione
                );

```

```

IF @capienza = @riempimento THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Impossibile eseguire procedura: SEZIONE GIA` PIENA';
ELSE
UPDATE `SEZIONE`
SET `Riempimento` = `Riempimento` + 1
WHERE `CodiceSezione` = NEW.Sezione;
END IF;
END $$
delimiter ;

delimiter $$
CREATE TRIGGER `SecondaRidondanzaDelete` AFTER DELETE ON `PIANTA` FOR EACH ROW BEGIN
UPDATE `SEZIONE`
SET `Riempimento` = `Riempimento` - 1
WHERE `CodiceSezione` = OLD.Sezione;
END $$
delimiter ;

-- -----
-- Records of `PIANTA`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`pianta` (`CodicePianta`, `Nome`, `Genere`, `Cultivar`,
`DimensioneMassima`, `Accrescimento`, `Infestante`, `NumeroFioritureAnnue`,
`NumeroFruttificazioniAnnue`, `Luce`, `Terreno`, `Irrigazione`, `Concimazione`,
`Manutenzione`, `PeriodoVegetazione`, `PeriodoRiposo`, `Sempreverde`, `Dioica`,
`IndiceManutenzione`, `Contenitore`, `Sezione`, `TemperaturaMinima`,
`SpeseManutenzioneIndicative`, `Prezzo`, `PeriodoFioritura`,
`PeriodoFruttificazione`) VALUES ('1', 'gladiolo', 'fiore', '1', '1', '10', '0', '1',
'3', '1', '1', '1', '1', '1', '8', '10', '0', '0', '1', '1', '5', '9', '19', '15', '1',
'2');
INSERT INTO `polliceverde`.`pianta` (`CodicePianta`, `Nome`, `Genere`, `Cultivar`,
`DimensioneMassima`, `Accrescimento`, `Infestante`, `NumeroFioritureAnnue`,
`NumeroFruttificazioniAnnue`, `Luce`, `Terreno`, `Irrigazione`, `Concimazione`,
`Manutenzione`, `PeriodoVegetazione`, `PeriodoRiposo`, `Sempreverde`, `Dioica`,
`IndiceManutenzione`, `Contenitore`, `Sezione`, `TemperaturaMinima`,
`SpeseManutenzioneIndicative`, `Prezzo`, `PeriodoFioritura`,
`PeriodoFruttificazione`) VALUES ('2', 'aloe', 'albero', '0', '3', '9', '1', '4', '2',
'2', '3', '2', '2', '2', '8', '10', '0', '0', '2', '2', '5', '7', '18', '18', '1', '2');
INSERT INTO `polliceverde`.`pianta` (`CodicePianta`, `Nome`, `Genere`, `Cultivar`,
`DimensioneMassima`, `Accrescimento`, `Infestante`, `NumeroFioritureAnnue`,
`NumeroFruttificazioniAnnue`, `Luce`, `Terreno`, `Irrigazione`, `Concimazione`,
`Manutenzione`, `PeriodoVegetazione`, `PeriodoRiposo`, `Sempreverde`, `Dioica`,
`IndiceManutenzione`, `Contenitore`, `Sezione`, `TemperaturaMinima`,
`SpeseManutenzioneIndicative`, `Prezzo`, `PeriodoFioritura`,
`PeriodoFruttificazione`) VALUES ('3', 'ginestra', 'fiore', '0', '5', '8', '0', '3',
'1', '3', '3', '3', '3', '3', '8', '10', '0', '0', '3', '3', '5', '6', '17', '26', '1',
'2');

```



```

INSERT INTO `polliceverde`.`pianta` (`CodicePianta`, `Nome`, `Genere`, `Cultivar`,
`DimensioneMassima`, `Accrescimento`, `Infestante`, `NumeroFioritureAnnu`,
`NumeroFruttificazioniAnnu`, `Luce`, `Terreno`, `Irrigazione`, `Concimazione`,
`Manutenzione`, `PeriodoVegetazione`, `PeriodoRiposo`, `Sempreverde`, `Dioica`,
`IndiceManutenzione`, `Contenitore`, `Sezione`, `TemperaturaMinima`,
`SpeseManutenzioneIndicative`, `Prezzo`, `PeriodoFioritura`,
`PeriodoFruttificazione`) VALUES ('4', 'girasole', 'fiore', '1', '1', '7', '1', '2',
'5', '4', '4', '4', '4', '4', '8', '10', '0', '1', '1', '4', '5', '5', '16', '3', '1',
'2');
INSERT INTO `polliceverde`.`pianta` (`CodicePianta`, `Nome`, `Genere`, `Cultivar`,
`DimensioneMassima`, `Accrescimento`, `Infestante`, `NumeroFioritureAnnu`,
`NumeroFruttificazioniAnnu`, `Luce`, `Terreno`, `Irrigazione`, `Concimazione`,
`Manutenzione`, `PeriodoVegetazione`, `PeriodoRiposo`, `Sempreverde`, `Dioica`,
`IndiceManutenzione`, `Contenitore`, `Sezione`, `TemperaturaMinima`,
`SpeseManutenzioneIndicative`, `Prezzo`, `PeriodoFioritura`,
`PeriodoFruttificazione`) VALUES ('5', 'crisantemo', 'fiore', '1', '1', '6', '0', '1',
'6', '5', '5', '5', '5', '5', '8', '10', '1', '1', '2', '5', '5', '4', '15', '43', '1',
'2');
INSERT INTO `polliceverde`.`pianta` (`CodicePianta`, `Nome`, `Genere`, `Cultivar`,
`DimensioneMassima`, `Accrescimento`, `Infestante`, `NumeroFioritureAnnu`,
`NumeroFruttificazioniAnnu`, `Luce`, `Terreno`, `Irrigazione`, `Concimazione`,
`Manutenzione`, `PeriodoVegetazione`, `PeriodoRiposo`, `Sempreverde`, `Dioica`,
`IndiceManutenzione`, `Contenitore`, `Sezione`, `TemperaturaMinima`,
`SpeseManutenzioneIndicative`, `Prezzo`, `PeriodoFioritura`,
`PeriodoFruttificazione`) VALUES ('6', 'cipresso', 'albero', '1', '15', '5', '1', '4',
'2', '6', '6', '6', '6', '6', '8', '10', '1', '0', '3', '6', '5', '3', '14', '7', '1',
'2');
INSERT INTO `polliceverde`.`pianta` (`CodicePianta`, `Nome`, `Genere`, `Cultivar`,
`DimensioneMassima`, `Accrescimento`, `Infestante`, `NumeroFioritureAnnu`,
`NumeroFruttificazioniAnnu`, `Luce`, `Terreno`, `Irrigazione`, `Concimazione`,
`Manutenzione`, `PeriodoVegetazione`, `PeriodoRiposo`, `Sempreverde`, `Dioica`,
`IndiceManutenzione`, `Contenitore`, `Sezione`, `TemperaturaMinima`,
`SpeseManutenzioneIndicative`, `Prezzo`, `PeriodoFioritura`,
`PeriodoFruttificazione`) VALUES ('7', 'ninfea', 'fiore', '0', '2', '4', '0', '3', '3',
'7', '7', '7', '7', '7', '8', '10', '1', '1', '1', '1', '5', '2', '13', '10', '1', '2');
INSERT INTO `polliceverde`.`pianta` (`CodicePianta`, `Nome`, `Genere`, `Cultivar`,
`DimensioneMassima`, `Accrescimento`, `Infestante`, `NumeroFioritureAnnu`,
`NumeroFruttificazioniAnnu`, `Luce`, `Terreno`, `Irrigazione`, `Concimazione`,
`Manutenzione`, `PeriodoVegetazione`, `PeriodoRiposo`, `Sempreverde`, `Dioica`,
`IndiceManutenzione`, `Contenitore`, `Sezione`, `TemperaturaMinima`,
`SpeseManutenzioneIndicative`, `Prezzo`, `PeriodoFioritura`,
`PeriodoFruttificazione`) VALUES ('8', 'viola', 'fiore', '1', '1', '3', '1', '2', '4',
'8', '8', '8', '8', '8', '8', '10', '0', '0', '3', '2', '5', '1', '12', '12', '1', '2');
INSERT INTO `polliceverde`.`pianta` (`CodicePianta`, `Nome`, `Genere`, `Cultivar`,
`DimensioneMassima`, `Accrescimento`, `Infestante`, `NumeroFioritureAnnu`,
`NumeroFruttificazioniAnnu`, `Luce`, `Terreno`, `Irrigazione`, `Concimazione`,
`Manutenzione`, `PeriodoVegetazione`, `PeriodoRiposo`, `Sempreverde`, `Dioica`,
`IndiceManutenzione`, `Contenitore`, `Sezione`, `TemperaturaMinima`,
`SpeseManutenzioneIndicative`, `Prezzo`, `PeriodoFioritura`,
`PeriodoFruttificazione`) VALUES ('9', 'enturium', 'fiore', '0', '3', '2', '0', '1',

```

```
'1', '9', '9', '9', '9', '9', '8', '10', '0', '1', '2', '3', '5', '0', '11', '6', '1',
'2');
INSERT INTO `polliceverde`.`pianta` (`CodicePianta`, `Nome`, `Genere`, `Cultivar`,
`DimensioneMassima`, `Accrescimento`, `Infestante`, `NumeroFioritureAnnu`,
`NumeroFruttificazioniAnnu`, `Luce`, `Terreno`, `Irrigazione`, `Concimazione`,
`Manutenzione`, `PeriodoVegetazione`, `PeriodoRiposo`, `Sempreverde`, `Dioica`,
`IndiceManutenzione`, `Contenitore`, `Sezione`, `TemperaturaMinima`,
`SpeseManutenzioneIndicative`, `Prezzo`, `PeriodoFioritura`,
`PeriodoFruttificazione`) VALUES ('10', 'rosa', 'fiore', '1', '1', '1', '0', '2', '2',
'10', '10', '10', '10', '10', '8', '10', '0', '1', '1', '1', '5', '5', '11', '7', '1',
'2');
COMMIT;
```

```
-- -----
-- Table structure for `QUARANTENA`
-- -----
DROP TABLE IF EXISTS `QUARANTENA`;
CREATE TABLE `QUARANTENA` (
  `CodiceQuarantena` INT(11) NOT NULL,
  `Pianta` INT(11) NOT NULL,
    FOREIGN KEY (`Pianta`) REFERENCES `PolliceVerde`.`PIANTA`
(`CodicePianta`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Cura` INT(11) NOT NULL,
    FOREIGN KEY (`Cura`) REFERENCES `PolliceVerde`.`CURA` (`CodiceCura`) ON
DELETE NO ACTION ON UPDATE CASCADE,
  `Patologia` INT(11) NOT NULL,
    FOREIGN KEY (`Patologia`) REFERENCES `PolliceVerde`.`PATOLOGIA`
(`CodicePatologia`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `DataInizio` DATE NOT NULL,
  `DataFine` DATE NULL,
  `Esito` ENUM ('Curata', 'Morta') NULL,
PRIMARY KEY (`CodiceQuarantena`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `QUARANTENA`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`quarantena` (`CodiceQuarantena`, `Pianta`, `Cura`,
`Patologia`, `DataInizio`, `DataFine`, `Esito`) VALUES ('1', '10', '7', '10',
'2016-12-15', '2016-12-20', '1'), ('4', '10', '2', '4', '2016-12-05', '2016-12-13',
'1'), ('6', '10', '8', '1', '2016-12-05', '2016-12-13', '1'), ('8', '5', '9', '8',
'2016-12-15', '2016-12-20', '1'), ('9', '5', '3', '5', '2016-12-15', '2016-12-20',
'1'), ('10', '1', '6', '6', '2016-12-28', '2017-01-09', '2');
INSERT INTO `polliceverde`.`quarantena` (`CodiceQuarantena`, `Pianta`,
`Cura`, `Patologia`, `DataInizio`) VALUES ('2', '6', '4', '10', '2017-02-05'), ('3',
'6', '1', '9', '2017-02-05'), ('5', '6', '5', '1', '2017-02-05'), ('7', '10', '10',
'6', '2017-02-05');
COMMIT;
```

```
-- -----
-- Table structure for `CONTENITORE`
```

```

-- -----
DROP TABLE IF EXISTS `CONTENITORE`;
CREATE TABLE `CONTENITORE` (
  `CodiceContenitore` INT(11) NOT NULL,
  `Ripiano` INT(11) NOT NULL,
  FOREIGN KEY (`Ripiano`) REFERENCES `PolliceVerde`.`RIPIANO`
  (`CodiceRipiano`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `SuperficieOccupata` INT(11) NOT NULL,
  `LivelloIdratazione` ENUM('basso', 'medio', 'alto') NOT NULL,
  PRIMARY KEY (`CodiceContenitore`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- -----
--   Records of `CONTENITORE`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`contenitore` (`CodiceContenitore`, `Ripiano`,
`SuperficieOccupata`, `LivelloIdratazione`) VALUES ('1', '1', '150', '1'), ('2', '1',
'150', '1'), ('3', '1', '150', '1'), ('4', '1', '150', '1'), ('5', '1', '150', '2'),
('6', '1', '150', '2'), ('7', '1', '150', '2'), ('8', '1', '150', '2'), ('9', '2', '100',
'3'), ('10', '2', '100', '3');
COMMIT;

-- -----
--   Table structure for `RIPIANO`
-- -----
DROP TABLE IF EXISTS `RIPIANO`;
CREATE TABLE `RIPIANO` (
  `CodiceRipiano` INT(11) NOT NULL,
  `Sezione` INT(11) NOT NULL,
  FOREIGN KEY (`Sezione`) REFERENCES `PolliceVerde`.`SEZIONE`
  (`CodiceSezione`) ON DELETE CASCADE ON UPDATE CASCADE,
  `N°contenitori` INT(2) NOT NULL DEFAULT 0,
  PRIMARY KEY (`CodiceRipiano`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- -----
--   Records of `RIPIANO`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`ripiano` (`CodiceRipiano`, `Sezione`, `N°contenitori`)
VALUES ('1', '1', '10'), ('2', '1', '10'), ('3', '1', '9'), ('4', '1', '7'), ('5', '1',
'3'), ('6', '2', '6'), ('7', '2', '6'), ('8', '2', '6'), ('9', '2', '5'), ('10', '2',
'3');
COMMIT;

-- -----
--   Table structure for `SERRA`
-- -----
DROP TABLE IF EXISTS `SERRA`;
CREATE TABLE `SERRA` (
  `CodiceSerra` INT(11) NOT NULL,
  `Sede` INT(11) NOT NULL,

```

```

        FOREIGN KEY (`Sede`) REFERENCES `PolliceVerde`.`SEDE` (`CodiceSede`) ON
DELETE NO ACTION ON UPDATE CASCADE,
    `Nome` VARCHAR(45) NOT NULL,
    `Indirizzo` VARCHAR(255) NOT NULL,
    `Larghezza` INT(3) NOT NULL,
    `Lunghezza` INT(3) NOT NULL,
    `Altezza` INT(3) NOT NULL,
    `PianteOspitabili` INT(3) NOT NULL,
    `PianteOspitate` INT(3) NOT NULL,
    PRIMARY KEY (`CodiceSerra`),
    UNIQUE INDEX `Nome_UNIQUE` (`Nome` ASC)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

-- -----
--   Records of `SERRA`
-- -----

```

```

BEGIN;
INSERT INTO `polliceverde`.`serra` (`CodiceSerra`, `Sede`, `Nome`, `Indirizzo`,
`Larghezza`, `Lunghezza`, `Altezza`, `PianteOspitabili`, `PianteOspitate`) VALUES
('1', '1', 'Susanna', 'Via Chiabrera 54', '25', '100', '8', '250', '150'), ('8', '2',
'Elvira', 'Via Terun 44', '30', '100', '10', '300', '200'), ('9', '2', 'Cristina', 'Via
Terun 45', '30', '100', '10', '300', '200'), ('10', '2', 'Carla', 'Via Dal Norde 28',
'50', '200', '15', '400', '300'), ('7', '2', 'Nicoletta', 'Via Terun 43', '30', '100',
'10', '300', '200'), ('6', '2', 'Giovanna', 'Via Terun 42', '30', '100', '10', '300',
'200'), ('5', '1', 'Maria', 'Via Chiabrera 58', '25', '100', '8', '250', '150'), ('4',
'1', 'Teresa', 'Via Chiabrera 57', '25', '100', '8', '250', '150'), ('3', '1',
'Francesca', 'Via Chiabrera 56', '25', '100', '8', '250', '150'), ('2', '1', 'Alberta',
'Via Chiabrera 55', '25', '100', '8', '250', '150');
COMMIT;

```

```

-- -----
--   Table structure for `SEDE`
-- -----

```

```

DROP TABLE IF EXISTS `SEDE`;
CREATE TABLE `SEDE` (
    `CodiceSede` INT(11) NOT NULL,
    `Nome` VARCHAR(45) NOT NULL,
    `Indirizzo` VARCHAR(255) NOT NULL,
    `NumeroDipendenti` INT(4) NOT NULL,
    PRIMARY KEY (`CodiceSede`, `Nome`, `Indirizzo`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;

```

