UNIVERSITÀ DEGLI STUDI DI UDINE
hic sunt futura

DIPARTIMENTO DI SCIENZE MATEMATICHE, INFORMATICHE E FISICHE

TESI DI LAUREA IN
INFORMATICA

Modellazione e realizzazione di una base di dati per il monitoraggio del batterio legionella

CANDIDATO
Francesco Bortuzzo

RELATORE
Professor Angelo Montanari

CORRELATORE
Dottor Andrea Brunello
Dottor Nicola Saccomano

Anno Accademico 2023/2024

Indice

1. Introduzione

1.a. Introduzione al batterio Legionella

Il batterio Legionella è un bacillo gram-negativo aerobio, non mobile, che prospera in ambienti acquatici e umidi, sia naturali, come acque sorgive, termali, di fiumi o laghi, sia artificiali, come tubature, serbatoi, fontane e piscine. La Legionella è in grado di sopravvivere in una vasta gamma di condizioni ambientali, tra cui temperature comprese tra venti e quarantacinque gradi Celsius, pH neutro o leggermente alcalino, e presenza di nutrienti organici.

Il genere comprende sessantadue specie diverse, suddivise in settantuno sierotipi, di cui circa venti sono patogeni per l'uomo. La specie più comune è *Legionella pneumophila*, responsabile della maggior parte dei casi riportati di legionellosi[1]. La malattia può essere contratta inalando aerosol contenenti il batterio, come quelli prodotti da docce, fontane, impianti di condizionamento o umidificatori.

È quindi di fondamentale importanza monitorare la diffusione di questo batterio negli ambienti umidi e acquatici. Particolare attenzione deve essere rivolta alle strutture ospedaliere, termali e alberghiere, che rappresentano per loro natura ambienti a rischio di diffusione del batterio.

1.b. Legionella in Friuli Venezia Giulia

A livello europeo, la raccolta di dati relativi alla presenza del batterio è effettuata dall'ECDC[2]. Nel nostro Paese, invece, questa attività è svolta da diversi enti e istituzioni. Un contributo significativo proviene dall'Istituto Superiore di

[1] Legionellosi, o malattia del legionario, è una malattia infettiva che si presenta con sintomi simili all'influenza, come febbre, tosse, dolori muscolari e mal di testa. In alcuni casi, può evolvere in una forma polmonare, con sintomi analoghi a quelli della polmonite, e comportare complicazioni gravi, come polmonite atipica o decesso.

[2] Centro europeo per la prevenzione e il controllo delle malattie, istituito nel 2005.

Sanità e dai vari organismi che costituiscono il SNPA[3], di cui fa parte l'ARPA FVG[4].

I dati raccolti sono utilizzati per valutare il rischio di diffusione del batterio e adottare le misure di prevenzione e controllo indicate dal Ministero della Salute nelle "Linee guida per la prevenzione ed il controllo della legionellosi"[5].

In questo ambito, l'ARPA FVG ha effettuato numerose indagini sul territorio e ha pubblicato i risultati in vari report. Ad esempio, nel 2019, una collaborazione con l'Università degli Studi di Udine ha portato alla pubblicazione di un articolo[6] relativo alla presenza di Legionella nei sistemi di raccolta e distribuzione dell'acqua nella regione. Lo studio ha coperto un periodo di sedici anni, dal 2002 al 2017, durante il quale sono stati raccolti e analizzati 20.319 campioni attraverso 3.983 indagini ambientali.

I risultati riferiti alle indagini non clinciche e dunque eseguite routinariamente nell'ambito del piano regionale di sorveglianza ambientale hanno evidenziato che la presenza di Legionella è diffusa sopratutto nei cluster di impianti termali, nei quali il batterio è stato individuato nel 57,8% dei siti indagati, e in quelli ospedalieri, in cui nel 50,8% delle strutture è stata riscontrata la Legionella almeno una volta, con picchi dei campioni positivi soprattutto nei mesi che segnano l'inizio del periodo autunnale.

Inoltre, si è osservato che la presenza del batterio ha registrato un notevole incremento tra la seconda metà del 2006 e l'inizio del 2009, seguito da una diminuzione fino al 2013 e da un nuovo aumento negli anni successivi. Questo andamento indica chiaramente che, per ridurre il rischio di diffusione del batterio, è essenziale implementare un piano di prevenzione adeguato, che comprenda sia la manutenzione degli impianti sia la sorveglianza ambientale.

1.c. Basi di dati relazionali

Pur riconoscendo l'importanza cruciale della sorveglianza ambientale per il controllo della legionellosi, in Friuli Venezia Giulia, come in molte altre regioni, manca un sistema efficiente per la memorizzazione, la gestione e l'analisi dei dati raccolti. Tale carenza rende estremamente oneroso lavorare con la mole di

[3]Sistema Nazionale per la Protezione dell'Ambiente

[4]Agenzia Regionale per la Protezione dell'Ambiente Friuli Venezia Giulia.