```

-- -----
--   Records of `SEDE`
-- -----

```

```

BEGIN;
INSERT INTO `polliceverde`.`sede` (`CodiceSede`, `Nome`, `Indirizzo`,
`NumeroDipendenti`) VALUES ('1', 'ROMAGARDEN', 'Via Chiabrera 54', '56'), ('2',
'PADANIAGARDEN', 'Via Terun 42', '48'), ('3', 'CALABRIAGARDEN', 'Via Diotisalvi 9',
'43'), ('4', 'SICILIAGARDEN', 'Via Garibaldi 41', '52'), ('5', 'MAREMMAGARDEN',
'Piazza Vittorio Emanuele II 16', '45'), ('6', 'AOSTAGARDEN', 'Via Barco 36', '41'),
('7', 'TIROLOGARDEN', 'Via Inventata 9', '54'), ('8', 'ABBRUZZOGARDEN', 'Via Don Bosco

```

```
57', '50'), ('9', 'SARDEGNAGARDEN', 'Via Eja 121', '49'), ('10', 'SEMPREGARDEN', 'Via
Damasco 72', '48'));
COMMIT;
```

```
-- -----
-- Table structure for `SINTOMATOLOGIA`
-- -----
DROP TABLE IF EXISTS `SINTOMATOLOGIA`;
CREATE TABLE `SINTOMATOLOGIA` (
  `CodiceSintomatologia` INT(11) NOT NULL,
  `Descrizione` TEXT NOT NULL,
  `PiantaColpita` INT(11) NOT NULL,
  FOREIGN KEY (`PiantaColpita`) REFERENCES `PolliceVerde`.`pianta`
  (`CodicePianta`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodiceSintomatologia`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `SINTOMATOLOGIA`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`sintomatologia` (`CodiceSintomatologia`, `Descrizione`,
`PiantaColpita`) VALUES ('1', 'arricciamento delle foglie', '10'), ('2', 'caduta
foglie', '9'), ('3', 'cambio di colore', '8'), ('4', 'cambio di turgidità', '7'), ('5',
'interruzione fioritura', '6'), ('6', 'interruzione fruttificazione', '5'), ('7',
'crescita stentata', '4'), ('8', 'interruzione fioritura, interruzione
fruttificazione', '3'), ('9', 'arricciamento delle foglie, caduta foglie', '2'),
('10', 'arricciamento delle foglie, cambio di colore', '1');
COMMIT;
```

```
-- -----
-- Table structure for `SINTOMO`
-- -----
DROP TABLE IF EXISTS `SINTOMO`;
CREATE TABLE `SINTOMO` (
  `CodiceSintomo` INT(11) NOT NULL,
  `SintomatologiaDiAppartenenza` INT(11) NOT NULL,
  FOREIGN KEY (`SintomatologiaDiAppartenenza`) REFERENCES
  `PolliceVerde`.`SINTOMATOLOGIA` (`CodiceSintomatologia`) ON DELETE NO ACTION ON
  UPDATE CASCADE,
  PRIMARY KEY (`CodiceSintomo`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `SINTOMO`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`sintomo` (`CodiceSintomo`,
`SintomatologiaDiAppartenenza`) VALUES ('1', '9'), ('2', '9'), ('3', '1'), ('4', '6'),
('5', '8'), ('6', '4'), ('7', '5'), ('8', '4'), ('9', '8'), ('10', '5');
COMMIT;
```

```
-- Table structure for `IMMAGINE`
-- -----
DROP TABLE IF EXISTS `IMMAGINE`;
CREATE TABLE `IMMAGINE` (
  `CodiceImmagine` INT(11) NOT NULL,
  `NumeroPixel` (x 10^5) DECIMAL NOT NULL,
  `SintomoRiscontrato` INT(11) NOT NULL,
  FOREIGN KEY (`SintomoRiscontrato`) REFERENCES `PolliceVerde`.`SINTOMO`
  (`CodiceSintomo`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Dimensione` (KB) DECIMAL NOT NULL,
  PRIMARY KEY (`CodiceImmagine`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- -----
-- Records of `IMMAGINE`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`immagine` (`CodiceImmagine`, `NumeroPixel` (x 10^5),
`SintomoRiscontrato`, `Dimensione` (KB)) VALUES ('1', '49', '1', '1600'), ('2', '50',
'1', '1700'), ('3', '45', '2', '1500'), ('4', '51', '2', '1800'), ('5', '52', '3',
'1900'), ('6', '43', '3', '1400'), ('7', '42', '4', '1300'), ('8', '49', '4', '1600'),
('9', '49', '5', '1600'), ('10', '49', '5', '1600');
COMMIT;

-- -----
-- Table structure for `PATOLOGIA`
-- -----
DROP TABLE IF EXISTS `PATOLOGIA`;
CREATE TABLE `PATOLOGIA` (
  `CodicePatologia` INT(11) NOT NULL,
  `Contraccettivo` ENUM('Chimico', 'Biologico') NOT NULL,
  `Sintomatologia` INT(11) NOT NULL,
  FOREIGN KEY (`Sintomatologia`) REFERENCES
`PolliceVerde`.`SINTOMATOLOGIA` (`CodiceSintomatologia`) ON DELETE NO ACTION ON
UPDATE CASCADE,
  `PeriodoInCuiColpisce` INT(11) NOT NULL,
  FOREIGN KEY (`PeriodoInCuiColpisce`) REFERENCES `PolliceVerde`.`PERIODO`
  (`CodicePeriodo`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodicePatologia`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

delimiter $$
CREATE TRIGGER `ValiditaPeriodoPatologia` BEFORE INSERT ON `PATOLOGIA` FOR EACH ROW
BEGIN
  IF (SELECT Tipo
      FROM PERIODO
      WHERE CodicePeriodo = NEW.PeriodoInCuiColpisce) <> 'Patologia' THEN
  SIGNAL SQLSTATE '45000'
  SET MESSAGE_TEXT = 'Errore da vincolo di integrita referenziale con la tabella PERIODO';
  END IF;
END $$
delimiter ;
```

```
-- -----
-- Records of `PATOLOGIA`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`patologia` (`CodicePatologia`, `Contraccettivo`,
`Sintomatologia`, `PeriodoInCuiColpisce`) VALUES ('1', 'Chimico', '10', '7');
INSERT INTO `polliceverde`.`patologia` (`CodicePatologia`, `Contraccettivo`,
`Sintomatologia`, `PeriodoInCuiColpisce`) VALUES ('2', 'Biologico', '9', '7');
INSERT INTO `polliceverde`.`patologia` (`CodicePatologia`, `Contraccettivo`,
`Sintomatologia`, `PeriodoInCuiColpisce`) VALUES ('3', 'Chimico', '8', '7');
INSERT INTO `polliceverde`.`patologia` (`CodicePatologia`, `Contraccettivo`,
`Sintomatologia`, `PeriodoInCuiColpisce`) VALUES ('4', 'Biologico', '7', '7');
INSERT INTO `polliceverde`.`patologia` (`CodicePatologia`, `Contraccettivo`,
`Sintomatologia`, `PeriodoInCuiColpisce`) VALUES ('5', 'Chimico', '6', '7');
INSERT INTO `polliceverde`.`patologia` (`CodicePatologia`, `Contraccettivo`,
`Sintomatologia`, `PeriodoInCuiColpisce`) VALUES ('6', 'Biologico', '5', '7');
INSERT INTO `polliceverde`.`patologia` (`CodicePatologia`, `Contraccettivo`,
`Sintomatologia`, `PeriodoInCuiColpisce`) VALUES ('7', 'Chimico', '4', '7');
INSERT INTO `polliceverde`.`patologia` (`CodicePatologia`, `Contraccettivo`,
`Sintomatologia`, `PeriodoInCuiColpisce`) VALUES ('8', 'Biologico', '3', '7');
INSERT INTO `polliceverde`.`patologia` (`CodicePatologia`, `Contraccettivo`,
`Sintomatologia`, `PeriodoInCuiColpisce`) VALUES ('9', 'Chimico', '2', '7');
INSERT INTO `polliceverde`.`patologia` (`CodicePatologia`, `Contraccettivo`,
`Sintomatologia`, `PeriodoInCuiColpisce`) VALUES ('10', 'Biologico', '1', '7');
COMMIT;
```

```
-- -----
-- Table structure for `AGENTEPATOGENO`
-- -----
DROP TABLE IF EXISTS `AGENTEPATOGENO`;
CREATE TABLE `AGENTEPATOGENO` (
  `CodiceAgentePatogeno` INT(11) NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Patologia` INT(11) NOT NULL,
    FOREIGN KEY (`Patologia`) REFERENCES `PolliceVerde`.`PATOLOGIA`
(`CodicePatologia`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Farmaco` INT(11) NOT NULL,
    FOREIGN KEY (`Farmaco`) REFERENCES `PolliceVerde`.`FARMACO`
(`CodiceFarmaco`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodiceAgentePatogeno`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `AGENTEPATOGENO`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`agentepatogeno` (`CodiceAgentePatogeno`, `Nome`,
`Patologia`, `Farmaco`) VALUES ('1', 'ponderosae ', '10', '2'), ('2', 'ljungiana', '9',
'3'), ('3', 'pulchellana', '8', '1'), ('4', 'fragariae', '7', '5'), ('5', 'pyri', '6',
'4'), ('6', 'sarta', '5', '6'), ('7', 'phloeocoptes', '4', '9'), ('8', 'erinea', '3',
'7'), ('9', 'sheldoni ', '2', '8'), ('10', 'tulipae', '1', '10');
COMMIT;
```

```

-- -----
-- Table structure for `FARMACO`
-- -----
DROP TABLE IF EXISTS `FARMACO`;
CREATE TABLE `FARMACO` (
  `CodiceFarmaco` INT(11) NOT NULL,
  `PrincipioAttivo` VARCHAR(45) NOT NULL,
  `Somministrazione` ENUM('Nebulizzazione', 'Irrigazione', 'Nebulizzazione e
Irrigazione') NOT NULL,
  `AmpioSpettro` TINYINT NOT NULL DEFAULT 0,
  `DosaggioConsigliato` INT(3) NOT NULL,
  `TempoMinPostsomministrazione` INT(3) NOT NULL,
  `PeriodoNonSomministrazione` INT(11) NOT NULL,
    FOREIGN KEY (`PeriodoNonSomministrazione`) REFERENCES
`PolliceVerde`.`PERIODO` (`CodicePeriodo`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Cura` INT(11) NOT NULL,
    FOREIGN KEY (`Cura`) REFERENCES `PolliceVerde`.`CURA` (`CodiceCura`) ON
DELETE NO ACTION ON UPDATE CASCADE,
  `AgentePatogenoCombattuto` INT(11) NOT NULL,
    FOREIGN KEY (`AgentePatogenoCombattuto`) REFERENCES
`PolliceVerde`.`AGENTEPATOGENO` (`CodiceAgentePatogeno`) ON DELETE NO ACTION ON
UPDATE CASCADE,
  PRIMARY KEY (`CodiceFarmaco`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- -----
-- Records of `FARMACO`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`farmaco` (`CodiceFarmaco`, `PrincipioAttivo`,
`Somministrazione`, `AmpioSpettro`, `DosaggioConsigliato`,
`TempoMinPostsomministrazione`, `PeriodoNonSomministrazione`, `Cura`,
`AgentePatogenoCombattuto`) VALUES ('1', 'Acido acetilsalicilico', 'Nebulizzazione',
'1', '3', '30', '6', '10', '7');
INSERT INTO `polliceverde`.`farmaco` (`CodiceFarmaco`, `PrincipioAttivo`,
`Somministrazione`, `AmpioSpettro`, `DosaggioConsigliato`,
`TempoMinPostsomministrazione`, `PeriodoNonSomministrazione`, `Cura`,
`AgentePatogenoCombattuto`) VALUES ('2', 'Aciclovir', 'Nebulizzazione', '0', '2',
'60', '9', '9', '4');
INSERT INTO `polliceverde`.`farmaco` (`CodiceFarmaco`, `PrincipioAttivo`,
`Somministrazione`, `AmpioSpettro`, `DosaggioConsigliato`,
`TempoMinPostsomministrazione`, `PeriodoNonSomministrazione`, `Cura`,
`AgentePatogenoCombattuto`) VALUES ('3', 'Betametasone', 'Nebulizzazione', '1', '3',
'15', '6', '8', '1');
INSERT INTO `polliceverde`.`farmaco` (`CodiceFarmaco`, `PrincipioAttivo`,
`Somministrazione`, `AmpioSpettro`, `DosaggioConsigliato`,
`TempoMinPostsomministrazione`, `PeriodoNonSomministrazione`, `Cura`,
`AgentePatogenoCombattuto`) VALUES ('4', 'Biperidene', 'Irrigazione', '0', '1', '30',
'9', '7', '10');
INSERT INTO `polliceverde`.`farmaco` (`CodiceFarmaco`, `PrincipioAttivo`,
`Somministrazione`, `AmpioSpettro`, `DosaggioConsigliato`,

```



```
`TempoMinPostsomministrazione`, `PeriodoNonSomministrazione`, `Cura`,
`AgentePatogenoCombattuto`) VALUES ('5', 'Felbamato', 'Irrigazione', '1', '2', '30',
'6', '6', '8');
INSERT INTO `polliceverde`.`farmaco` (`CodiceFarmaco`, `PrincipioAttivo`,
`Somministrazione`, `AmpioSpettro`, `DosaggioConsigliato`,
`TempoMinPostsomministrazione`, `PeriodoNonSomministrazione`, `Cura`,
`AgentePatogenoCombattuto`) VALUES ('6', 'Levetiracetam', 'Irrigazione', '0', '2',
'15', '9', '5', '5');
INSERT INTO `polliceverde`.`farmaco` (`CodiceFarmaco`, `PrincipioAttivo`,
`Somministrazione`, `AmpioSpettro`, `DosaggioConsigliato`,
`TempoMinPostsomministrazione`, `PeriodoNonSomministrazione`, `Cura`,
`AgentePatogenoCombattuto`) VALUES ('7', 'Modafinil', 'Nebulizzazione e Irrigazione',
'1', '2', '15', '6', '1', '2');
INSERT INTO `polliceverde`.`farmaco` (`CodiceFarmaco`, `PrincipioAttivo`,
`Somministrazione`, `AmpioSpettro`, `DosaggioConsigliato`,
`TempoMinPostsomministrazione`, `PeriodoNonSomministrazione`, `Cura`,
`AgentePatogenoCombattuto`) VALUES ('8', 'Naltrexone', 'Nebulizzazione e
Irrigazione', '0', '3', '60', '9', '4', '3');
INSERT INTO `polliceverde`.`farmaco` (`CodiceFarmaco`, `PrincipioAttivo`,
`Somministrazione`, `AmpioSpettro`, `DosaggioConsigliato`,
`TempoMinPostsomministrazione`, `PeriodoNonSomministrazione`, `Cura`,
`AgentePatogenoCombattuto`) VALUES ('9', 'Propafenone', 'Nebulizzazione e
Irrigazione', '1', '1', '30', '6', '3', '9');
INSERT INTO `polliceverde`.`farmaco` (`CodiceFarmaco`, `PrincipioAttivo`,
`Somministrazione`, `AmpioSpettro`, `DosaggioConsigliato`,
`TempoMinPostsomministrazione`, `PeriodoNonSomministrazione`, `Cura`,
`AgentePatogenoCombattuto`) VALUES ('10', 'Zonisamide', 'Nebulizzazione', '0', '3',
'30', '9', '2', '6');
COMMIT;
```

```
-- -----
-- Table structure for `CURA`
-- -----
```

```
DROP TABLE IF EXISTS `CURA`;
CREATE TABLE `CURA` (
  `CodiceCura` INT(11) NOT NULL,
  `Data` DATE NOT NULL,
  `DosaggioEffettivo` INT(3) NOT NULL,
  `Farmaco` INT(11) NOT NULL,
  FOREIGN KEY (`Farmaco`) REFERENCES `PolliceVerde`.`FARMACO`
(`CodiceFarmaco`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodiceCura`, `Data`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `CURA`
-- -----
```

```
BEGIN;
INSERT INTO `polliceverde`.`cura` (`CodiceCura`, `Data`, `DosaggioEffettivo`,
`Farmaco`) VALUES ('1', '2016-12-05', '1', '10'), ('2', '2016-08-05', '1', '9'), ('3',
'2015-06-23', '1', '8'), ('4', '2016-12-25', '2', '7'), ('5', '2016-04-11', '2', '6'),
```

```
('6', '2015-02-01', '2', '5'), ('7', '2014-07-20', '3', '4'), ('8', '2016-08-21', '3', '3'), ('9', '2016-06-22', '3', '2'), ('10', '2015-04-15', '3', '1');
COMMIT;
```

```
-- -----
-- Table structure for `ACCRESIMENTO`
-- -----
```

```
DROP TABLE IF EXISTS `ACCRESIMENTO`;
CREATE TABLE `ACCRESIMENTO` (
  `CodiceAccrescimento` INT(11) NOT NULL,
  `IndiceAccrescimento` DOUBLE GENERATED ALWAYS AS
(GREATEST(AccrescimentoTop,AccrescimentoRoot)) VIRTUAL,
  `AccrescimentoTop` DOUBLE NOT NULL,
  `AccrescimentoRoot` DOUBLE NOT NULL,
  `Pianta` INT(11) NOT NULL,
  FOREIGN KEY (`Pianta`) REFERENCES `PolliceVerde`.`PIANTA`
(`CodicePianta`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodiceAccrescimento`) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `ACCRESIMENTO`
-- -----
```

```
BEGIN;
INSERT INTO `polliceverde`.`accrescimento` (`CodiceAccrescimento`,
`AccrescimentoTop`, `AccrescimentoRoot`, `Pianta`) VALUES ('1', '1.4', '0.6', '1'),
('2', '0.4', '0.6', '2'), ('3', '1', '3', '3'), ('4', '6', '2', '4'), ('5', '8', '2', '5'), ('6', '3', '0.7', '6'), ('7', '1.1', '0.5', '7'), ('8', '3', '3', '8'), ('9', '1.8', '1.8', '9'), ('10', '0.6', '0.8', '10');
COMMIT;
```

```
-- -----
-- Table structure for `LUCE`
-- -----
```

```
DROP TABLE IF EXISTS `LUCE`;
CREATE TABLE `LUCE` (
  `CodiceLuce` INT(11) NOT NULL,
  `EsposizioneVegetativaMin` INT(2) NOT NULL DEFAULT 0, /*ore al giorno*/
  `EsposizioneRiposoMin` INT(2) NOT NULL DEFAULT 0,
  `Tipo` ENUM('Luce', 'Penombra', 'Ombra') NOT NULL,
  `LuceDiretta` TINYINT NOT NULL DEFAULT 0,
  PRIMARY KEY (`CodiceLuce`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `LUCE`
-- -----
```

```
BEGIN;
INSERT INTO `polliceverde`.`luce` (`CodiceLuce`, `EsposizioneVegetativaMin`,
`EsposizioneRiposoMin`, `Tipo`, `LuceDiretta`) VALUES ('1', '5', '3', '1', '1'), ('2', '6', '4', '2', '0'), ('3', '7', '5', '3', '1'), ('4', '8', '5', '1', '0'), ('5', '9', '4', '2', '1'), ('6', '10', '6', '3', '1'), ('7', '6', '5', '2', '0'), ('8', '7', '6', '1', '0'), ('9', '4', '4', '3', '0'), ('10', '7', '7', '1', '1');
```

COMMIT;

```
-- -----
-- Table structure for `IRRIGAZIONE`
-- -----
```

```
DROP TABLE IF EXISTS `IRRIGAZIONE`;
CREATE TABLE `IRRIGAZIONE` (
  `CodiceIrrigazione` INT(11) NOT NULL,
  `QuantitaVegetativo` ENUM('Basso', 'Medio', 'Alto') NOT NULL,
  `QuantitaRiposo` ENUM('Basso', 'Medio', 'Alto') NOT NULL,
  `PeriodicitaVegetativa` INT(1) NOT NULL DEFAULT 0, /*giorni alla settimana*/
  `PeriodicitaRiposo` INT(1) NOT NULL DEFAULT 0,
  PRIMARY KEY (`CodiceIrrigazione`) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `IRRIGAZIONE`
-- -----
```

```
BEGIN;
INSERT INTO `polliceverde`.`irrigazione` (`CodiceIrrigazione`, `QuantitaVegetativo`,
`QuantitaRiposo`, `PeriodicitaVegetativa`, `PeriodicitaRiposo`) VALUES ('1', '1',
'1', '7', '5');
INSERT INTO `polliceverde`.`irrigazione` (`CodiceIrrigazione`, `QuantitaVegetativo`,
`QuantitaRiposo`, `PeriodicitaVegetativa`, `PeriodicitaRiposo`) VALUES ('2', '2',
'1', '5', '3');
INSERT INTO `polliceverde`.`irrigazione` (`CodiceIrrigazione`, `QuantitaVegetativo`,
`QuantitaRiposo`, `PeriodicitaVegetativa`, `PeriodicitaRiposo`) VALUES ('3', '3',
'1', '3', '1');
INSERT INTO `polliceverde`.`irrigazione` (`CodiceIrrigazione`, `QuantitaVegetativo`,
`QuantitaRiposo`, `PeriodicitaVegetativa`, `PeriodicitaRiposo`) VALUES ('4', '1',
'1', '6', '4');
INSERT INTO `polliceverde`.`irrigazione` (`CodiceIrrigazione`, `QuantitaVegetativo`,
`QuantitaRiposo`, `PeriodicitaVegetativa`, `PeriodicitaRiposo`) VALUES ('5', '2',
'2', '5', '2');
INSERT INTO `polliceverde`.`irrigazione` (`CodiceIrrigazione`, `QuantitaVegetativo`,
`QuantitaRiposo`, `PeriodicitaVegetativa`, `PeriodicitaRiposo`) VALUES ('6', '3',
'2', '4', '3');
INSERT INTO `polliceverde`.`irrigazione` (`CodiceIrrigazione`, `QuantitaVegetativo`,
`QuantitaRiposo`, `PeriodicitaVegetativa`, `PeriodicitaRiposo`) VALUES ('7', '1',
'1', '7', '6');
INSERT INTO `polliceverde`.`irrigazione` (`CodiceIrrigazione`, `QuantitaVegetativo`,
`QuantitaRiposo`, `PeriodicitaVegetativa`, `PeriodicitaRiposo`) VALUES ('8', '2',
'2', '5', '2');
INSERT INTO `polliceverde`.`irrigazione` (`CodiceIrrigazione`, `QuantitaVegetativo`,
`QuantitaRiposo`, `PeriodicitaVegetativa`, `PeriodicitaRiposo`) VALUES ('9', '3',
'3', '3', '1');
INSERT INTO `polliceverde`.`irrigazione` (`CodiceIrrigazione`, `QuantitaVegetativo`,
`QuantitaRiposo`, `PeriodicitaVegetativa`, `PeriodicitaRiposo`) VALUES ('10', '1',
'1', '7', '3');
COMMIT;
```

```
-- -----
```

```
-- Table structure for `CONCIMAZIONE`
-- -----
DROP TABLE IF EXISTS `CONCIMAZIONE`;
CREATE TABLE `CONCIMAZIONE` (
  `CodiceConcimazione` INT(11) NOT NULL,
  `Somministrazione` ENUM('Aerea', 'Radicale') NOT NULL,
  `Quantita` FLOAT NOT NULL, /*totale*/
  `NumeroSomministrazioni` INT(2) NOT NULL DEFAULT 1,
  `Periodicita` INT(2) NULL,
  PRIMARY KEY (`CodiceConcimazione`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- -----
-- Records of `CONCIMAZIONE`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`concimazione` (`CodiceConcimazione`, `Somministrazione`,
`Quantita`, `NumeroSomministrazioni`, `Periodicita`) VALUES ('1', '1', '1', '2', '1'),
('2', '2', '1', '1', '1'), ('3', '1', '1', '3', '3'), ('4', '2', '1.5', '1', '2'), ('5',
'1', '0.5', '3', '1'), ('6', '2', '2', '1', '3'), ('7', '1', '1.5', '2', '1'), ('8',
'2', '2', '1', '2'), ('9', '1', '0.5', '3', '1'), ('10', '2', '2', '1', '3');
COMMIT;

-- -----
-- Table structure for `RINVASO`
-- -----
DROP TABLE IF EXISTS `RINVASO`;
CREATE TABLE `RINVASO` (
  `CodiceRinvaso` INT(11) NOT NULL,
  `DimensioneVasoNuovo` INT(11) NULL,
  PRIMARY KEY (`CodiceRinvaso`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- -----
-- Records of `RINVASO`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`rinvaso` (`CodiceRinvaso`, `DimensioneVasoNuovo`) VALUES
('1', '50');
INSERT INTO `polliceverde`.`rinvaso` (`CodiceRinvaso`, `DimensioneVasoNuovo`) VALUES
('2', '100');
INSERT INTO `polliceverde`.`rinvaso` (`CodiceRinvaso`, `DimensioneVasoNuovo`) VALUES
('3', '45');
INSERT INTO `polliceverde`.`rinvaso` (`CodiceRinvaso`, `DimensioneVasoNuovo`) VALUES
('4', '50');
INSERT INTO `polliceverde`.`rinvaso` (`CodiceRinvaso`, `DimensioneVasoNuovo`) VALUES
('5', '60');
INSERT INTO `polliceverde`.`rinvaso` (`CodiceRinvaso`, `DimensioneVasoNuovo`) VALUES
('6', '85');
INSERT INTO `polliceverde`.`rinvaso` (`CodiceRinvaso`, `DimensioneVasoNuovo`) VALUES
('7', '55');
INSERT INTO `polliceverde`.`rinvaso` (`CodiceRinvaso`, `DimensioneVasoNuovo`) VALUES
('8', '15');
```

```
INSERT INTO `polliceverde`.`rinvaso` (`CodiceRinvaso`, `DimensioneVasoNuovo`) VALUES
('9', '30');
INSERT INTO `polliceverde`.`rinvaso` (`CodiceRinvaso`, `DimensioneVasoNuovo`) VALUES
('10', '20');
COMMIT;
```

```
-- -----
-- Table structure for `POTATURA`
-- -----
DROP TABLE IF EXISTS `POTATURA`;
CREATE TABLE `POTATURA` (
  `CodicePotatura` INT(10) NOT NULL,
  `Tipo` ENUM('CONTENIMENTO DIMENSIONI', 'AUMENTO DI FIORI E FRUTTI', 'RIMOZIONI DI
PARTI DANNEGGIATE', 'INTENSA') NOT NULL,
  `Perido` ENUM('Vegetativo', 'Riposo', 'Sempre'),
  PRIMARY KEY (`CodicePotatura`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `POTATURA`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`potatura` (`CodicePotatura`, `Tipo`, `Perido`) VALUES
('1', '1', '1'), ('2', '1', '2'), ('3', '1', '3'), ('4', '2', '1'), ('5', '2', '2'),
('6', '2', '3'), ('7', '3', '1'), ('8', '3', '2'), ('9', '3', '3'), ('10', '4', '1');
COMMIT;
```

```
-- -----
-- Table structure for `FABBISOGNO`
-- -----
DROP TABLE IF EXISTS `FABBISOGNO`;
CREATE TABLE `FABBISOGNO` (
  `CodiceFabbisogno` INT(11) NOT NULL,
  `Tipo` ENUM('Concimazione', 'Rinvaso', 'Potatura') NOT NULL,
  `CodiceTipo` INT(11) NOT NULL,
  `PeriodoConsigliato` INT(11) NOT NULL,
  FOREIGN KEY (`PeriodoConsigliato`) REFERENCES `PolliceVerde`.`PERIODO`
(`CodicePeriodo`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Pianta` INT(11) NOT NULL,
  FOREIGN KEY (`Pianta`) REFERENCES `PolliceVerde`.`PIANTA`
(`CodicePianta`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodiceFabbisogno`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
delimiter $$
CREATE TRIGGER `ControlloCodiceTipo` BEFORE INSERT ON `FABBISOGNO` FOR EACH ROW BEGIN
CASE
WHEN NEW.Tipo = 'Concimazione' THEN
IF NEW.CodiceTipo NOT IN ( SELECT CodiceConcimazione
FROM CONCIMAZIONE) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Intervento di concimazione inesistente';
END IF;
```

```

WHEN NEW.Tipo = 'Rinvaso' THEN
IF NEW.CodiceTipo NOT IN ( SELECT CodiceRinvaso
                           FROM RINVASO) THEN

SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Intervento di rinvaso inesistente';
END IF;
WHEN NEW.Tipo = 'Potatura' THEN
IF NEW.CodiceTipo NOT IN ( SELECT CodicePotatura
                           FROM POTATURA) THEN

SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Intervento di potatura inesistente';
END IF;
END CASE;
END $$
delimiter ;

delimiter $$
CREATE TRIGGER `ValiditaPeriodoFabbisogno` BEFORE INSERT ON `FABBISOGNO` FOR EACH ROW
BEGIN
IF (SELECT Tipo
    FROM PERIODO
    WHERE CodicePeriodo = NEW.PeriodoConsigliato) <> 'Fabbisogno' THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da vincolo di integrita referenziale con la tabella PERIODO';
END IF;
END $$
delimiter ;

-- -----
--  Records of `FABBISOGNO`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`fabbisogno` (`CodiceFabbisogno`, `Tipo`, `CodiceTipo`,
`PeriodoConsigliato`, `Pianta`) VALUES ('1', 'Concimazione', '1', '4', '1');
INSERT INTO `polliceverde`.`fabbisogno` (`CodiceFabbisogno`, `Tipo`, `CodiceTipo`,
`PeriodoConsigliato`, `Pianta`) VALUES ('2', 'Concimazione', '2', '3', '1');
INSERT INTO `polliceverde`.`fabbisogno` (`CodiceFabbisogno`, `Tipo`, `CodiceTipo`,
`PeriodoConsigliato`, `Pianta`) VALUES ('3', 'Concimazione', '3', '4', '1');
INSERT INTO `polliceverde`.`fabbisogno` (`CodiceFabbisogno`, `Tipo`, `CodiceTipo`,
`PeriodoConsigliato`, `Pianta`) VALUES ('4', 'Rinvaso', '1', '3', '5');
INSERT INTO `polliceverde`.`fabbisogno` (`CodiceFabbisogno`, `Tipo`, `CodiceTipo`,
`PeriodoConsigliato`, `Pianta`) VALUES ('5', 'Rinvaso', '2', '4', '1');
INSERT INTO `polliceverde`.`fabbisogno` (`CodiceFabbisogno`, `Tipo`, `CodiceTipo`,
`PeriodoConsigliato`, `Pianta`) VALUES ('6', 'Rinvaso', '3', '3', '1');
INSERT INTO `polliceverde`.`fabbisogno` (`CodiceFabbisogno`, `Tipo`, `CodiceTipo`,
`PeriodoConsigliato`, `Pianta`) VALUES ('7', 'Concimazione', '9', '4', '2');
INSERT INTO `polliceverde`.`fabbisogno` (`CodiceFabbisogno`, `Tipo`, `CodiceTipo`,
`PeriodoConsigliato`, `Pianta`) VALUES ('8', 'Potatura', '1', '3', '1');
INSERT INTO `polliceverde`.`fabbisogno` (`CodiceFabbisogno`, `Tipo`, `CodiceTipo`,
`PeriodoConsigliato`, `Pianta`) VALUES ('9', 'Potatura', '2', '4', '2');

```

```
INSERT INTO `polliceverde`.`fabbisogno` (`CodiceFabbisogno`, `Tipo`, `CodiceTipo`,
`PeriodoConsigliato`, `Pianta`) VALUES ('10', 'Potatura', '5', '3', '3');
COMMIT;
```

```
-- -----
-- Table structure for `INTERVENTO`
-- -----
```

```
DROP TABLE IF EXISTS `INTERVENTO`;
CREATE TABLE `INTERVENTO` (
  `CodiceIntervento` INT(11) NOT NULL,
  `Tipo` ENUM('Concimazione', 'Rinvaso', 'Potatura') NOT NULL,
  `CodiceTipo` INT(11) NOT NULL,
  `Data` DATE NOT NULL,
  `DataStimaProssimo` DATE NULL,
  `Pianta` INT(11) NOT NULL,
  FOREIGN KEY (`Pianta`) REFERENCES `PolliceVerde`.`PIANTA`
(`CodicePianta`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodiceIntervento`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
delimiter $$
CREATE TRIGGER `ControlloCodiceTipo2` BEFORE INSERT ON `INTERVENTO` FOR EACH ROW BEGIN
CASE
WHEN NEW.Tipo = 'Concimazione' THEN
IF NEW.CodiceTipo NOT IN ( SELECT CodiceConcimazione
                           FROM CONCIMAZIONE) THEN

SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Intervento di concimazione inesistente';
END IF;
WHEN NEW.Tipo = 'Rinvaso' THEN
IF NEW.CodiceTipo NOT IN ( SELECT CodiceRinvaso
                           FROM RINVASO) THEN

SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Intervento di rinvaso inesistente';
END IF;
WHEN NEW.Tipo = 'Potatura' THEN
IF NEW.CodiceTipo NOT IN ( SELECT CodicePotatura
                           FROM POTATURA) THEN

SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Intervento di potatura inesistente';
END IF;
END CASE;
END $$
delimiter ;
```

```
delimiter $$
CREATE TRIGGER `ControlloData` BEFORE INSERT ON `INTERVENTO` FOR EACH ROW BEGIN
IF NEW.Data >= NEW.DataStimaProssimo THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da inserimento della data';
END IF;
END $$
```