[5][1] M. della Salute, «Linee guida per la prevenzione e il controllo della legionellosi», 2015, [Online]. Disponibile su: https://www.salute.gov.it/portale/malattieInfettive/dettaglioPubblicazioniMalattieInfettive.jsp?id=2362

[6][2] A. Felice, M. Franchi, S. De Martin, N. Vitacolonna, L. Iacumin, e M. Civilini, «Environmental surveillance and spatio-temporal analysis of Legionella spp. in a region of northeastern Italy (2002–2017)», *PLOS ONE*, vol. 14, fasc. 7, p. e218687, 2019, doi: 10.1371/journal.pone.0218687.

informazioni raccolte nelle indagini ambientali, ostacolando così lo svolgimento di analisi e ricerche mirate. Inoltre, tale mancanza rende poco sicura la conservazione dei dati. La memorizzazione delle informazioni in file di testo o in fogli di calcolo, infatti, non assicura né la protezione né l'integrità dei dati, che potrebbero essere facilmente persi, alterati o resi inconsistenti a causa di errori umani.

In questo contesto, i sistemi di basi di dati giocano un ruolo fondamentale, in quanto permettono di memorizzare grandi quantità di dati e di effettuare ricerche complesse in modo rapido ed efficiente.

In particolare, il modello relazionale rappresenta la soluzione più largamente adottata per la memorizzazione delle informazioni. Tale rappresentazione, introdotta per la prima volta dall'informatico inglese Edgar F. Codd attraverso la pubblicazione, nel 1970, dell'articolo "*A Relational Model of Data for Large Shared Data Banks*"[7], sfrutta il concetto di relazione, nella sua coniugazione matematica, per organizzare i dati mediante tabelle, che rappresentano l'insieme delle entità coinvolte nel sistema e dele associazioni tra di esse.

Nello specifico, l'architettura relazionale si propone di rappresentare il dominio informativo, ovvero l'insieme dei dati che si vogliono memorizzare, attraverso tabelle identificate da un nome, con un numero fisso di colonne, dette attributi, e un insieme di righe, denominate tuple, tutte distinte tra loro. Questa struttura, in contrapposizione a quelle precedentemente in voga, come quella gerarchica o reticolare, si è rivelata particolarmente adatta per la memorizzazione di dati e per la gestione delle relazioni tra essi. In particolare, l'efficacia del modello relazionale si deve a caratteristiche fondamentali, come la possibilità di definire chiavi primarie ed esterne e la gestione delle operazioni sui dati mediante software specializzati, ovvero i RDBMS[8]. Questi sistemi assicurano, tra le altre, la corretta applicazione di vincoli di integrità e l'esecuzione di transazioni ACID[9], che garantiscono la coerenza dei dati in ogni momento e la loro persistenza nel tempo.

L'adozione del modello relazionale rimane prevalente nell'ambito della gestione dei dati, nonostante siano emerse soluzioni alternative come i database NoSQL. Tale successo è attribuibile diversi fattori: la struttura a tabelle del modello garantisce intuitività e chiarezza nella rappresentazione dei dati e favorisce elevati standard di sicurezza e integrità dei dati. Inoltre è supportata l'indipendenza dei dati, ovvero la possibilità di modificare la struttura del database senza dover modificare le applicazioni che vi accedono. Infine, l'introduzione

[7] [3] E. F. Codd, «A relational model of data for large shared data banks», *Communications of the ACM*, vol. 13, fasc. 6, pp. 377–387, 1970, doi: 10.1145/362384.362685.

[8] Relational Database Management Systems

[9] Acronimo per Atomicity, Consistency, Isolation, Durability

dei RDBMS ha assicurato, grazie all'integrazione degli indici, che migliorano l'efficienza delle operazioni, la definizione di linguaggi specifici, come SQL, e l'adozione di alcuni linguaggi di programmazione noti, tra cui python, che il modello relazionale rimanga una soluzione attuale e facilmente adattabile alle esigenze più moderne.

Nell'ambito di questa tesi si ritiene opportuno implementare il sistema tramite un database relazionale. Tale scelta è motivata dalla necessità di garantire la sicurezza, l'integrità e la coerenza dei dati raccolti durante il monitoraggio della Legionella. Inoltre, si ritiene fondamentale fornire un sistema intuitivo e facilmente integrabile con le applicazioni pre-esistenti già in uso dai ricercatori dell'ARPA. In questo contesto, si ritiene che il modello relazionale e, più specificamente, il RDBMS PostgreSQL rappresentino una soluzione appropriata per la memorizzazione e la gestione dei dati. D'altra parte, soluzioni alternative, come i database NoSQL, potrebbero presentare limitazioni in termini di integrità e coerenza dei dati, senza tuttavia garantire vantaggi significativi in termini di prestazioni.

1.d. Obiettivo della tesi

Il presente elaborato si propone di delineare gli aspetti principali per la progettazione di un database relazionale destinato alla memorizzazione dei dati relativi alla diffusione della Legionella. Più specificamente, nel capitolo 2 viene condotta un'analisi critica approfondita di una soluzione pre-esistente, individuandone vantaggi e criticità. Tale analisi costituisce la base per le modifiche proposte nel capitolo successivo, dove vengono introdotte soluzioni migliorative volte ad adattare il sistema alle nuove esigenze emerse durante i colloqui condotti in collaborazione con i ricercatori dell'ARPA FVG. La sezione 4, invece, è dedicata alla ristrutturazione dello schema originario e alla sua traduzione in un modello logico. Infine, il capitolo 5 si concentra sull'implementazione della base di dati, ponendo particolare attenzione alla definizione dei domini e dei vincoli che garantiscono l'integrità dei dati, oltre che la definizione di alcune relative all'inserimento e alla modifica dei dati.

2. Analisi critica di una soluzione pre-esistente

Come accennato nel capitolo introduttivo, una delle principali sfide riscontrate nell'attuale sistema di gestione dei dati riguarda la realizzazione di soluzioni efficienti per la memorizzazione delle informazioni raccolte durante le indagini ambientali. In questa sezione si procede a un'analisi critica di un database relazionale utilizzato per archiviare i dati relativi alla diffusione della Legionella. Il database oggetto di analisi è stato sviluppato dal dottor Dario Garlatti nell'ambito della sua tesi di laurea triennale in informatica, dal titolo "Base di dati e applicazione web per il monitoraggio del batterio della Legionella"[10].

2.a. Requisiti

Prima di procedere con lo studio del database, è necessario definire i requisiti del sistema informativo. Questi sono di natura qualitativa e descrivono le caratteristiche che il sistema deve possedere per soddisfare le esigenze degli utenti e degli stakeholder. I criteri alla base della progettazione della soluzione in analisi riguardano l'intera fase di acquisizione dei dati relativi alle indagini ambientali portate a termine dai ricercatori di ARPA FVG per il monitoraggio della Legionella in regione.

Di seguito sono riportati i requisiti, non strutturati, che hanno guidato la progettazione della base di dati.

Il sistema deve consentire la registrazione delle indagini ambientali relative alla presenza di Legionella nei sistemi di adduzione e conservazione dell'acqua. Ogni indagine è definita dal tipo, dalla data e dal sito presso il quale viene eseguita, ed è, eventualmente, associata al richiedente qualora si tratti di un'indagine di follow-up. Un sito è identificato dall'indirizzo e dalla categoria di appartenenza. Le indagini comprendono il prelievo di campioni, ciascuno dei quali è associato ad una e una sola indagine. Tali campioni sono caratterizzati dalla

[10][4] D. Garlatti, «Base di dati e applicazione web per il monitoraggio del batterio della legionella», 2020.

temperatura, caldo o freddo, al momento dell'estrazione e dal punto di prelievo, all'interno del sito presso cui è svolta l'indagine cui afferiscono, e sono univocamente identificati da un codice. Tutti i campioni prelevati sono sottoposti a diverse analisi per accertare la presenza di Legionella. Si annoverano: la PCR qualitativa, che consente di rilevare la presenza del DNA del batterio; la PCR quantitativa, che misura la concentrazione di Legionella nei campioni, espressa in µg/l; l'analisi colturale, che consente di isolare e identificare le unità formanti colonia (UFC_L) e, in caso di positività, di determinare il sierogruppo.

2.a.I. Note

Si segnala che la PCR non costituisce un metodo diagnostico definitivo per la legionellosi, ma piuttosto un test di screening che necessita di conferma attraverso la coltura. Infatti, «poiché, così come specificato nella norma ISO "*Water quality- Detection and quantification of Legionella spp and/or Legionella pneumophila by concentration and genic amplification by quantitative polymerase chain reaction (qPCR)*" (ISO/TS 12869, 2012), la qPCR non da informazione riguardo lo stato delle cellule, la quantificazione dovrà sempre essere determinata mediante esame colturale»[11].

Inoltre, si osserva che i metodi analitici utilizzati per la rilevazione del batterio, come indicato nell'allegato 4 delle "Linee Guida per la prevenzione e il controllo della legionellosi"[12], variano in base alla matrice da analizzare (acqua, biofilm, aria); tuttavia, i risultati ottenuti sono espressi in modo uniforme, a prescindere dal tipo di analisi effettuata. Pertanto, considerata l'esigenza di conservare le informazioni relative ai risultati delle analisi sui campioni, si ritiene lecito mantenere le tre tipologie di analisi sopra menzionate, senza ulteriori distinzioni.

2.b. Schema concettuale-logico

Di seguito viene presentato lo schema concettuale-logico del database sviluppato dal dottor Garlatti. Tale schema è stato modellato utilizzando il linguaggio IDEF1X[13]. Questo linguaggio appartiene alla famiglia dei linguaggi di modellazione IDEF[14]. Per una corretta comprensione dello schema, è essenziale definire i concetti di entità e relazione, che rappresentano i fondamenti della modellazione dei dati.

[11] [1],«Linee guida per la prevenzione ed il controllo della legionellosi», p. 21
[12] [1], «Linee guida per la prevenzione ed il controllo della legionellosi», p. 91
[13] Integration DEFinition for information modeling.
[14] https://www.idef.com/

2.B.I. Notazione IDEF1X

Nella notazione IDEF1X, le entità sono rappresentate attraverso tabelle contenenti attributi che ne descrivono le proprietà, e ciascuna entità è identificata da una chiave primaria, costituita da un singolo attributo o da una combinazione di attributi in grado di identificare univocamente ogni riga della tabella. Un'entità può essere classificata come indipendente se può essere identificata senza necessità di relazioni con altre entità, mentre si considera dipendente quando il suo significato emerge solo in relazione a un'altra tabella associata.