```

delimiter ;
-- -----
-- Records of `INTERVENTO`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`intervento` (`CodiceIntervento`, `Tipo`, `CodiceTipo`,
`Data`, `DataStimaProssimo`, `Pianta`) VALUES ('1', '1', '6', '2016-01-01',
'2016-03-01', '6');
INSERT INTO `polliceverde`.`intervento` (`CodiceIntervento`, `Tipo`, `CodiceTipo`,
`Data`, `DataStimaProssimo`, `Pianta`) VALUES ('2', '1', '5', '2016-01-01',
'2016-03-01', '9');
INSERT INTO `polliceverde`.`intervento` (`CodiceIntervento`, `Tipo`, `CodiceTipo`,
`Data`, `DataStimaProssimo`, `Pianta`) VALUES ('3', '1', '8', '2016-01-01',
'2016-03-01', '5');
INSERT INTO `polliceverde`.`intervento` (`CodiceIntervento`, `Tipo`, `CodiceTipo`,
`Data`, `DataStimaProssimo`, `Pianta`) VALUES ('4', '1', '2', '2016-02-01',
'2016-08-01', '1');
INSERT INTO `polliceverde`.`intervento` (`CodiceIntervento`, `Tipo`, `CodiceTipo`,
`Data`, `DataStimaProssimo`, `Pianta`) VALUES ('5', '2', '2', '2016-07-01',
'2017-07-01', '1');
INSERT INTO `polliceverde`.`intervento` (`CodiceIntervento`, `Tipo`, `CodiceTipo`,
`Data`, `DataStimaProssimo`, `Pianta`) VALUES ('6', '2', '3', '2016-08-01',
'2016-09-01', '5');
INSERT INTO `polliceverde`.`intervento` (`CodiceIntervento`, `Tipo`, `CodiceTipo`,
`Data`, `DataStimaProssimo`, `Pianta`) VALUES ('7', '2', '1', '2016-08-01',
'2016-09-01', '4');
INSERT INTO `polliceverde`.`intervento` (`CodiceIntervento`, `Tipo`, `CodiceTipo`,
`Data`, `DataStimaProssimo`, `Pianta`) VALUES ('8', '2', '5', '2016-08-01',
'2016-09-01', '3');
INSERT INTO `polliceverde`.`intervento` (`CodiceIntervento`, `Tipo`, `CodiceTipo`,
`Data`, `DataStimaProssimo`, `Pianta`) VALUES ('9', '3', '7', '2016-08-15',
'2016-10-01', '2');
INSERT INTO `polliceverde`.`intervento` (`CodiceIntervento`, `Tipo`, `CodiceTipo`,
`Data`, `DataStimaProssimo`, `Pianta`) VALUES ('10', '3', '10', '2016-08-15',
'2016-10-01', '1');
COMMIT;

-- -----
-- Table structure for `TERRENO`
-- -----
DROP TABLE IF EXISTS `TERRENO`;
CREATE TABLE `TERRENO` (
  `CodiceTerreno` INT(11) NOT NULL,
  `PH` INT(2) NOT NULL DEFAULT 7,
  `Consistenza` ENUM('Alto', 'Medio', 'Basso') NOT NULL,
  `Permeabilita` ENUM('Alto', 'Medio', 'Basso', 'Inesistente') NOT NULL,
  PRIMARY KEY (`CodiceTerreno`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- -----
-- Records of `TERRENO`
-- -----

```



```

BEGIN;
INSERT INTO `polliceverde`.`terreno` (`CodiceTerreno`, `PH`, `Consistenza`,
`Permeabilita`) VALUES ('1', '5', '1', '1'), ('2', '6', '1', '2'), ('3', '1', '1', '3'),
('4', '0', '2', '4'), ('5', '7', '2', '1'), ('6', '9', '2', '2'), ('7', '13', '3', '3'),
('8', '6', '3', '4'), ('9', '8', '3', '1'), ('10', '4', '1', '2');
COMMIT;

-- -----
-- Table structure for `ELEMENTODISCIOLTO`
-- -----
DROP TABLE IF EXISTS `ELEMENTODISCIOLTO`;
CREATE TABLE `ELEMENTODISCIOLTO` (
  `CodiceElementoDisciolto` INT(11) NOT NULL,
  `PercentualePresenza` INT(3) NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Terreno` INT(11) NULL,
    FOREIGN KEY (`Terreno`) REFERENCES `PolliceVerde`.`TERRENO`
(`CodiceTerreno`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Concimazione` INT(11) NULL,
    FOREIGN KEY (`Concimazione`) REFERENCES `PolliceVerde`.`CONCIMAZIONE`
(`CodiceConcimazione`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Microelemento` TINYINT NOT NULL DEFAULT 1,
  PRIMARY KEY (`CodiceElementoDisciolto`, `PercentualePresenza`)) ENGINE=InnoDB
DEFAULT CHARSET=latin1;

delimiter $$
CREATE TRIGGER `ValiditaElemento` BEFORE INSERT ON `ELEMENTODISCIOLTO` FOR EACH ROW
BEGIN
SET @controllopercentuale = (SELECT SUM(PercentualePresenza)
                                FROM ELEMENTODISCIOLTO
                                WHERE Terreno = NEW.Terreno
                                );
IF NEW.`Nome` IN ( SELECT Nome
                    FROM ELEMENTODISCIOLTO
                    WHERE Terreno = NEW.Terreno) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore! Elemento già inserito in questo terreno';
ELSEIF NEW.`Nome` IN ( SELECT Nome
                        FROM ELEMENTODISCIOLTO
                        WHERE Concimazione = NEW.Concimazione) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore! Elemento già inserito in questo intervento di
concimazione';
ELSEIF (NEW.`PercentualePresenza` + @controllopercentuale > 100) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore! Percentuale complessiva superiore al 100%';
END IF;
END $$
delimiter ;

-- -----
-- Records of `ELEMENTODISCIOLTO`

```

```

-----
BEGIN;
INSERT INTO `polliceverde`.`elementodisciolto` (`CodiceElementoDisciolto`,
`PercentualePresenza`, `Terreno`, `Concimazione`, `Microelemento`, `Nome`) VALUES
('1', '10', '10', '10', '1', 'Boro'), ('2', '33', '1', '9', '0', 'Azoto'), ('3', '33',
'1', '8', '1', 'Ferro'), ('4', '33', '1', '7', '1', 'Zinco'), ('5', '25', '3', '6',
'0', 'Fosforo'), ('6', '25', '4', '5', '0', 'Ferro'), ('7', '25', '2', '4', '1',
'Ferro'), ('8', '25', '3', '3', '0', 'Potassio'), ('9', '50', '2', '2', '1', 'Azoto'),
('10', '50', '5', '1', '0', 'Ferro');
COMMIT;

-- -----
-- Table structure for `ORDINE`
-- -----
DROP TABLE IF EXISTS `ORDINE`;
CREATE TABLE `ORDINE` (
  `CodiceOrdine` INT(11) NOT NULL,
  `Data` DATE NOT NULL,
  `Acquirente` INT(11) NOT NULL,
    FOREIGN KEY (`Acquirente`) REFERENCES `PolliceVerde`.`CLIENT`
(`CodiceCliente`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Stato` ENUM('In processazione', 'In preparazione', 'Spedito', 'Evaso', 'Pendente')
NOT NULL,
  `Scheda` INT(11) NOT NULL,
    FOREIGN KEY (`Scheda`) REFERENCES `PolliceVerde`.`SCHEDA`
(`CodiceScheda`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Pianta` INT(11) NOT NULL,
    FOREIGN KEY (`Pianta`) REFERENCES `PolliceVerde`.`PIANTA`
(`CodicePianta`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodiceOrdine`, `Data`, `Acquirente`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;

-- -----
-- Records of `ORDINE`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`ordine` (`CodiceOrdine`, `Data`, `Acquirente`, `Stato`,
`Scheda`, `Pianta`) VALUES ('1', '2015-04-15', '10', '1', '7', '5'), ('2',
'2016-06-22', '9', '1', '10', '6'), ('3', '2016-08-21', '8', '2', '9', '10'), ('4',
'2014-07-20', '7', '2', '8', '5'), ('5', '2015-02-01', '6', '3', '4', '6'), ('6',
'2016-04-11', '5', '3', '6', '4'), ('7', '2016-12-25', '4', '4', '5', '8'), ('8',
'2015-06-23', '3', '4', '1', '5'), ('9', '2016-08-05', '2', '5', '3', '10'), ('10',
'2016-12-05', '1', '5', '2', '7');
COMMIT;

-- -----
-- Table structure for `SCHEDA`
-- -----
DROP TABLE IF EXISTS `SCHEDA`;
CREATE TABLE `SCHEDA` (
  `CodiceScheda` INT(11) NOT NULL,

```

```

`NomePianta` VARCHAR(45) NULL,
`CodicePianta` INT(11) NULL,
    FOREIGN KEY (`CodicePianta`) REFERENCES `PolliceVerde`.`PIANTA`
(`CodicePianta`) ON DELETE NO ACTION ON UPDATE CASCADE,
`DataAcquisto` DATE NOT NULL,
`Cliente` INT(11) NOT NULL,
    FOREIGN KEY (`Cliente`) REFERENCES `PolliceVerde`.`CLIENT`
(`CodiceCliente`) ON DELETE NO ACTION ON UPDATE CASCADE,
`DimensioneContenitore` INT(11) NULL,
`DimensioneAcquisto` INT(3) NOT NULL,
`TerrenoAperto` TINYINT(1) NULL DEFAULT 0,
`Tipo` ENUM('Ordine', 'Manutenzione Automatica', 'Manutenzione Programmata',
'Manutenzione Privata', 'Manutenzione su richiesta') NOT NULL,
`Prezzo` FLOAT NOT NULL,
PRIMARY KEY (`CodiceScheda`, `Cliente`, `DataAcquisto`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;

```

delimiter \$\$

```

CREATE TRIGGER `ValiditaScheda` BEFORE INSERT ON `SCHEDA` FOR EACH ROW BEGIN
IF (NEW.DimensioneContenitore IS NOT NULL AND NEW.TerrenoAperto = 1) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da inserimento! La pianta o è in terreno aperto
                                o ha contenitore di dimensione definita';
ELSEIF NEW.Tipo = 'Ordine' AND NEW.CodicePianta IN (SELECT CodicePianta

```

FROM

SCHEDA

WHERE Tipo = 'Ordine') THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE_TEXT = 'Errore! Pianta gia` ordinata!';

END IF;

END \$\$

delimiter ;

delimiter \$\$

```

CREATE TRIGGER `InserisciAttributi` BEFORE INSERT ON `SCHEDA` FOR EACH ROW BEGIN
SET NEW.DimensioneContenitore = (SELECT SuperficieOccupata

```

FROM CONTENITORE

WHERE CodiceContenitore = (SELECT

Contenitore

FROM Pianta

WHERE CodicePianta = NEW.CodicePianta)

);

IF NEW.DimensioneContenitore IS NULL THEN

SET NEW.TerrenoAperto = 1;

END IF;

END\$\$

delimiter ;

-- -----

-- Records of `SCHEDA`

```

-----
BEGIN;
INSERT INTO `polliceverde`.`scheda` (`CodiceScheda`, `NomePianta`, `DataAcquisto`,
`Cliente`, `DimensioneAcquisto`, `Tipo`, `Prezzo`, `CodicePianta`) VALUES ('1',
'rosa', '2016-06-22', '10', '6', '1', '20.00', '1');
INSERT INTO `polliceverde`.`scheda` (`CodiceScheda`, `NomePianta`, `DataAcquisto`,
`Cliente`, `DimensioneAcquisto`, `Tipo`, `Prezzo`, `CodicePianta`) VALUES ('2',
'enturium', '2015-04-15', '9', '5', '1', '2.00', '2');
INSERT INTO `polliceverde`.`scheda` (`CodiceScheda`, `NomePianta`, `DataAcquisto`,
`Cliente`, `DimensioneAcquisto`, `Tipo`, `Prezzo`, `CodicePianta`) VALUES ('3',
'viola', '2016-12-05', '8', '5', '2', '24.00', '1');
INSERT INTO `polliceverde`.`scheda` (`CodiceScheda`, `NomePianta`, `DataAcquisto`,
`Cliente`, `DimensioneAcquisto`, `Tipo`, `Prezzo`, `CodicePianta`) VALUES ('4',
'ninfea', '2016-08-05', '7', '10', '2', '30.00', '3');
INSERT INTO `polliceverde`.`scheda` (`CodiceScheda`, `NomePianta`, `DataAcquisto`,
`Cliente`, `DimensioneAcquisto`, `Tipo`, `Prezzo`, `CodicePianta`) VALUES ('5',
'cipresso', '2015-06-23', '6', '5', '3', '25.00', '1');
INSERT INTO `polliceverde`.`scheda` (`CodiceScheda`, `NomePianta`, `DataAcquisto`,
`Cliente`, `DimensioneAcquisto`, `Tipo`, `Prezzo`, `CodicePianta`) VALUES ('6',
'crisantemo', '2016-12-25', '5', '8', '3', '20.50', '4');
INSERT INTO `polliceverde`.`scheda` (`CodiceScheda`, `NomePianta`, `DataAcquisto`,
`Cliente`, `DimensioneAcquisto`, `Tipo`, `Prezzo`, `CodicePianta`) VALUES ('7',
'ginestra', '2016-04-11', '4', '10', '4', '10.00', '1');
INSERT INTO `polliceverde`.`scheda` (`CodiceScheda`, `NomePianta`, `DataAcquisto`,
`Cliente`, `DimensioneAcquisto`, `Tipo`, `Prezzo`, `CodicePianta`) VALUES ('8',
'girasole', '2015-02-01', '3', '8', '4', '15.00', '5');
INSERT INTO `polliceverde`.`scheda` (`CodiceScheda`, `NomePianta`, `DataAcquisto`,
`Cliente`, `DimensioneAcquisto`, `Tipo`, `Prezzo`, `CodicePianta`) VALUES ('9',
'aloee', '2014-07-20', '2', '10', '5', '22.30', '1');
INSERT INTO `polliceverde`.`scheda` (`CodiceScheda`, `NomePianta`, `DataAcquisto`,
`Cliente`, `DimensioneAcquisto`, `Tipo`, `Prezzo`, `CodicePianta`) VALUES ('10',
'gladiolo', '2016-08-21', '1', '5', '5', '21.00', '9');
COMMIT;

```

-- Table structure for `MANUTENZIONEPRIVATA`

```

-----
DROP TABLE IF EXISTS `MANUTENZIONEPRIVATA`;
CREATE TABLE `MANUTENZIONEPRIVATA` (
  `CodiceManutenzionePrivata` INT(11) NOT NULL,
  `DataEsecuzione` DATE NOT NULL,
  `Tipo` ENUM('Rinvaso', 'Piantumazione', 'Trattamento contro patologie',
'Concimazione', 'Potatura') NOT NULL,
  `Scheda` INT(11) NOT NULL,
  PRIMARY KEY (`CodiceManutenzionePrivata`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

-- Records of `MANUTENZIONEPRIVATA`

```

-----
BEGIN;

```

```

INSERT INTO `polliceverde`.`manutenzioneprivata` (`CodiceManutenzionePrivata`,
`DataEsecuzione`, `Tipo`, `Scheda`) VALUES ('1', '2014-06-23', '1', '5');
INSERT INTO `polliceverde`.`manutenzioneprivata` (`CodiceManutenzionePrivata`,
`DataEsecuzione`, `Tipo`, `Scheda`) VALUES ('2', '2015-02-15', '2', '4');
INSERT INTO `polliceverde`.`manutenzioneprivata` (`CodiceManutenzionePrivata`,
`DataEsecuzione`, `Tipo`, `Scheda`) VALUES ('3', '2015-06-21', '3', '6');
INSERT INTO `polliceverde`.`manutenzioneprivata` (`CodiceManutenzionePrivata`,
`DataEsecuzione`, `Tipo`, `Scheda`) VALUES ('4', '2016-02-15', '4', '1');
INSERT INTO `polliceverde`.`manutenzioneprivata` (`CodiceManutenzionePrivata`,
`DataEsecuzione`, `Tipo`, `Scheda`) VALUES ('5', '2016-05-15', '5', '3');
INSERT INTO `polliceverde`.`manutenzioneprivata` (`CodiceManutenzionePrivata`,
`DataEsecuzione`, `Tipo`, `Scheda`) VALUES ('6', '2015-12-08', '5', '8');
INSERT INTO `polliceverde`.`manutenzioneprivata` (`CodiceManutenzionePrivata`,
`DataEsecuzione`, `Tipo`, `Scheda`) VALUES ('7', '2015-05-15', '4', '9');
INSERT INTO `polliceverde`.`manutenzioneprivata` (`CodiceManutenzionePrivata`,
`DataEsecuzione`, `Tipo`, `Scheda`) VALUES ('8', '2015-05-15', '3', '7');
INSERT INTO `polliceverde`.`manutenzioneprivata` (`CodiceManutenzionePrivata`,
`DataEsecuzione`, `Tipo`, `Scheda`) VALUES ('9', '2015-04-15', '1', '10');
INSERT INTO `polliceverde`.`manutenzioneprivata` (`CodiceManutenzionePrivata`,
`DataEsecuzione`, `Tipo`, `Scheda`) VALUES ('10', '2015-05-15', '2', '2');
COMMIT;

```

```

-- -----
-- Table structure for `MANUTENZIONESURICHIESTA`
-- -----

```

```

DROP TABLE IF EXISTS `MANUTENZIONESURICHIESTA`;
CREATE TABLE `MANUTENZIONESURICHIESTA` (
  `CodiceManutenzioneSuRichiesta` INT(11) NOT NULL,
  `DataScadenza` DATE NOT NULL,
  `Tipo` ENUM('Rinvaso', 'Piantumazione', 'Trattamento contro patologie',
'Concimazione', 'Potatura') NOT NULL,
  `Scheda` INT(11) NOT NULL,
  `Stato` ENUM('Eseguita', 'In attesa') NOT NULL,
  `PersonaleRichiesto` INT(2) NOT NULL,
  `Prezzo` FLOAT NOT NULL,
  PRIMARY KEY (`CodiceManutenzioneSuRichiesta`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;