Le relazioni di connessione, o associazioni, sono rappresentate mediante linee che collegano due entità, segnalando l'esistenza di un legame tra di esse. In particolare, si distinguono due tipi di relazioni: le associazioni identificative, in cui l'entità figlia è identificata in relazione all'entità genitore e la cui chiave primaria include quella del genitore, rappresentate da una linea continua; le associazioni non identificative, in cui l'entità figlia è comunque identificata in relazione all'entità genitore, ma la chiave primaria della figlia non include quella del genitore, rappresentate da una linea tratteggiata. La cardinalità di queste associazioni è indicata da lettere: "p" denota una relazione uno a uno o uno a molti, "z" indica una relazione uno a zero o uno a uno, e "n" specifica una relazione uno a esattamente n.

Le relazioni di categorizzazione, invece, sono rappresentate da linee che collegano un'entità genitore a una o più entità figlie, sottolineando che queste ultime ereditano le proprietà dell'entità genitore, pur mantenendo attributi distintivi. Le entità di categoria[15] sono mutuamente esclusive e si distinguono grazie a un attributo discriminatore, il cui valore è univoco per ciascuna entità di categoria. Esistono due tipologie di categorizzazione: le categorizzazioni complete, in cui ogni entità genitore deve essere associata a una figlia, rappresentate da un pallino vuoto e due linee; le categorizzazioni incomplete, in cui un'entità genitore può non essere associata a nessuna entità figlia, rappresentate da un pallino pieno e una linea.

[15]Entità che costituisce un sottotipo di un'altra.

2.B.II. Schema concettuale-logico

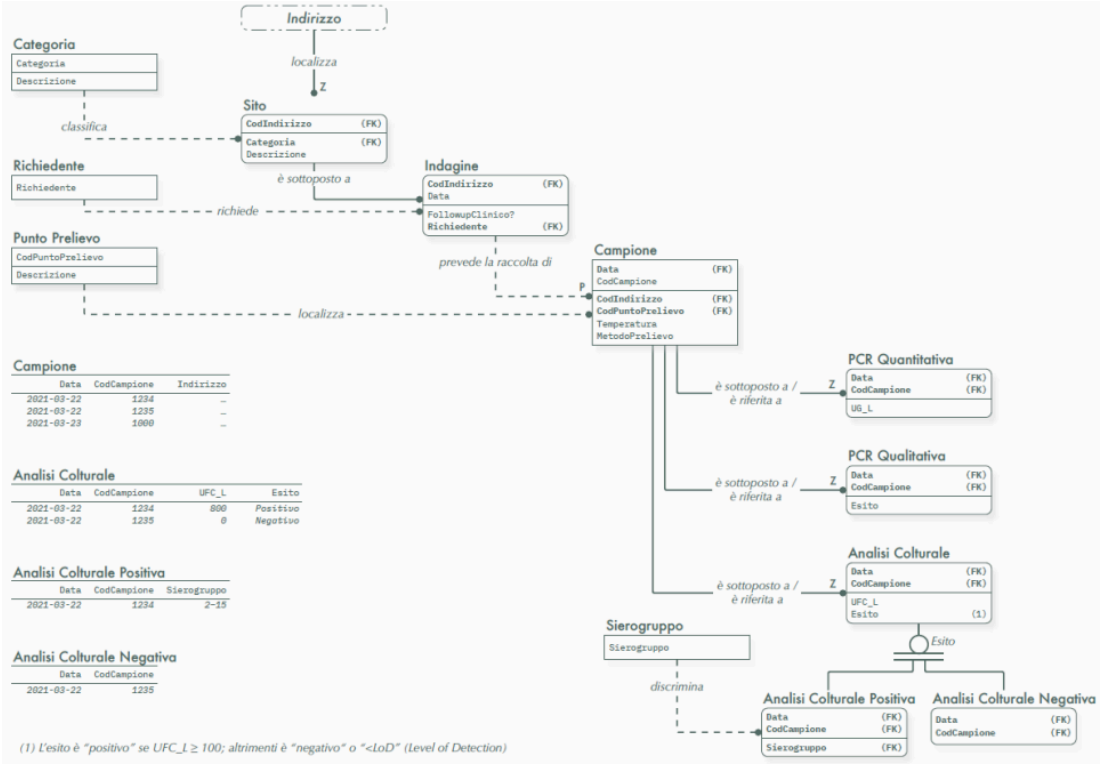

Figura 1: Diagramma ER

2.C. Analisi dello schema: Considerazioni e proposte di modifica

Lo schema illustrato è stato concepito per rispondere ai requisiti di memorizzazione dei dati relativi alla diffusione della Legionella. Tuttavia, durante una prima fase di analisi del database, sono stati individuati alcuni difetti che richiedono un'accurata valutazione e una eventuale revisione dello schema.

Alcune entità, come *indirizzo* e *categoria*, sono state inizialmente progettate come entità autonome, ma potrebbe essere più efficace trattarle come attributi dell'entità *sito*. Questo approccio non solo semplificherebbe lo schema, ma migliorerebbe anche la sua chiarezza strutturale. In particolare, l'attributo descrizione dell'entità *categoria* è superfluo, poiché il nome della categoria dovrebbe essere sufficiente per identificarla in modo univoco. Inoltre, l'aggiunta di un attributo nome all'entità *sito* potrebbe facilitare la consultazione dei dati, specialmente per quanto riguarda gli ospedali, che sono generalmente riconosciuti dalla combinazione di nome e città, piuttosto che unicamente dall'indirizzo. In aggiunta, si propone di arricchire l'entità sito con nuovi attributi che ne descrivano le caratteristiche principali nel contesto specifico. Questi attributi

includono dettagli sull'impiantistica del sito, come la tipologia di caldaia, il materiale delle tubature, l'uso del cloro, e altre informazioni di carattere generale, come l'anno dell'ultima ristrutturazione.