```

```

-- -----
-- Records of `MANUTENZIONESURICHIESTA`
-- -----

```

```

BEGIN;
INSERT INTO `polliceverde`.`manutenzionesurichiasta`
(`CodiceManutenzioneSuRichiesta`, `DataScadenza`, `Tipo`, `Scheda`, `Stato`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('1', '2015-01-15', 'Potatura', '10', 'In
attesa', '6', '50');
INSERT INTO `polliceverde`.`manutenzionesurichiasta`
(`CodiceManutenzioneSuRichiesta`, `DataScadenza`, `Tipo`, `Scheda`, `Stato`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('2', '2015-03-15', 'Concimazione', '9',
'Eseguita', '3', '20');

```

```

INSERT INTO `polliceverde`.`manutenzionesusurichiesta`
(`CodiceManutenzioneSuRichiesta`, `DataScadenza`, `Tipo`, `Scheda`, `Stato`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('3', '2015-05-20', 'Trattamento contro
patologie', '8', 'In attesa', '4', '30');
INSERT INTO `polliceverde`.`manutenzionesusurichiesta`
(`CodiceManutenzioneSuRichiesta`, `DataScadenza`, `Tipo`, `Scheda`, `Stato`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('4', '2015-05-15', 'Piantumazione', '7',
'Eseguita', '3', '20');
INSERT INTO `polliceverde`.`manutenzionesusurichiesta`
(`CodiceManutenzioneSuRichiesta`, `DataScadenza`, `Tipo`, `Scheda`, `Stato`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('5', '2015-05-10', 'Rinvaso', '6', 'In
attesa', '3', '20');
INSERT INTO `polliceverde`.`manutenzionesusurichiesta`
(`CodiceManutenzioneSuRichiesta`, `DataScadenza`, `Tipo`, `Scheda`, `Stato`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('6', '2015-04-15', 'Potatura', '5', 'In
attesa', '3', '25');
INSERT INTO `polliceverde`.`manutenzionesusurichiesta`
(`CodiceManutenzioneSuRichiesta`, `DataScadenza`, `Tipo`, `Scheda`, `Stato`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('7', '2015-12-15', 'Concimazione', '4',
'Eseguita', '2', '15');
INSERT INTO `polliceverde`.`manutenzionesusurichiesta`
(`CodiceManutenzioneSuRichiesta`, `DataScadenza`, `Tipo`, `Scheda`, `Stato`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('8', '2016-05-15', 'Trattamento contro
patologie', '3', 'Eseguita', '4', '35');
INSERT INTO `polliceverde`.`manutenzionesusurichiesta`
(`CodiceManutenzioneSuRichiesta`, `DataScadenza`, `Tipo`, `Scheda`, `Stato`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('9', '2015-05-15', 'Piantumazione', '2', 'In
attesa', '1', '10');
INSERT INTO `polliceverde`.`manutenzionesusurichiesta`
(`CodiceManutenzioneSuRichiesta`, `DataScadenza`, `Tipo`, `Scheda`, `Stato`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('10', '2016-02-13', 'Rinvaso', '1',
'Eseguita', '2', '20');
COMMIT;

```

```

-- -----
-- Table structure for `MANUTENZIONEPROGRAMMATA`
-- -----
DROP TABLE IF EXISTS `MANUTENZIONEPROGRAMMATA`;
CREATE TABLE `MANUTENZIONEPROGRAMMATA` (
  `CodiceManutenzioneProgrammata` INT(11) NOT NULL,
  `Periodo` INT(11) NOT NULL
    REFERENCES `PolliceVerde`.`PERIODO` (`CodicePeriodo`) ON DELETE NO ACTION
ON UPDATE CASCADE,
  `Tipo` ENUM('Rinvaso', 'Piantumazione', 'Trattamento contro patologie',
'Concimazione', 'Potatura') NOT NULL,
  `Scheda` INT(11) NOT NULL,
  `Periodicita` INT(11) NOT NULL, /*mesi*/
  `PersonaleRichiesto` INT(2) NOT NULL,
  `Prezzo` FLOAT NOT NULL,
  PRIMARY KEY (`CodiceManutenzioneProgrammata`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;

```

```

delimiter $$
CREATE TRIGGER `ValiditaPeriodoProgrammata` BEFORE INSERT ON
`MANUTENZIONEPROGRAMMATA` FOR EACH ROW BEGIN
IF (SELECT Tipo
      FROM PERIODO
      WHERE CodicePeriodo = NEW.Periodo) <> 'Manutenzione programmata' THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da vincolo di integrita referenziale con la tabella PERIODO';
END IF;
END $$
delimiter ;

```

```

-- -----
-- Records of `MANUTENZIONEPROGRAMMATA`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`manutenzioneprogrammata`
(`CodiceManutenzioneProgrammata`, `Periodo`, `Tipo`, `Scheda`, `Periodicita`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('1', '5', '4', '2', '1', '1', '15.00');
INSERT INTO `polliceverde`.`manutenzioneprogrammata`
(`CodiceManutenzioneProgrammata`, `Periodo`, `Tipo`, `Scheda`, `Periodicita`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('2', '5', '3', '4', '2', '2', '15.00');
INSERT INTO `polliceverde`.`manutenzioneprogrammata`
(`CodiceManutenzioneProgrammata`, `Periodo`, `Tipo`, `Scheda`, `Periodicita`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('3', '5', '2', '8', '3', '2', '15.00');
INSERT INTO `polliceverde`.`manutenzioneprogrammata`
(`CodiceManutenzioneProgrammata`, `Periodo`, `Tipo`, `Scheda`, `Periodicita`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('4', '5', '5', '6', '2', '1', '10.00');
INSERT INTO `polliceverde`.`manutenzioneprogrammata`
(`CodiceManutenzioneProgrammata`, `Periodo`, `Tipo`, `Scheda`, `Periodicita`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('5', '5', '1', '9', '3', '2', '15.00');
INSERT INTO `polliceverde`.`manutenzioneprogrammata`
(`CodiceManutenzioneProgrammata`, `Periodo`, `Tipo`, `Scheda`, `Periodicita`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('6', '5', '2', '7', '4', '3', '25.00');
INSERT INTO `polliceverde`.`manutenzioneprogrammata`
(`CodiceManutenzioneProgrammata`, `Periodo`, `Tipo`, `Scheda`, `Periodicita`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('8', '5', '1', '5', '2', '3', '20.00');
INSERT INTO `polliceverde`.`manutenzioneprogrammata`
(`CodiceManutenzioneProgrammata`, `Periodo`, `Tipo`, `Scheda`, `Periodicita`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('9', '5', '3', '3', '1', '3', '30.00');
INSERT INTO `polliceverde`.`manutenzioneprogrammata`
(`CodiceManutenzioneProgrammata`, `Periodo`, `Tipo`, `Scheda`, `Periodicita`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('10', '5', '4', '10', '1', '4', '35.00');
INSERT INTO `polliceverde`.`manutenzioneprogrammata`
(`CodiceManutenzioneProgrammata`, `Periodo`, `Tipo`, `Scheda`, `Periodicita`,
`PersonaleRichiesto`, `Prezzo`) VALUES ('7', '5', '5', '1', '2', '3', '20.00');
COMMIT;

```

```

-- -----
-- Table structure for `CLIENTE`

```

```

-----
DROP TABLE IF EXISTS `CLIENT`;
CREATE TABLE `CLIENT` (
  `CodiceCliente` INT(11) NOT NULL,
  `Nickname` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `Password` VARCHAR(45) NOT NULL,
  `DomandaDiSicurezza` VARCHAR(45) NOT NULL,
  `RispostaDomandaSicurezza` VARCHAR(45) NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Cognome` VARCHAR(45) NOT NULL,
  `Residenza` VARCHAR(45) NOT NULL,
  `NumeroValutazioni` INT(3) NOT NULL DEFAULT 0,
  PRIMARY KEY (`CodiceCliente`, `Nickname`, `Email`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;

-- -----
--   Records of `CLIENT`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`client` (`CodiceCliente`, `Nickname`, `Email`,
`Password`, `DomandaDiSicurezza`, `RispostaDomandaSicurezza`, `Nome`, `Cognome`,
`Residenza`, `NumeroValutazioni`) VALUES
('1', 'ddc95', 'yesiam01@gmail.com', 'uffa', 'hai voluto fare lo scemo', 'e mo
t\`attacchi', 'diego', 'del castello', 'roma', '12'),
('2', 'tdl96', 'yesiam02@gmail.com', 'non', 'perché', 'boh', 'teresa', 'di lascio',
'pisa', '25'),
('3', 'banana33', 'yesiam03@gmail.com', 'riesco', 'quante', 'tante', 'pistolesi',
'annonna', 'livorno', '75'),
('4', 'padoin23', 'yesiam04@gmail.com', 'a', 'tutte', 'niente', 'radja',
'nainggolan', 'bruxelles', '84'),
('5', 'fragola86', 'yesiam05@gmail.com', 'decidere', 'ricordarme', 'no', 'walter',
'mazzarri', 'firenze', '55'),
('6', 'kitten90', 'yesiam06@gmail.com', 'la', 'figuriamoci', 'ma infatti', 'woicech',
'szczesny', 'varsavia', '63'),
('7', 'muort60', 'yesiam07@gmail.com', 'password', 'poi', 'sarà come morire', 'edin',
'dzeko', 'sarajevo', '74'),
('8', 'hola69', 'yesiam08@gmail.com', 'mia', 'altro', 'che facile', 'kevin',
'strootman', 'amsterdam', '25'),
('9', 'lettera6', 'yesiam09@gmail.com', 'figuriamoci', 'qualcun', 'dovrà pur
esserci', 'kostas', 'manolas', 'atene', '34'),
('10', '45questoprimailnumero', 'yesiam10@gmail.com', 'quella', 'di ', 'là',
'mohamed', 'salah', 'il cairo', '66');
COMMIT;

-- -----
--   Table structure for `POST`
-- -----
DROP TABLE IF EXISTS `POST`;
CREATE TABLE `POST` (
  `CodicePost` INT(11) NOT NULL,

```



```

`Timestamp` TIMESTAMP NOT NULL,
`Nickname` VARCHAR(45) NOT NULL
    REFERENCES `PolliceVerde`.`CLIENT` (`Nickname`) ON DELETE NO ACTION ON
UPDATE CASCADE,
`Thread` INT(11) NOT NULL,
`Testo` TEXT(255) NOT NULL,
`Link` LONGTEXT NULL,
    PRIMARY KEY (`CodicePost`, `Timestamp`, `Thread`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;

-- -----
--   Records of `POST`
-- -----

BEGIN;
INSERT INTO `polliceverde`.`post` (`CodicePost`, `Timestamp`, `Nickname`, `Thread`,
`Testo`, `Link`) VALUES
('1', '2017-01-01 00:00:01', 'banana33', '7895', 'buon anno a tutti',
'http://www.pollicegreen.com/6-fiori-portafortuna-capodanno-foto/32852/'),
('3', '2016-08-08 15:01:17', 'ddc95', '84', 'non c'è cactus senza spine',
'https://www.giardinaggio.it/grasse/piante-grasse.asp'),
('5', '2016-04-25 17:10:26', 'banana33', '321', 'maremma tra tutti sti farmaci non ci
si capisce più nulla torno ai rimedi della nonna',
'https://www.ideegreen.it/piante-medicinali-elenco-schede-71252.html'),
('8', '2016-07-25 12:54:01', 'tdl96', '515151', 'ecco cosa ci vuole',
'http://www.ecoage.it/deforestazione.htm'),
('9', '2016-06-23 13:17:49', 'ddc95', '3213165', 'per il mio compleanno guardate cosa
mi hanno regalato!',
'https://s-media-cache-ak0.pinimg.com/736x/0f/7a/cd/0f7acdc9feba374482b5faf75e5e82
b2.jpg');
INSERT INTO `polliceverde`.`post` (`CodicePost`, `Timestamp`, `Nickname`, `Thread`,
`Testo`) VALUES
('2', '2017-01-02 11:43:01', 'padoin23', '351', 'gente che per prima cosa nell'anno
nuovo fa gli auguri alle piante'),
('4', '2015-06-30 17:48:54', 'tdl96', '49841', 'un giorno da giardiniere è un giorno
bellissimo'),
('6', '2016-02-17 22:26:32', 'padoin23', '8716', 'juventino infame per te solo lame'),
('7', '2015-01-09 08:32:10', 'tdl96', '6516512', 'un giorno a implementare databases
è un giorno snervante'),
('10', '2015-04-15 23:08:01', 'fragola86', '846513', 'avresti fatto 10 anni...');
COMMIT;

-- -----
--   Table structure for `COMMENTODIRISPOSTA`
-- -----

DROP TABLE IF EXISTS `COMMENTODIRISPOSTA`;
CREATE TABLE `COMMENTODIRISPOSTA` (
  `CodiceCommentoDiRisposta` INT(11) NOT NULL,
  `Timestamp` TIMESTAMP NOT NULL,
  `Post` INT(11) NOT NULL,
    FOREIGN KEY (`Post`) REFERENCES `PolliceVerde`.`POST` (`CodicePost`) ON
DELETE NO ACTION ON UPDATE CASCADE,

```

```
`Nickname` VARCHAR(45) NOT NULL
REFERENCES `PolliceVerde`.`CLIENT` (`Nickname`) ON DELETE NO ACTION ON UPDATE
CASCADE,
`Testo` TEXT(255) NOT NULL,
PRIMARY KEY (`CodiceCommentoDiRisposta`, `Timestamp`, `Post`)) ENGINE=InnoDB
DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `COMMENTODIRISPOSTA`
-- -----
```

```
BEGIN;
INSERT INTO `polliceverde`.`commentodirisposta` (`CodiceCommentoDiRisposta`,
`Timestamp`, `Post`, `Nickname`, `Testo`) VALUES ('1', '2017-01-02 11:53:01', '2',
'tdl96', 'ahahahaha muoio'), ('2', '2017-01-02 12:43:01', '2', 'ddc95', 'gente
strana'), ('3', '2016-06-23 13:49:49', '9', 'banana33', 'chiamala rritaa'), ('4',
'2016-07-25 13:54:01', '8', 'padoin23', 'sarebbe ora!!'), ('5', '2015-01-09 08:34:10',
'7', 'banana33', '30 e lode va bene o preferisce ridarlo?'), ('6', '2016-02-17
22:32:26', '6', 'ddc95', 'fagliene un paio che ti ho pure sul fanta godo doppio!!'),
('7', '2016-04-25 17:26:10', '5', 'fragola86', 'sono sempre i migliori'), ('8',
'2015-06-30 17:54:48', '4', 'ddc95', 'e tanto quello andremo a fare!!'), ('9',
'2016-08-08 15:17:01', '3', 'fragola86', 'un poeta'), ('10', '2017-01-01 00:01:00',
'1', 'fragola86', 'grazie');
COMMIT;
```

```
-- -----
-- Table structure for `VALUTAZIONE`
-- -----
```

```
DROP TABLE IF EXISTS `VALUTAZIONE`;
CREATE TABLE `VALUTAZIONE` (
`CodiceValutazione` INT(11) NOT NULL,
`CommentoDiRisposta` INT(11) NOT NULL
REFERENCES `PolliceVerde`.`COMMENTODIRISPOSTA` (`CodiceCommentoDiRisposta`)
ON DELETE NO ACTION ON UPDATE CASCADE,
`Punteggio` INT(1) NOT NULL,
`Nickname` VARCHAR(45) NOT NULL
REFERENCES `PolliceVerde`.`CLIENT` (`Nickname`) ON DELETE NO ACTION ON UPDATE
CASCADE,
PRIMARY KEY (`Codicevalutazione`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
delimiter $$
CREATE TRIGGER `PrimaRidondanzaInsert` AFTER INSERT ON `VALUTAZIONE` FOR EACH ROW BEGIN
SET @ClientInteressato = (SELECT Nickname
FROM COMMENTODIRISPOSTA
WHERE NEW.CommentoDiRisposta = CodiceCommentoDiRisposta
);
UPDATE `CLIENT`
SET `NumeroValutazioni` = `NumeroValutazioni` + 1
WHERE `Nickname` = @ClientInteressato;
END $$
delimiter ;
```

```
delimiter $$
CREATE TRIGGER `PrimaRidondanzaDelete` AFTER DELETE ON `VALUTAZIONE` FOR EACH ROW BEGIN
SET @ClientInteressato = (SELECT Nickname
                           FROM COMMENTODIRISPOSTA
                           WHERE OLD.CommentoDiRisposta = CodiceCommentoDiRisposta
                           );
UPDATE `CLIENT`
SET `NumeroValutazioni` = `NumeroValutazioni` - 1
WHERE `Nickname` = @ClientInteressato;
END $$
delimiter ;
```

```
delimiter $$
CREATE TRIGGER `VerificaValutazione` BEFORE INSERT ON `VALUTAZIONE` FOR EACH ROW BEGIN
IF NEW.Punteggio > 5 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Valutazione fuori parametro!!';
END IF;
END $$
delimiter ;
```

```
-- -----
-- Records of `VALUTAZIONE`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`valutazione` (`CodiceValutazione`, `CommentoDiRisposta`,
`Punteggio`, `Nickname`) VALUES ('1', '10', '1', 'ddc95'), ('2', '9', '2',
'fragola86'), ('3', '8', '3', 'padoin23'), ('4', '7', '4', 'banana33'), ('5', '6', '5',
'tdl96'), ('6', '5', '1', 'fragola86'), ('7', '4', '2', 'ddc95'), ('8', '3', '3',
'tdl96'), ('9', '2', '4', 'padoin23'), ('10', '1', '5', 'banana33');
COMMIT;
```

```
-- -----
-- Table structure for `GIARDINO`
-- -----
DROP TABLE IF EXISTS `GIARDINO`;
CREATE TABLE `GIARDINO` (
  `CodiceGiardino` INT(11) NOT NULL,
  `Cliente` INT(11) NOT NULL,
  FOREIGN KEY (`Cliente`) REFERENCES `PolliceVerde`.`CLIENT`
(`CodiceCliente`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `NumeroOggetti` INT(3) NOT NULL DEFAULT 0,
  `NumeroSettori` INT(2) NOT NULL DEFAULT 1,
  `Lunghezza` INT(2) NOT NULL,
  `Larghezza` INT(2) NOT NULL,
  PRIMARY KEY (`CodiceGiardino`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `GIARDINO`
-- -----
BEGIN;
```

```

INSERT INTO `polliceverde`.`giardino` (`CodiceGiardino`, `Cliente`, `NumeroOggetti`,
`NumeroSettori`, `Lunghezza`, `Larghezza`) VALUES ('1', '10', '10', '7', '8', '5');
INSERT INTO `polliceverde`.`giardino` (`CodiceGiardino`, `Cliente`, `NumeroOggetti`,
`NumeroSettori`, `Lunghezza`, `Larghezza`) VALUES ('2', '9', '12', '5', '15', '25');
INSERT INTO `polliceverde`.`giardino` (`CodiceGiardino`, `Cliente`, `NumeroOggetti`,
`NumeroSettori`, `Lunghezza`, `Larghezza`) VALUES ('3', '8', '6', '3', '20', '30');
INSERT INTO `polliceverde`.`giardino` (`CodiceGiardino`, `Cliente`, `NumeroOggetti`,
`NumeroSettori`, `Lunghezza`, `Larghezza`) VALUES ('4', '7', '8', '5', '20', '15');
INSERT INTO `polliceverde`.`giardino` (`CodiceGiardino`, `Cliente`, `NumeroOggetti`,
`NumeroSettori`, `Lunghezza`, `Larghezza`) VALUES ('5', '6', '16', '4', '15', '20');
INSERT INTO `polliceverde`.`giardino` (`CodiceGiardino`, `Cliente`, `NumeroOggetti`,
`NumeroSettori`, `Lunghezza`, `Larghezza`) VALUES ('6', '5', '4', '5', '30', '20');
INSERT INTO `polliceverde`.`giardino` (`CodiceGiardino`, `Cliente`, `NumeroOggetti`,
`NumeroSettori`, `Lunghezza`, `Larghezza`) VALUES ('7', '4', '7', '1', '25', '15');
INSERT INTO `polliceverde`.`giardino` (`CodiceGiardino`, `Cliente`, `NumeroOggetti`,
`NumeroSettori`, `Lunghezza`, `Larghezza`) VALUES ('8', '3', '9', '3', '25', '30');
INSERT INTO `polliceverde`.`giardino` (`CodiceGiardino`, `Cliente`, `NumeroOggetti`,
`NumeroSettori`, `Lunghezza`, `Larghezza`) VALUES ('9', '2', '11', '4', '15', '20');
INSERT INTO `polliceverde`.`giardino` (`CodiceGiardino`, `Cliente`, `NumeroOggetti`,
`NumeroSettori`, `Lunghezza`, `Larghezza`) VALUES ('10', '1', '10', '3', '20', '15');
COMMIT;