Un ulteriore elemento di riflessione riguarda l'associazione del *richiedente* alle *indagini ambientali*. Superando quanto indicato nei requisiti, si ritiene opportuno che l'entità *richiedente* sia messa in relazione con indagini che non siano unicamente di follow-up. Inoltre, si suggerisce l'introduzione di una nuova entità denominata *follow-up clinico*, associata a una o più indagini ambientali. Questa modifica si dimostra particolarmente efficace nella gestione dei dati relativi ai pazienti affetti da legionellosi e nella valutazione del rischio di diffusione del batterio. Infatti, «per avere un quadro globale della situazione, è fondamentale disporre, per ciascun paziente affetto da legionellosi, di informazioni precise su una eventuale esposizione a rischio nei dieci giorni precedenti l'insorgenza dei sintomi»[16]. La possibilità di associare un paziente a una o più indagini ambientali risulterebbe, dunque, vantaggiosa.

L'entità *follow-up clinico* potrebbe essere ulteriormente arricchita con attributi volti a descrivere il paziente e la sua esposizione al rischio, quali la data di insorgenza dei sintomi, il luogo di residenza, il luogo di lavoro e le attività svolte nei dieci giorni precedenti l'insorgenza dei sintomi. Questi dettagli, tuttavia, non sono modellati nello schema attuale né saranno inclusi nello schema finale, poiché non sono stati considerati nei requisiti né approfonditi con i ricercatori. Ciononostante, potrebbero rivelarsi utili per una valutazione più accurata del rischio di diffusione del batterio.

Per quanto concerne l'entità *campione*, è opportuno valutare l'introduzione di un attributo volume per specificare la quantità d'acqua prelevata per l'analisi. Sebbene non strettamente necessario, tale attributo trova pertinenza nel definire parametri di riferimento relativi al prelievo dei campioni, come il volume minimo d'acqua richiesto per eseguire tutte le analisi previste. Inoltre, poiché è possibile prelevare campioni di diversa matrice ambientale, come acqua, biofilm o aria, si presenta la proposta di introdurre un attributo "matrice" che consenta di specificare il tipo di campione analizzato.

Infine, si propone di riorganizzare la disposizione delle entità *indagine ambientale* e *campione* all'interno dello schema. In particolare, per come definita nel glossario, un'indagine ambientale non è altro che una collezione di campioni prelevati in un sito specifico in una data determinata. Pertanto, risulta più coerente associare solo l'entità *campione* alle informazioni spaziali contenute nelle tabelle *punto di prelievo* e *sito*. Si noti che tale modifica comporta l'introduzione di un vincolo di integrità che stabilisce che tutti i campioni associati a un'indagine devono essere prelevati nello stesso sito.

[16][1], «Linee guida per la prevenzione ed il controllo della legionellosi», p. 30

In questo contesto, appare vantaggioso apportare una modifica alla struttura delle entità *sito* e *punto di prelievo* nel modo seguente: si consiglia di aggiungere l'attributo coordinate all'entità *sito*, associandolo a una coppia di coordinate, ad esempio riferite al centro geografico o all'ingresso principale dell'edificio, che costituirebbero una chiave per l'entità. Inoltre, l'entità *punto di prelievo* potrebbe essere trasformata in un'entità debole rispetto al *sito*, implicitando il vincolo imposto dall'associazione di un punto di prelievo a un sito, secondo il quale un punto di prelievo deve essere situato all'interno del perimetro del sito di cui fa parte. Al *punto di prelievo* potrebbero essere attribuite proprietà che ne descrivano la posizione all'interno del sito, come il piano, la stanza o il tipo di componente idraulico, da cui è stato prelevato il campione.

Complessivamente, gli adeguamenti proposti esercitano un impatto positivo sulla gestione dei vincoli di integrità del database, poiché risultano logicamente più immediati e più facili da implementare rispetto alle soluzioni precedenti, e contribuiscono a fornire una visione ordinata e completa dei dati relativi alla diffusione della Legionella.

2.c.I. Diagramma E-R che raccoglie le modifiche proposte

A seguito di queste considerazioni, si propone una revisione dello schema. La nuova versione è modellata secondo la notazione classica E-R[17] che consente di rappresentare in modo chiaro e conciso le entità, le relazioni e gli attributi del database.

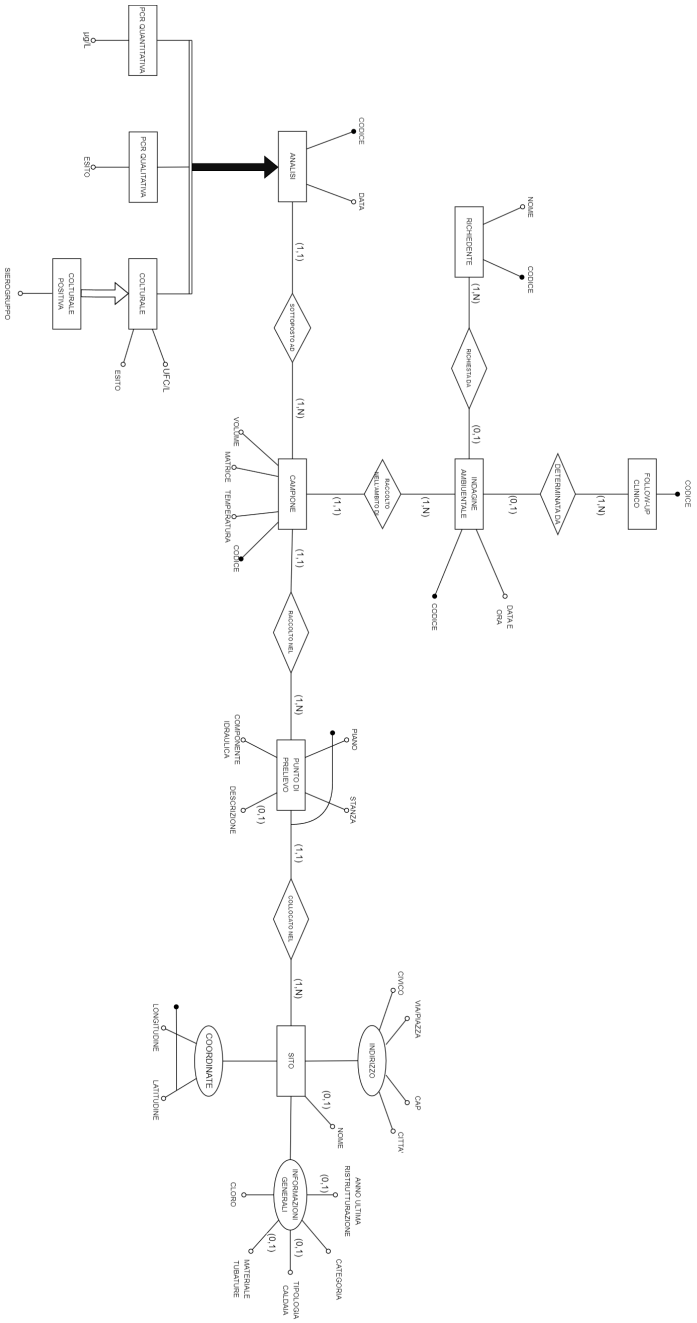

Figura 2: Diagramma ER

[17][5] P. Atzeni, S. Ceri, S. Paraboschi, e R. Torlone, *Database Systems: Concepts, Languages & Architectures*. 1999.

3. Integrazione dei nuovi requisiti nella base di dati

Come accennato in precedenza, la progettazione concettuale della base di dati deve essere adeguata alle nuove esigenze emerse a seguito dei colloqui con i ricercatori di ARPA FVG. In questa sezione si procede con l'integrazione dei nuovi requisiti nella base di dati, partendo dallo schema concettuale proposto in conclusione del capitolo precedente.

3.a. Requisiti e proposte di modfifica dello schema

Le nuove informazioni sono finalizzate a rendere la base di dati più completa e funzionale. In particolare, è stata considerata l'opportunità di introdurre ulteriori entità e attributi, allo scopo di memorizzare dati aggiuntivi relativi ai campioni raccolti nel corso delle indagini ambientali e ai siti coinvolti. Di seguito sono elencati i requisiti, non strutturati, che hanno guidato l'integrazione dei nuovi elementi e le corrispondenti proposte di modifica dello schema.

Dati meteorologici

Si ritiene opportuno mantenere le informazioni relative agli aspetti meteorologici e climatici dei siti in cui vengono condotte le indagini ambientali, poiché tali dati possono essere utili per valutare l'impatto delle condizioni ambientali sulla diffusione del batterio e per individuare eventuali correlazioni tra la presenza di Legionella e particolari fattori climatici. Tali informazioni sono raccolte presso le stazioni meteorologiche presenti sul territorio e comprendono dati relativi a temperatura, umidità e pressione atmosferica. Nella base di dati si propone di introdurre un'entità denominata *stazione meteorologica*, identificata dalla posizione geografica, che può essere rappresentata attraverso l'indirizzo oppure le coordinate, e che conserva i dati meteorologici raccolti. Questa entità è associata alla tabella *sito* nel seguente modo: ogni sito è in relazione con la stazione meteorologica più vicina, la quale forsnisce i dati relativi alle condizioni climatiche del luogo.

Analisi del pH

Una seconda considerazione riguarda l'opportunità di ampliare il campo di azione delle analisi condotte sui campioni prelevati durante le indagini ambientali. In particolare, si suggerisce di introdurre un nuovo tipo di analisi, denominata *analisi del pH*, volta a misurare il livello di acidità o alcalinità dell'acqua campionata. Questo parametro è di fondamentale importanza per valutare la qualità dell'acqua e la presenza di Legionella, poiché il batterio prospera in acque con pH neutro o leggermente alcalino.