```

```

-- -----
-- Table structure for `SETTORE`
-- -----
DROP TABLE IF EXISTS `SETTORE`;
CREATE TABLE `SETTORE` (
  `CodiceSettore` INT(11) NOT NULL,
  `Giardino` INT(11) NOT NULL,
  FOREIGN KEY (`Giardino`) REFERENCES `PolliceVerde`.`GIARDINO`
(`CodiceGiardino`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Ascissa` INT(11) NOT NULL,
  `Ordinata` INT(11) NOT NULL,
  `Lunghezza` INT(2) NOT NULL,
  `Larghezza` INT(2) NOT NULL,
  PRIMARY KEY (`CodiceSettore`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

delimiter $$
CREATE TRIGGER `ValiditaSettore` BEFORE INSERT ON `SETTORE` FOR EACH ROW BEGIN
SET @lunghezza = (SELECT Lunghezza
                  FROM GIARDINO
                  WHERE CodiceGiardino = NEW.Giardino);
SET @larghezza = (SELECT Larghezza
                  FROM GIARDINO
                  WHERE CodiceGiardino = NEW.Giardino);
IF !(NEW.Ascissa BETWEEN 0 AND @larghezza*100
    AND NEW.Ordinata BETWEEN 0 AND @lunghezza*100
    AND NEW.Ascissa + NEW.Larghezza <= @larghezza*100
    AND NEW.Ordinata + NEW.Lunghezza <= @lunghezza*100) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore! Il settore fuoriesce dal giardino';

```

```
END IF;  
END $$  
delimiter ;
```

```
delimiter $$  
CREATE TRIGGER `IntersezioneSettore` BEFORE INSERT ON `SETTORE` FOR EACH ROW BEGIN  
  
DECLARE ascissa INT(11);  
DECLARE fineascissa INT(11);  
DECLARE ordinata INT(11);  
DECLARE fineordinata INT(11);  
  
DECLARE cursoreascisse CURSOR FOR  
SELECT Ascissa  
FROM SETTORE  
WHERE Giardino = NEW.Giardino;  
  
DECLARE cursorefineascisse CURSOR FOR  
SELECT (Ascissa + Larghezza)  
FROM SETTORE  
WHERE Giardino = NEW.Giardino;  
  
DECLARE cursoreordinate CURSOR FOR  
SELECT Ordinata  
FROM SETTORE  
WHERE Giardino = NEW.Giardino;  
  
DECLARE cursorefineordinate CURSOR FOR  
SELECT (Ordinata + Lunghezza)  
FROM SETTORE  
WHERE Giardino = NEW.Giardino;  
  
DECLARE CONTINUE HANDLER  
FOR NOT FOUND SET @finito = 1;  
  
OPEN cursoreascisse;  
OPEN cursorefineascisse;  
OPEN cursoreordinate;  
OPEN cursorefineordinate;  
  
verifica: LOOP  
FETCH cursoreascisse INTO ascissa;  
FETCH cursorefineascisse INTO fineascissa;  
FETCH cursoreordinate INTO ordinata;  
FETCH cursorefineordinate INTO fineordinata;  
  
IF @finito = 1 THEN  
LEAVE verifica;  
END IF;  
IF (NEW.Ascissa BETWEEN ascissa AND fineascissa
```

```

        AND NEW.Ordinata BETWEEN ordinata AND fineordinata) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da intersezione';
END IF;
END LOOP verifica;

```

```

CLOSE cursoreascisse;
CLOSE cursorefineascisse;

```

```

END $$
delimiter ;

```

```

-- -----
-- Records of `SETTORE`
-- -----

```

```

BEGIN;
INSERT INTO `polliceverde`.`settores` (`CodiceSettores`, `Giardino`, `Ascissa`,
`Ordinata`, `Lunghezza`, `Larghezza`) VALUES ('1', '1', '0', '0', '600', '200');
INSERT INTO `polliceverde`.`settores` (`CodiceSettores`, `Giardino`, `Ascissa`,
`Ordinata`, `Lunghezza`, `Larghezza`) VALUES ('2', '1', '0', '0', '600', '200', '500');
INSERT INTO `polliceverde`.`settores` (`CodiceSettores`, `Giardino`, `Ascissa`,
`Ordinata`, `Lunghezza`, `Larghezza`) VALUES ('3', '1', '200', '0', '100', '300');
INSERT INTO `polliceverde`.`settores` (`CodiceSettores`, `Giardino`, `Ascissa`,
`Ordinata`, `Lunghezza`, `Larghezza`) VALUES ('4', '10', '0', '0', '1000', '1000');
INSERT INTO `polliceverde`.`settores` (`CodiceSettores`, `Giardino`, `Ascissa`,
`Ordinata`, `Lunghezza`, `Larghezza`) VALUES ('5', '9', '0', '0', '1000', '1000');
INSERT INTO `polliceverde`.`settores` (`CodiceSettores`, `Giardino`, `Ascissa`,
`Ordinata`, `Lunghezza`, `Larghezza`) VALUES ('6', '8', '0', '0', '1000', '2000');
INSERT INTO `polliceverde`.`settores` (`CodiceSettores`, `Giardino`, `Ascissa`,
`Ordinata`, `Lunghezza`, `Larghezza`) VALUES ('7', '7', '0', '0', '2000', '1000');
INSERT INTO `polliceverde`.`settores` (`CodiceSettores`, `Giardino`, `Ascissa`,
`Ordinata`, `Lunghezza`, `Larghezza`) VALUES ('8', '6', '0', '0', '1500', '1500');
INSERT INTO `polliceverde`.`settores` (`CodiceSettores`, `Giardino`, `Ascissa`,
`Ordinata`, `Lunghezza`, `Larghezza`) VALUES ('9', '5', '0', '0', '1000', '1000');
INSERT INTO `polliceverde`.`settores` (`CodiceSettores`, `Giardino`, `Ascissa`,
`Ordinata`, `Lunghezza`, `Larghezza`) VALUES ('10', '4', '0', '0', '1500', '1000');
COMMIT;

```

```

-- -----
-- Table structure for `VASO`
-- -----

```

```

DROP TABLE IF EXISTS `VASO`;
CREATE TABLE `VASO` (
  `CodiceVaso` INT(11) NOT NULL,
  `Settores` INT(11) NOT NULL,
  FOREIGN KEY (`Settores`) REFERENCES `PolliceVerde`.`SETTORES`
(`CodiceSettores`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `Ascissa` INT(11) NOT NULL,
  `Ordinata` INT(11) NOT NULL,
  `Lato` INT(2) NOT NULL,
  PRIMARY KEY (`CodiceVaso`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

delimiter $$
CREATE TRIGGER `ValiditaVaso` BEFORE INSERT ON `VASO` FOR EACH ROW BEGIN
SET @iniziosettoreX = (SELECT Ascissa
                        FROM SETTORE
                        WHERE CodiceSettore = NEW.Settore);
SET @finesettoreX = (SELECT Ascissa + Larghezza
                     FROM SETTORE
                     WHERE CodiceSettore = NEW.Settore);
SET @iniziosettoreY = (SELECT Ordinata
                       FROM SETTORE
                       WHERE CodiceSettore = NEW.Settore);
SET @finesettoreY = (SELECT Ordinata + Lunghezza
                     FROM SETTORE
                     WHERE CodiceSettore = NEW.Settore);
IF !(NEW.Ascissa BETWEEN @iniziosettoreX AND @finesettoreX
    AND NEW.Ordinata BETWEEN @iniziosettoreY AND @finesettoreY
    AND NEW.Ascissa + NEW.Lato BETWEEN @iniziosettoreX AND @finesettoreX
    AND NEW.Ordinata + NEW.Lato BETWEEN @iniziosettoreY AND @finesettoreY) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore! Il vaso fuoriesce dal settore';
END IF;
END $$
delimiter ;

```

```

delimiter $$
CREATE TRIGGER `IntersezioneVaso` BEFORE INSERT ON `VASO` FOR EACH ROW BEGIN

DECLARE ascissa INT(11);
DECLARE fineascissa INT(11);
DECLARE ordinata INT(11);
DECLARE fineordinata INT(11);

DECLARE cursoreascisse CURSOR FOR
SELECT Ascissa
FROM VASO
WHERE Settore = NEW.Settore;

DECLARE cursorefineascisse CURSOR FOR
SELECT (Ascissa + Lato) AS Fine
FROM VASO
WHERE Settore = NEW.Settore;

DECLARE cursoreordinate CURSOR FOR
SELECT Ordinata
FROM VASO
WHERE Settore = NEW.Settore;

DECLARE cursorefineordinate CURSOR FOR
SELECT (Ordinata + Lato)
FROM VASO

```

```
WHERE Settore = NEW.Settore;
```

```
DECLARE CONTINUE HANDLER
FOR NOT FOUND SET @finito = 1;
```

```
OPEN cursoreascisse;
OPEN cursorefineascisse;
OPEN cursoreordinate;
OPEN cursorefineordinate;
```

```
verifica: LOOP
FETCH cursoreascisse INTO ascissa;
FETCH cursorefineascisse INTO fineascissa;
FETCH cursoreordinate INTO ordinata;
FETCH cursorefineordinate INTO fineordinata;
```

```
IF @finito = 1 THEN
LEAVE verifica;
END IF;
IF (NEW.Ascissa BETWEEN ascissa AND fineascissa
AND NEW.Ordinata BETWEEN ordinata AND fineordinata) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da intersezione';
END IF;
END LOOP verifica;
```

```
CLOSE cursoreascisse;
CLOSE cursorefineascisse;
```

```
END $$
delimiter ;
```

```
-- -----
-- Records of `VASO`
-- -----
```

```
BEGIN;
INSERT INTO `polliceverde`.`vaso` (`CodiceVaso`, `Settore`, `Ascissa`, `Ordinata`,
`Lato`) VALUES ('1', '9', '0', '0', '200');
INSERT INTO `polliceverde`.`vaso` (`CodiceVaso`, `Settore`, `Ascissa`, `Ordinata`,
`Lato`) VALUES ('2', '8', '0', '0', '500');
INSERT INTO `polliceverde`.`vaso` (`CodiceVaso`, `Settore`, `Ascissa`, `Ordinata`,
`Lato`) VALUES ('3', '7', '0', '0', '700');
INSERT INTO `polliceverde`.`vaso` (`CodiceVaso`, `Settore`, `Ascissa`, `Ordinata`,
`Lato`) VALUES ('4', '1', '0', '0', '200');
INSERT INTO `polliceverde`.`vaso` (`CodiceVaso`, `Settore`, `Ascissa`, `Ordinata`,
`Lato`) VALUES ('5', '1', '0', '200', '200');
INSERT INTO `polliceverde`.`vaso` (`CodiceVaso`, `Settore`, `Ascissa`, `Ordinata`,
`Lato`) VALUES ('6', '5', '0', '0', '500');
INSERT INTO `polliceverde`.`vaso` (`CodiceVaso`, `Settore`, `Ascissa`, `Ordinata`,
`Lato`) VALUES ('7', '4', '0', '0', '400');
```



```
INSERT INTO `polliceverde`.`vaso` (`CodiceVaso`, `Settore`, `Ascissa`, `Ordinata`,
`Lato`) VALUES ('8', '3', '200', '0', '80');
INSERT INTO `polliceverde`.`vaso` (`CodiceVaso`, `Settore`, `Ascissa`, `Ordinata`,
`Lato`) VALUES ('9', '2', '0', '600', '150');
INSERT INTO `polliceverde`.`vaso` (`CodiceVaso`, `Settore`, `Ascissa`, `Ordinata`,
`Lato`) VALUES ('10', '1', '0', '400', '200');
COMMIT;
```

```
-- -----
-- Table structure for `ESECUZIONE_CONCIMAZIONE`
-- -----
DROP TABLE IF EXISTS `ESECUZIONE_CONCIMAZIONE`;
CREATE TABLE `ESECUZIONE_CONCIMAZIONE` (
  `CodiceConcimazione` INT(11) NOT NULL,
    FOREIGN KEY (`CodiceConcimazione`) REFERENCES
`PolliceVerde`.`CONCIMAZIONE` (`CodiceConcimazione`) ON DELETE NO ACTION ON UPDATE
CASCADE,
  `CodiceIntervento` INT(11) NOT NULL,
    FOREIGN KEY (`CodiceIntervento`) REFERENCES `PolliceVerde`.`INTERVENTO`
(`CodiceIntervento`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodiceIntervento`, `CodiceConcimazione`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;
```

```
-- -----
-- Records of `ESECUZIONE_CONCIMAZIONE`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`esecuzione_concimazione` (`CodiceConcimazione`,
`CodiceIntervento`) VALUES ('1', '1');
INSERT INTO `polliceverde`.`esecuzione_concimazione` (`CodiceConcimazione`,
`CodiceIntervento`) VALUES ('2', '2');
INSERT INTO `polliceverde`.`esecuzione_concimazione` (`CodiceConcimazione`,
`CodiceIntervento`) VALUES ('3', '3');
INSERT INTO `polliceverde`.`esecuzione_concimazione` (`CodiceConcimazione`,
`CodiceIntervento`) VALUES ('4', '4');
INSERT INTO `polliceverde`.`esecuzione_concimazione` (`CodiceConcimazione`,
`CodiceIntervento`) VALUES ('5', '5');
INSERT INTO `polliceverde`.`esecuzione_concimazione` (`CodiceConcimazione`,
`CodiceIntervento`) VALUES ('6', '6');
INSERT INTO `polliceverde`.`esecuzione_concimazione` (`CodiceConcimazione`,
`CodiceIntervento`) VALUES ('7', '7');
INSERT INTO `polliceverde`.`esecuzione_concimazione` (`CodiceConcimazione`,
`CodiceIntervento`) VALUES ('8', '8');
INSERT INTO `polliceverde`.`esecuzione_concimazione` (`CodiceConcimazione`,
`CodiceIntervento`) VALUES ('9', '9');
INSERT INTO `polliceverde`.`esecuzione_concimazione` (`CodiceConcimazione`,
`CodiceIntervento`) VALUES ('10', '10');
COMMIT;
```

```
-- -----
```

```
-- Table structure for `PRESENZA`
-- -----
DROP TABLE IF EXISTS `PRESENZA`;
CREATE TABLE `PRESENZA` (
  `CodiceElementoDisciolto` INT(11) NOT NULL,
    FOREIGN KEY (`CodiceElementoDisciolto`) REFERENCES
`PolliceVerde`.`ELEMENTODISCIOLTO` (`CodiceElementoDisciolto`) ON DELETE NO ACTION ON
UPDATE CASCADE,
  `CodiceTerreno` INT(11) NOT NULL,
    FOREIGN KEY (`CodiceTerreno`) REFERENCES `PolliceVerde`.`TERRENO`
(`CodiceTerreno`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodiceElementoDisciolto`, `CodiceTerreno`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;

-- -----
-- Records of `PRESENZA`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`presenza` (`CodiceElementoDisciolto`, `CodiceTerreno`)
VALUES ('1', '1');
INSERT INTO `polliceverde`.`presenza` (`CodiceElementoDisciolto`, `CodiceTerreno`)
VALUES ('2', '2');
INSERT INTO `polliceverde`.`presenza` (`CodiceElementoDisciolto`, `CodiceTerreno`)
VALUES ('3', '3');
INSERT INTO `polliceverde`.`presenza` (`CodiceElementoDisciolto`, `CodiceTerreno`)
VALUES ('4', '4');
INSERT INTO `polliceverde`.`presenza` (`CodiceElementoDisciolto`, `CodiceTerreno`)
VALUES ('5', '5');
INSERT INTO `polliceverde`.`presenza` (`CodiceElementoDisciolto`, `CodiceTerreno`)
VALUES ('6', '6');
INSERT INTO `polliceverde`.`presenza` (`CodiceElementoDisciolto`, `CodiceTerreno`)
VALUES ('7', '7');
INSERT INTO `polliceverde`.`presenza` (`CodiceElementoDisciolto`, `CodiceTerreno`)
VALUES ('8', '8');
INSERT INTO `polliceverde`.`presenza` (`CodiceElementoDisciolto`, `CodiceTerreno`)
VALUES ('9', '9');
INSERT INTO `polliceverde`.`presenza` (`CodiceElementoDisciolto`, `CodiceTerreno`)
VALUES ('10', '10');
COMMIT;

-- -----
-- Table structure for `ESORDIO`
-- -----
DROP TABLE IF EXISTS `ESORDIO`;
CREATE TABLE `ESORDIO` (
  `CodiceSintomatologia` INT(11) NOT NULL,
    FOREIGN KEY (`CodiceSintomatologia`) REFERENCES
`PolliceVerde`.`SINTOMATOLOGIA` (`CodiceSintomatologia`) ON DELETE NO ACTION ON
UPDATE CASCADE,
  `CodicePatologia` INT(11) NOT NULL,
```

```

        FOREIGN KEY (`CodicePatologia`) REFERENCES `PolliceVerde`.`PATOLOGIA`
(`CodicePatologia`) ON DELETE NO ACTION ON UPDATE CASCADE,
        PRIMARY KEY (`CodiceSintomatologia`, `CodicePatologia`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;

```

```

-- -----
--   Records of `ESORDIO`
-- -----

```

```

BEGIN;
INSERT INTO `polliceverde`.`esordio` (`CodiceSintomatologia`, `CodicePatologia`)
VALUES ('1', '1');
INSERT INTO `polliceverde`.`esordio` (`CodiceSintomatologia`, `CodicePatologia`)
VALUES ('2', '2');
INSERT INTO `polliceverde`.`esordio` (`CodiceSintomatologia`, `CodicePatologia`)
VALUES ('3', '3');
INSERT INTO `polliceverde`.`esordio` (`CodiceSintomatologia`, `CodicePatologia`)
VALUES ('4', '4');
INSERT INTO `polliceverde`.`esordio` (`CodiceSintomatologia`, `CodicePatologia`)
VALUES ('5', '5');
INSERT INTO `polliceverde`.`esordio` (`CodiceSintomatologia`, `CodicePatologia`)
VALUES ('6', '6');
INSERT INTO `polliceverde`.`esordio` (`CodiceSintomatologia`, `CodicePatologia`)
VALUES ('7', '7');
INSERT INTO `polliceverde`.`esordio` (`CodiceSintomatologia`, `CodicePatologia`)
VALUES ('8', '8');
INSERT INTO `polliceverde`.`esordio` (`CodiceSintomatologia`, `CodicePatologia`)
VALUES ('9', '9');
INSERT INTO `polliceverde`.`esordio` (`CodiceSintomatologia`, `CodicePatologia`)
VALUES ('10', '10');
COMMIT;

```

```

-- -----
--   Table structure for `ESIGENZA`
-- -----

```

```

DROP TABLE IF EXISTS `ESIGENZA`;
CREATE TABLE `ESIGENZA` (
  `CodiceFabbisogno` INT(11) NOT NULL,
    FOREIGN KEY (`CodiceFabbisogno`) REFERENCES `PolliceVerde`.`FABBISOGNO`
(`CodiceFabbisogno`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `CodicePianta` INT(11) NOT NULL,
    FOREIGN KEY (`CodicePianta`) REFERENCES `PolliceVerde`.`PIANTA`
(`CodicePianta`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodicePianta`, `CodiceFabbisogno`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;

```

```

-- -----
--   Records of `ESIGENZA`
-- -----

```

```

BEGIN;
INSERT INTO `polliceverde`.`esigenza` (`CodiceFabbisogno`, `CodicePianta`) VALUES
('1', '1');

```

```

INSERT INTO `polliceverde`.`esigenza` (`CodiceFabbisogno`, `CodicePianta`) VALUES
('2', '2');
INSERT INTO `polliceverde`.`esigenza` (`CodiceFabbisogno`, `CodicePianta`) VALUES
('3', '3');
INSERT INTO `polliceverde`.`esigenza` (`CodiceFabbisogno`, `CodicePianta`) VALUES
('4', '4');
INSERT INTO `polliceverde`.`esigenza` (`CodiceFabbisogno`, `CodicePianta`) VALUES
('6', '5');
INSERT INTO `polliceverde`.`esigenza` (`CodiceFabbisogno`, `CodicePianta`) VALUES
('5', '6');
INSERT INTO `polliceverde`.`esigenza` (`CodiceFabbisogno`, `CodicePianta`) VALUES
('7', '7');
INSERT INTO `polliceverde`.`esigenza` (`CodiceFabbisogno`, `CodicePianta`) VALUES
('8', '8');
INSERT INTO `polliceverde`.`esigenza` (`CodiceFabbisogno`, `CodicePianta`) VALUES
('9', '9');
INSERT INTO `polliceverde`.`esigenza` (`CodiceFabbisogno`, `CodicePianta`) VALUES
('10', '10');
COMMIT;

```

```

-- -----
-- Table structure for `ESECUZIONE_RINVASO`
-- -----
DROP TABLE IF EXISTS `ESECUZIONE_RINVASO`;
CREATE TABLE `ESECUZIONE_RINVASO` (
  `CodiceRinvaso` INT(11) NOT NULL,
    FOREIGN KEY (`CodiceRinvaso`) REFERENCES `PolliceVerde`.`RINVASO`
(`CodiceRinvaso`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `CodiceIntervento` INT(11) NOT NULL,
    FOREIGN KEY (`CodiceIntervento`) REFERENCES `PolliceVerde`.`INTERVENTO`
(`CodiceIntervento`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodiceIntervento`, `CodiceRinvaso`)) ENGINE=InnoDB DEFAULT
CHARSET=latin1;

```

```

-- -----
-- Records of `ESECUZIONE_RINVASO`
-- -----
INSERT INTO `polliceverde`.`esecuzione_rinvaso` (`CodiceRinvaso`,
`CodiceIntervento`) VALUES ('1', '1');
INSERT INTO `polliceverde`.`esecuzione_rinvaso` (`CodiceRinvaso`,
`CodiceIntervento`) VALUES ('2', '2');
INSERT INTO `polliceverde`.`esecuzione_rinvaso` (`CodiceRinvaso`,
`CodiceIntervento`) VALUES ('3', '3');
INSERT INTO `polliceverde`.`esecuzione_rinvaso` (`CodiceRinvaso`,
`CodiceIntervento`) VALUES ('4', '4');
INSERT INTO `polliceverde`.`esecuzione_rinvaso` (`CodiceRinvaso`,
`CodiceIntervento`) VALUES ('5', '5');
INSERT INTO `polliceverde`.`esecuzione_rinvaso` (`CodiceRinvaso`,
`CodiceIntervento`) VALUES ('6', '6');
INSERT INTO `polliceverde`.`esecuzione_rinvaso` (`CodiceRinvaso`,
`CodiceIntervento`) VALUES ('7', '7');

```

```
INSERT INTO `polliceverde`.`esecuzione_rinvaso` (`CodiceRinvaso`,
`CodiceIntervento`) VALUES ('8', '8');
INSERT INTO `polliceverde`.`esecuzione_rinvaso` (`CodiceRinvaso`,
`CodiceIntervento`) VALUES ('9', '9');
INSERT INTO `polliceverde`.`esecuzione_rinvaso` (`CodiceRinvaso`,
`CodiceIntervento`) VALUES ('10', '10');
```

```
-- -----
-- Table structure for `INDIVIDUAZIONE`
-- -----
DROP TABLE IF EXISTS `INDIVIDUAZIONE`;
CREATE TABLE `INDIVIDUAZIONE` (
  `Nome` VARCHAR(45) NOT NULL
    REFERENCES `PolliceVerde`.`AGENTEPATOGENO` (`Nome`) ON DELETE NO ACTION
ON UPDATE CASCADE,
  `CodicePatologia` INT(11) NOT NULL,
    FOREIGN KEY (`CodicePatologia`) REFERENCES `PolliceVerde`.`PATOLOGIA`
(`CodicePatologia`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`CodicePatologia`, `Nome`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
-- Records of `INDIVIDUAZIONE`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`individuazione` (`Nome`, `CodicePatologia`) VALUES
('ponderosae', '1');
INSERT INTO `polliceverde`.`individuazione` (`Nome`, `CodicePatologia`) VALUES
('ljungiana', '2');
INSERT INTO `polliceverde`.`individuazione` (`Nome`, `CodicePatologia`) VALUES
('pulchellana', '3');
INSERT INTO `polliceverde`.`individuazione` (`Nome`, `CodicePatologia`) VALUES
('fragariae', '4');
INSERT INTO `polliceverde`.`individuazione` (`Nome`, `CodicePatologia`) VALUES
('pyri', '5');
INSERT INTO `polliceverde`.`individuazione` (`Nome`, `CodicePatologia`) VALUES
('sarta', '6');
INSERT INTO `polliceverde`.`individuazione` (`Nome`, `CodicePatologia`) VALUES
('phloeocoptes', '7');
INSERT INTO `polliceverde`.`individuazione` (`Nome`, `CodicePatologia`) VALUES
('erinea', '8');
INSERT INTO `polliceverde`.`individuazione` (`Nome`, `CodicePatologia`) VALUES
('sheldoni', '9');
INSERT INTO `polliceverde`.`individuazione` (`Nome`, `CodicePatologia`) VALUES
('tulipae', '10');
COMMIT;
```

```
-- -----
-- Table structure for `APPARTENENZA`
-- -----
DROP TABLE IF EXISTS `APPARTENENZA`;
```

```

CREATE TABLE `APPARTENENZA` (
  `CodiceSintomo` INT(11) NOT NULL,
    FOREIGN KEY (`CodiceSintomo`) REFERENCES `PolliceVerde`.`SINTOMO`
  (`CodiceSintomo`) ON DELETE NO ACTION ON UPDATE CASCADE,
  `CodiceSintomatologia` INT(11) NOT NULL,
    FOREIGN KEY (`CodiceSintomatologia`) REFERENCES
  `PolliceVerde`.`SINTOMATOLOGIA` (`CodiceSintomatologia`) ON DELETE NO ACTION ON
  UPDATE CASCADE,
  PRIMARY KEY (`CodiceSintomo`, `CodiceSintomatologia`)) ENGINE=InnoDB DEFAULT
  CHARSET=latin1;

-- -----
--   Records of `APPARTENENZA`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`appartenenza` (`CodiceSintomo`, `CodiceSintomatologia`)
VALUES ('1', '1');
INSERT INTO `polliceverde`.`appartenenza` (`CodiceSintomo`, `CodiceSintomatologia`)
VALUES ('2', '2');
INSERT INTO `polliceverde`.`appartenenza` (`CodiceSintomo`, `CodiceSintomatologia`)
VALUES ('3', '3');
INSERT INTO `polliceverde`.`appartenenza` (`CodiceSintomo`, `CodiceSintomatologia`)
VALUES ('4', '4');
INSERT INTO `polliceverde`.`appartenenza` (`CodiceSintomo`, `CodiceSintomatologia`)
VALUES ('5', '5');
INSERT INTO `polliceverde`.`appartenenza` (`CodiceSintomo`, `CodiceSintomatologia`)
VALUES ('6', '6');
INSERT INTO `polliceverde`.`appartenenza` (`CodiceSintomo`, `CodiceSintomatologia`)
VALUES ('7', '7');
INSERT INTO `polliceverde`.`appartenenza` (`CodiceSintomo`, `CodiceSintomatologia`)
VALUES ('8', '8');
INSERT INTO `polliceverde`.`appartenenza` (`CodiceSintomo`, `CodiceSintomatologia`)
VALUES ('9', '9');
INSERT INTO `polliceverde`.`appartenenza` (`CodiceSintomo`, `CodiceSintomatologia`)
VALUES ('10', '10');
COMMIT;

-- -----
--   Table structure for `TRATTAMENTO`
-- -----
DROP TABLE IF EXISTS `TRATTAMENTO`;
CREATE TABLE `TRATTAMENTO` (
  `Nome` VARCHAR(45) NOT NULL
    REFERENCES `PolliceVerde`.`AGENTEPATOGENO` (`Nome`) ON DELETE NO ACTION
  ON UPDATE CASCADE,
  `CodiceFarmaco` INT(11) NOT NULL,
    FOREIGN KEY (`CodiceFarmaco`) REFERENCES `PolliceVerde`.`FARMACO`
  (`CodiceFarmaco`) ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (`Nome`, `CodiceFarmaco`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```
-- Records of `TRATTAMENTO`
-- -----
BEGIN;
INSERT INTO `polliceverde`.`trattamento` (`Nome`, `CodiceFarmaco`) VALUES
('ponderosae ', '1');
INSERT INTO `polliceverde`.`trattamento` (`Nome`, `CodiceFarmaco`) VALUES
('ljangiana', '2');
INSERT INTO `polliceverde`.`trattamento` (`Nome`, `CodiceFarmaco`) VALUES
('pulchellana', '3');
INSERT INTO `polliceverde`.`trattamento` (`Nome`, `CodiceFarmaco`) VALUES
('fragariae', '4');
INSERT INTO `polliceverde`.`trattamento` (`Nome`, `CodiceFarmaco`) VALUES ('pyri',
'5');
INSERT INTO `polliceverde`.`trattamento` (`Nome`, `CodiceFarmaco`) VALUES ('sarta',
'6');
INSERT INTO `polliceverde`.`trattamento` (`Nome`, `CodiceFarmaco`) VALUES
('phloeocoptes', '7');
INSERT INTO `polliceverde`.`trattamento` (`Nome`, `CodiceFarmaco`) VALUES ('erinea',
'8');
INSERT INTO `polliceverde`.`trattamento` (`Nome`, `CodiceFarmaco`) VALUES
('sheldoni', '9');
INSERT INTO `polliceverde`.`trattamento` (`Nome`, `CodiceFarmaco`) VALUES ('tulipae',
'10');
COMMIT;
```

```
-- -----
-- Funzione per il calcolo della credibilità
-- -----
```

```
DROP FUNCTION IF EXISTS `CalcolaCredibilita`;

DELIMITER $$
CREATE FUNCTION `CalcolaCredibilita`(nickname VARCHAR(45)) RETURNS DOUBLE
BEGIN
DECLARE commentiDiRispostaComplessivi INT(11) DEFAULT 0;
DECLARE media DOUBLE DEFAULT 0;
DECLARE credibilita DOUBLE DEFAULT 0;
SET commentiDiRispostaComplessivi = (
        SELECT COUNT(*)
        FROM COMMENTODIRISPOSTA
        WHERE Nickname = nickname
    );
SET media = ( SELECT AVG(V.Punteggio)
        FROM COMMENTODIRISPOSTA CDR INNER JOIN VALUTAZIONE V
        ON CDR.CodiceCommentoDiRisposta = V.CommentoDiRisposta
```

```

        WHERE CDR.Nickname= nickname
    );
SET credibilita= media * commentiDiRispostaComplessivi;
RETURN media;
END$$
DELIMITER ;

-- -----
-- Funzione per il calcolo dell'indice di manutenzione
-- -----

DROP FUNCTION IF EXISTS `IndiceManutenzione`;

DELIMITER $$
CREATE FUNCTION `IndiceManutenzione`(pianta INT(11)) RETURNS VARCHAR(45)
BEGIN
SET @indiceAccrescimento = (SELECT IndiceAccrescimento
                            FROM ACCRESCIMENTO
                            WHERE Pianta = pianta);

IF @indiceAccrescimento <= 1 THEN
SET @manutenzioneAccrescimento = 2;
ELSEIF @indiceAccrescimento > 1 AND @indiceAccrescimento <=3 THEN
SET @manutenzioneAccrescimento = 5;
ELSE
SET @manutenzioneAccrescimento = 7;
END IF;
SET @volteinQuarantena = (SELECT COUNT(*)
                        FROM QUARANTENA
                        WHERE Pianta = pianta);

IF @volteinQuarantena < 3 THEN
SET @manutenzioneQuarantena = 2;
ELSEIF @volteinQuarantena >= 3 AND @volteinQuarantena <6 THEN
SET @manutenzioneQuarantena = 5;
ELSE
SET @manutenzioneQuarantena = 10;
END IF;

IF @manutenzioneQuarantena + @manutenzioneAccrescimento < 8 THEN
SET @IndiceManutenzione = 'Basso';
ELSEIF @manutenzioneQuarantena + @manutenzioneAccrescimento BETWEEN 8 AND 14 THEN
SET @IndiceManutenzione = 'Medio';
ELSE
SET @IndiceManutenzione = 'Alto';
END IF;

RETURN @IndiceManutenzione;
END$$
DELIMITER ;

UPDATE PIANTA
SET IndiceManutenzione = IndiceManutenzione(CodicePianta);