Informazioni genomiche

Sempre in relazione alle analisi condotte sui campioni, durante i colloqui è emerso il proposito di memorizzare le informazioni genomiche relative al batterio. In particolare, si intende raccogliere dati sulla presenza, o assenza, di specifici geni e individuare i fattori genetici che influenzano la diffusione del batterio. A tale scopo, è necessario eseguire un'analisi genomica sui campioni prelevati per identificare la sequenza del DNA di Legionella. Questa informazione è memorizzata in un'entità *analisi genomica*, che rappresenta una specializzazione dell'entità *analisi*, e contiene l'intera sequenza del DNA di Legionella, espressa mediante le quattro lettere che indicano le basi azotate (A, T, C, G).

A ciascun genoma sequenziato si intende associare i geni noti di Legionella, presenti nei database di riferimento di BLAST[18] corrispondenti. Tali geni sono memorizzati in un'entità *gene*, identificata univocamente mediante una chiave corrispondente al relativo protein ID[19] e caratterizzata dal nome del gene, se presente nel database utilizzato per l'analisi. A questa entità, che ha lo scopo di conservare informazini stabili e ben definite sui geni noti di Legionella, si propone di associare un'entità *gene del genoma*, che rappresenta i geni individuati per ogni genoma sequenziato. Si tiene traccia, tramite i principali parametri restituiti dalle query BLAST, del fattore di similarità tra i geni noti e quelli individuati tramite l'analisi. Questo approccio ha lo scopo di consentire, in futuro, a seguito del progresso delle tecniche di riconoscimento genetico e dell'aumento dei dati disponibili, una rivalutazione dei geni identificati, al fine di determinare se emergono geni con maggiore somiglianza rispetto a quelli attualmente presenti nel genoma analizzato. Ogni entry della tabella *gene di un genoma* è distinta dall'insieme formato dal protein-ID, dal genoma di cui è parte e dalla posizione assoluta all'interno del genoma.

[18] «Basic Local Alignment Search Tool (Altschul et al., 1990 & 1997) is a sequence comparison algorithm optimized for speed used to search sequence databases for optimal local alignments to a query.» [6] «BLAST Glossary». [Online]. Disponibile su: https://www.ncbi.nlm.nih.gov/books/NBK62051

[19] Identificativo univoco associato a ciascuna proteina mappata nei database di riferimento di BLAST.

Infine, si intende registrare per ciascun *gene del genoma* la sua posizione relativa rispetto agli altri geni all'interno del profilo genetico sequenziato. Questa informazione è essenziale per valutare la prossimità tra i geni e per identificare eventuali pattern di distribuzione specifici all'interno del genoma di Legionella. In termini pratici, si suggerisce l'introduzione di una relazione auto-referenziale che coinvolga l'entità *gene del genoma*, al fine di stabilire un legame tra i geni identificati e la loro posizione, relativa[20] all'interno del genoma sequenziato. La cardinalità di tale relazione sarà definita come (0,1) a (0,1), indicando che ogni gene può essere associato a zero o un gene rispetto al quale è sequenziale nel profilo genetico. Questa configurazione tiene conto della limitata conoscenza attuale sui geni di Legionella, che potrebbe comportare l'assenza di associazioni per alcune aree del genoma. Si noti che la relazione di sequenzialità tra i geni è monodirezionale, ovvero è conservata, per ogni gene, solamente l'informazione relativa al predecessore nel profilo genetico. Tale scelta è dettata dal proposito di mantenere basso il costo computazionale per la gestione delle informazioni, evitando così il rischio di inconsistenza dovuto alla duplicazione dei dati. In questo modo si elude l'introduzione di vincoli di integrità aggiuntivi, preferendovi piuttosto l'aumento della complessità di un'eventuale query finalizzata a ottenere l'informazione nel senso opposto a quello definito dalla relazione. Si osserva che gli unici vincoli di integrità che si rendono necessari sono i seguenti: un gene del genoma non può essere associato a se stesso, né può essere associato a un altro gene se esistono geni noti che hanno posizione assoluta maggiore rispetto al gene con il quale si vuole stabilire la relazione di sequenzialità, ma minore rispetto al gene inserito.

[20]Definita in relazione alla prossimità ad altri geni all'interno del genoma sequenziato.

3.b. Diagramma E-R integrante le nuove esigenze

A seguito delle modifiche proposte, è realizzato il seguente diagramma E-R.