```



```

-- -----
-- Funzione per il calcolo della distanza minima tra due piante
-- -----

DROP FUNCTION IF EXISTS `DistanzaMinima`;

DELIMITER $$
CREATE FUNCTION `DistanzaMinima`(pianta1 INT(11), pianta2 INT(11)) RETURNS DOUBLE
BEGIN
SET @contenitore1 = (SELECT Contenitore
                      FROM PIANTA
                      WHERE CodicePianta= pianta1);
SET @contenitore2 = (SELECT Contenitore
                      FROM PIANTA
                      WHERE CodicePianta= pianta2);

IF @contenitore1 IS NOT NULL OR @contenitore2 IS NOT NULL THEN
SET @distanzamin = (SELECT MAX(A.AccrescimentoTop)/4
                    FROM PIANTA P INNER JOIN ACCRESCIMENTO A
                    ON P.Accrescimento = A.CodiceAccrescimento
                    WHERE P.CodicePianta = pianta1 OR P.CodicePianta =
pianta2
                    );
ELSE
SET @terreno1 = (SELECT Terreno
                 FROM PIANTA
                 WHERE CodicePianta= pianta1);
SET @terreno2 = (SELECT Terreno
                 FROM PIANTA
                 WHERE CodicePianta= pianta2);
SET @fattoreElementi = (SELECT MAX(ABS(ED1.PercentualePresenza -
ED2.PercentualePresenza))/4
                        FROM ELEMENTODISCIOLTO ED1 INNER JOIN
ELEMENTODISCIOLTO ED2
                        ON (ED1.Terreno = @terreno1 AND
ED2.Terreno = @terreno2
                        AND ED1.Nome = ED2.Nome));
SET @fattoreAccrescimento = (SELECT MAX(A.IndiceAccrescimento)/4
                             FROM PIANTA P INNER JOIN ACCRESCIMENTO A
                             ON P.Accrescimento =
A.CodiceAccrescimento
                             WHERE P.CodicePianta = pianta1 OR
P.CodicePianta = pianta2
                             );
SET @distanzamin = GREATEST(@fattoreElementi,@fattoreAccrescimento);
END IF;
SET @verifica1 = (SELECT Infestante
                  FROM PIANTA
                  WHERE CodicePianta= pianta1);
SET @verifica2 = (SELECT Infestante

```

```

                FROM PIANTA
                WHERE CodicePianta= pianta2);
IF @verifica1 = 1 OR @verifica2 = 1 THEN
SET @distanzamin = @distanzamin * 2;
END IF;

RETURN @distanzamin;
END$$
DELIMITER ;

-- -----
-- Procedura per la funzione analytic: Smart Design
-- -----

DROP PROCEDURE IF EXISTS `SmartDesign`;

DELIMITER $$
CREATE PROCEDURE `SmartDesign` (IN _costomax FLOAT, IN _periodofioritura INT(11), IN
_indicemanutenzione VARCHAR(45))
BEGIN
SELECT *
FROM PIANTA
WHERE IndiceManutenzione = _indicemanutenzione
      AND Prezzo <= _costomax
      AND PeriodoFioritura = _periodofioritura;
END$$
DELIMITER ;

-- -----
-- Procedura per la funzione analytic: Indagini Statistiche
-- -----

DROP PROCEDURE IF EXISTS `IndaginiStatistiche`;

DELIMITER $$
CREATE PROCEDURE `IndaginiStatistiche` (IN _patologia INT(11), IN _inizioPeriodo DATE,
IN _finePeriodo DATE)
BEGIN
SELECT CodiceSezione, Temperatura, Umidita, Irrigazione, Illuminazione
FROM SEZIONE
WHERE CodiceSezione IN (SELECT Sezione
                        FROM PIANTA
                        WHERE CodicePianta in (SELECT Pianta
                        FROM
QUARANTENA
                        WHERE
Patologia = _patologia
                        AND
DataInizio - INTERVAL 1 DAY >= _inizioPeriodo
                        AND
DataInizio - INTERVAL 1 DAY <= _finePeriodo

```

```

);

END$$
DELIMITER ;

-- -----
-- Procedura per la funzione analytic: Reporting
-- -----

DROP PROCEDURE IF EXISTS `Reporting`;

DELIMITER $$

CREATE PROCEDURE `Reporting` (IN _inizio DATE, IN _fine DATE)
BEGIN

CREATE OR REPLACE VIEW quarantene AS
SELECT Q.*, COUNT(*) AS NumQuarantene
FROM QUARANTENA Q INNER JOIN PIANTA P ON P.CodicePianta = Q.Pianta
WHERE P.Cultivar = 1
GROUP BY Q.Pianta;

CREATE OR REPLACE VIEW ordini AS
SELECT O.*, COUNT(*) AS NumOrdini
FROM ORDINE O INNER JOIN PIANTA P ON P.CodicePianta = O.Pianta
WHERE P.Cultivar = 1
GROUP BY O.Pianta;

(
    SELECT *, 'Elevata manutenzione'
    FROM (
        SELECT Q.Pianta, Q.CodiceQuarantena, Q.NumQuarantene,
            (SELECT 1 + COUNT(*)
             FROM quarantene Q0
             WHERE Q0.NumQuarantene > Q.NumQuarantene
              AND Q0.DataInizio BETWEEN _inizio AND _fine
            ) AS Rank
        FROM quarantene Q
    ) AS D
    WHERE D.Rank <=3
    ORDER BY D.Pianta, D.Rank
) UNION (
    SELECT *, 'Pochi ordini'
    FROM (
        SELECT O.Pianta, O.CodiceOrdine, O.NumOrdini,
            (SELECT 1 + COUNT(*)
             FROM ordini O0
             WHERE O0.Data BETWEEN _inizio AND _fine
              AND O0.NumOrdini < O.NumOrdini
            ) AS Rank
    )

```

```

                FROM ordini O
            ) AS D
        WHERE D.Rank <=3
        ORDER BY D.Pianta, D.Rank
    );
END$$

DELIMITER ;

CREATE OR REPLACE VIEW ordiniperreport AS
SELECT O.*, COUNT(*) AS NumOrdini
FROM ORDINE O INNER JOIN PIANTA P ON P.CodicePianta = O.Pianta
GROUP BY O.Pianta;

CREATE OR REPLACE VIEW ordininitarget AS
(SELECT Nome, Genere
FROM PIANTA
WHERE CodicePianta IN (
                        SELECT Pianta
                        FROM ORDINE
                        WHERE Stato = 'Pendente'
                        ))
UNION(
SELECT D.Nome, D.Genere
FROM (
        SELECT P.Nome, P.Genere,
        (SELECT 1 + COUNT(*)
        FROM ordiniperreport O0 INNER JOIN PIANTA P0 ON P0.CodicePianta =
O0.Pianta
        WHERE P0.Nome = P.Nome
        AND O0.NumOrdini > O.NumOrdini
        ) AS Rank
        FROM ordiniperreport O INNER JOIN PIANTA P ON P.CodicePianta = O.Pianta
    ) AS D
WHERE D.Rank <=3
ORDER BY D.Nome, D.Genere, D.Rank);

END$$

DELIMITER ;

-- -----
-- Table structure for `REPORT_ORDINE`
-- -----
DROP TABLE IF EXISTS `REPORT_ORDINE`;
CREATE TABLE `REPORT_ORDINE` (
  `CodiceReport` INT(11) NOT NULL DEFAULT 1,
  `NomePianta` VARCHAR(45) NOT NULL,
  `GenerePianta` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CodiceReport`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```
-- -----  
--  Records of `REPORT_ORDINE`  
-- -----  
BEGIN;  
INSERT INTO `REPORT_ORDINE`  
SELECT (SELECT IF(EXISTS (SELECT CodiceReport  
                        FROM REPORT_ORDINE), MAX(CodiceReport)+1, 1)  
      FROM REPORT_ORDINE) AS CodiceReport, 0.*  
FROM ordininitarget 0  
COMMIT;
```

9. FUNZIONALITA' DI BACK-END E DOCUMENTAZIONE DELLE SCELTE DI IMPLEMENTAZIONE

INDICE DI ACCRESCIMENTO

È richiesta la formulazione di un criterio di calcolo dell'indice di accrescimento di un esemplare di pianta e la soluzione proposta è:

Indice Accrescimento = max (accrescimento top, accrescimento root)

Si tratta di un calcolo molto semplice fatto sugli attributi della tabella accrescimento che però sembra essere quanto di più vicino alla realtà delle cose ci sia e contemporaneamente ottimizza lo sfruttamento degli spazi.

Esso indica quanto velocemente cresce una pianta e quindi per ricavarne la grandezza basterà moltiplicare l'indice di accrescimento per l'età.

INDICE DI MANUTENZIONE

È richiesta la formulazione di un criterio per il calcolo dell'indice di manutenzione di un esemplare di pianta. Per calcolarlo abbiamo deciso di attribuire un valore alla manutenzione dovuta all'indice d'accrescimento (che chiameremo X) e uno alla manutenzione dovuta al numero di volte che la pianta è finita in quarantena (e lo chiameremo Y).

$X = 2$

se l'indice d'accrescimento è strettamente minore di 1;

$X = 5$

se l'indice d'accrescimento è compreso tra 1 e 3;

$X = 7$

se l'indice d'accrescimento è strettamente maggiore di 3;

$Y = 2$

se la pianta è stata in quarantena meno di 3 volte;

$Y = 10$

se la pianta è stata in quarantena più di 6 volte;

$Y = 5$

altrimenti;

- Se $X + Y < 8$ l'indice di manutenzione è BASSO
- Se $8 \leq X + Y < 15$ l'indice di manutenzione è MEDIO
- Se $X + Y \geq 15$ l'indice di manutenzione è ALTO

La scala è stata creata dopo aver fatto una media delle esigenze di manutenzione di diversi generi di pianta.

CONFLITTO NELLA DISTANZA MINIMA TRA PIANTE

È richiesta la risoluzione del problema del conflitto esistente tra piante sia nel caso in cui esso sia causato dall'indice d'accrescimento, sia dal fabbisogno, potendo avere la stessa esigenza di elementi disciolti e la soluzione proposta è:

Per quanto riguarda il conflitto che si genera a causa degli elementi disciolti nel terreno, si è deciso di calcolare la differenza in modulo tra le percentuali degli elementi comuni al fabbisogno delle due piante candidate e di scegliere la quarta parte del massimo di tali moduli.

conflitto singolo elemento = $|\% \text{ presenza nel primo esemplare} - \% \text{ presenza nel secondo esemplare}|$

conflitto totale terreno = $\frac{1}{4}$ conflitti tutti gli elementi

L'algoritmo che calcola la distanza minima a cui una pianta deve essere posta rispetto si basa sulla seguente logica:

- Se la pianta risiede in un contenitore la distanza minima è costituita dall'attributo AccrescimentoTop della tabella ACCRESCIMENTO.
- Se le piante sono poste in terreno aperto la distanza è data dal massimo tra la funzione usata per il calcolo della distanza da fabbisogno e l'indice d'accrescimento, in modo che nell'eventuale comparsa di entrambi i conflitti si scelga comunque il più restrittivo:

Distanza min = max (indice di accrescimento, conflitto totale terreno)

- Se si tratta di una pianta infestante si è deciso di raddoppiare il valore finale della distanza minima, calcolato seguendo i parametri precedenti.

CREDIBILITA' DI UN ACCOUNT

È richiesta la formulazione di un criterio per il calcolo della credibilità di un utente nel forum e la soluzione proposta è:

$$\text{credibilità} = \text{numero CommentiDiRisposta pubblicati} * \text{AVG (valutazioni)}$$

Si è optato per una soluzione di questo genere per rendere la credibilità un qualcosa di attendibile su ampia scala e non solo con la valutazione complessiva dei post.

```
delimiter $$
CREATE TRIGGER `ValiditaPeriodoPatologia` BEFORE INSERT ON `PATOLOGIA` FOR EACH ROW
BEGIN
  IF (SELECT Tipo
      FROM PERIODO
      WHERE CodicePeriodo = NEW.PeriodoInCuiColpisce) <> 'Patologia' THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Errore da vincolo di integrita referenziale con la tabella PERIODO';
  END IF;
END $$
delimiter ;
```

Questo trigger viene utilizzato per verificare la validità di una patologia nel momento in cui ne viene indicato il periodo in cui essa colpisce maggiormente. In particolare il trigger `ValiditaPeriodoPatologia` impedisce che si crei una patologia facente riferimento ad un periodo caratterizzante altro.

```
delimiter $$
CREATE TRIGGER `ControlloCodiceTipo` BEFORE INSERT ON `FABBISOGNO` FOR EACH ROW BEGIN
CASE
WHEN NEW.Tipo = 'Concimazione' THEN
  IF NEW.CodiceTipo NOT IN ( SELECT CodiceConcimazione
                          FROM CONCIMAZIONE) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Intervento di concimazione inesistente';
  END IF;
WHEN NEW.Tipo = 'Rinvaso' THEN
  IF NEW.CodiceTipo NOT IN ( SELECT CodiceRinvaso
                          FROM RINVASO) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Intervento di rinvaso inesistente';
END
```



```

END IF;
WHEN NEW.Tipo = 'Potatura' THEN
IF NEW.CodiceTipo NOT IN ( SELECT CodicePotatura
                           FROM POTATURA) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Intervento di potatura inesistente';
END IF;
END CASE;
END $$
delimiter ;

```

Questo trigger viene utilizzato per verificare la validità di un fabbisogno nel momento in cui ne viene indicato il tipo che lo caratterizza. In particolare il trigger `ControlloCodiceTipo` impedisce che si crei un fabbisogno facente riferimento ad un intervento di concimazione, potatura o rinvaso inesistente.

```

delimiter $$
CREATE TRIGGER `ValiditaPeriodoFabbisogno` BEFORE INSERT ON `FABBISOGNO` FOR EACH ROW
BEGIN
IF (SELECT Tipo
    FROM PERIODO
    WHERE CodicePeriodo = NEW.PeriodoConsigliato) <> 'Fabbisogno' THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Errore da vincolo di integrita referenziale con la tabella PERIODO';
END IF;
END $$
delimiter ;

```

Questo trigger viene utilizzato per verificare la validità di un fabbisogno nel momento in cui ne viene indicato il periodo a cui si riferisce. In particolare il trigger `ValiditaPeriodoPatologia` impedisce che si crei un fabbisogno facente riferimento ad un periodo caratterizzante altro.

10. GESTIONE DELL'AREA ANALYTICS

Soluzione proposta per quanto riguarda la funzionalità di Smart Design:

Si selezionano il prezzo limite, il periodo di fioritura e l'indice di manutenzione richiesti dal cliente e si ricavano le piante che potrebbero soddisfare queste esigenze del cliente.

Soluzione proposta per quanto riguarda la funzionalità di Reporting:

Si seleziona il periodo scelto per l'indagine e in base al numero di quarantene subite da ogni esemplare appartenente ad un tipo di cultivar particolare e al numero di ordini che le riguardano, si ottiene una statistica che indica le tre piante (più eventuali pari merito) che sono state vendute di meno e le tre che hanno richiesto maggiormente manutenzione. In questo modo è possibile avere un criterio utile alla creazione di una gestione intelligente del magazzino.

Soluzione proposta per quanto riguarda la funzionalità di Indagini Statistiche:

Data una patologia e un periodo di interesse, vengono riportate la sezione, temperatura, umidità, illuminazione e terreno delle piante che in quel periodo sono state colpite da quella patologia. In questo modo si vuole fare un'analisi delle cause che possono portare all'esordio di una patologia.