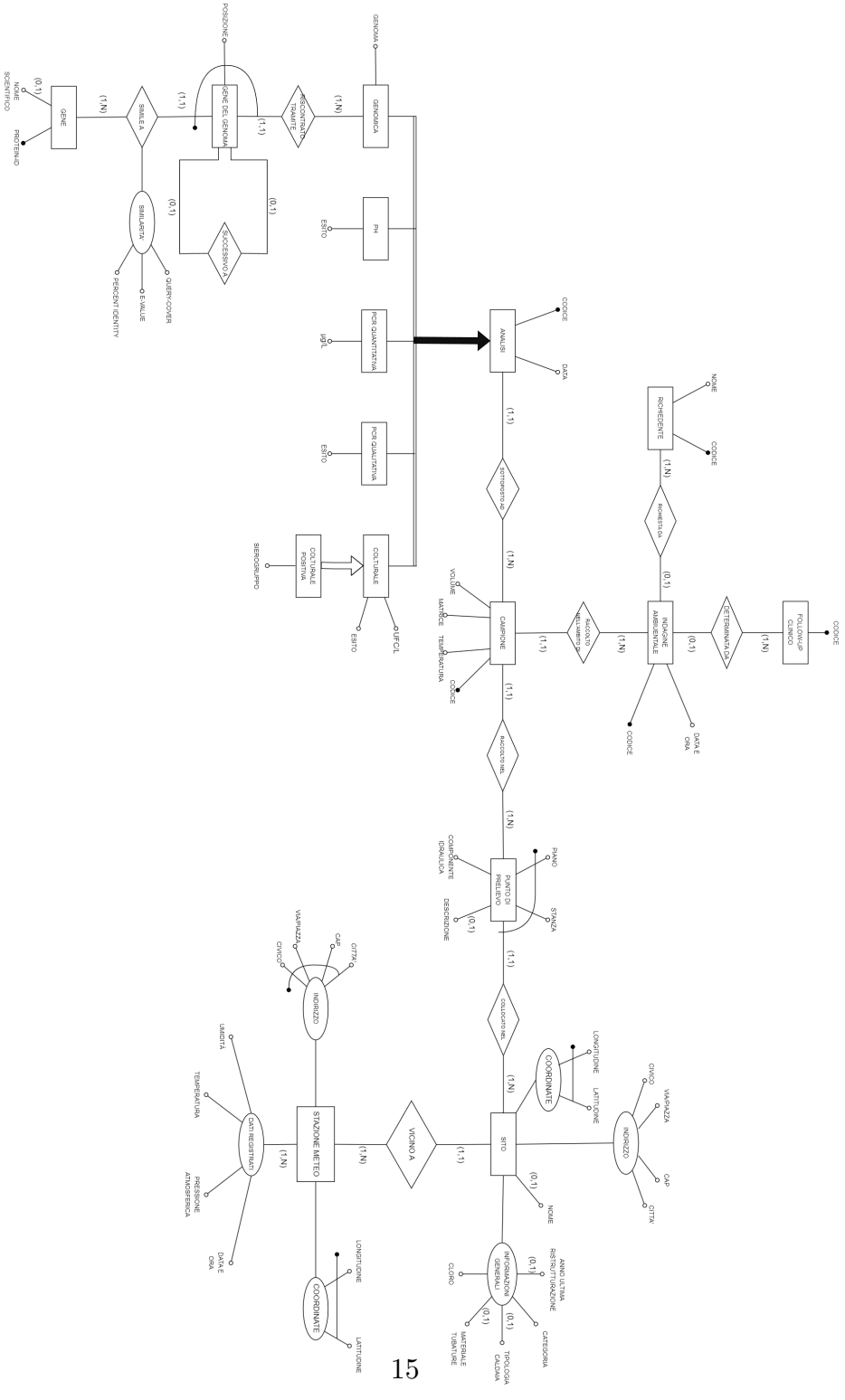

Figura 3: Diagramma ER

3.B.I. Note

Si noti come l'introduzione di nuove entità e relazioni, sebbene arricchisca il quadro di informazioni memorizzate nel database, comporta un forte aumento della complessità del sistema. Più specificamente, l'aggiunta delle entità coinvolte nella memorizzazione delle informazioni genomiche richiede maggiore attenzione, in quanto, per garantire la coerenza dei dati con le informazioni disponibili nei database di BLAST o degli altri strumenti che possono essere utilizzati per l'individuazione e la classificazione dei geni, è necessario aggiornare costantemente le istanze dell'entità *gene* con i dati più recenti.

In ultimo, si segnala che le principali operazioni eseguite sulla base di dati riguardano l'inserimento, la modifica e la cancellazione dei dati. Al contrario, le operazioni di interrogazione sono limitate a un numero ristretto di query, finalizzate a ottenere informazioni di tipo spaziale sui campioni, sulle analisi effettuate e sui risultati ottenuti oppure informazioni genetiche. Pertanto, si preferisce adottare una struttura facilmente manutenibile e ottimizzata per le operazioni fondamentali, che risulta già adeguata per l'esecuzione delle operazioni sopra menzionate, piuttosto che una struttura più complessa, progettata per ottimizzare le interrogazioni, ma che comporterebbe un costo maggiore per la gestione dei dati.

Le considerazioni relative ai vincoli di integrità sono posticipate al capitolo successivo, nel quale, terminata la fase di progettazione, sarà possibile ottenere una visione del tutto trasparente e definitiva delle entità coinvolte nel sistema dei relativi attributi e delle relazioni tra di esse.

4. Progettazione logica della base di dati

4.a. Ristrutturazione del modello concettuale: Semplificazione delle generalizzazioni e degli attributi composti

Ultimata la fase di progettazione concettuale della base di dati, è opportuno effettuare un'ultima revisione del modello al fine di elaborarne la struttura finale, priva di elementi discrezionali. In questa unità sono riportate le modifiche, congiuntamente alle motivazioni che le guidano, apportate allo schema E-R proposto al paragrafo 3.b, con l'obbiettivo di risolvere generalizzazioni, attributi composti e attributi multivalori presenti in figura.

In prima istanza, sono trattati gli aspetti riguardanti le entità coinvolte nelle relazioni di generalizzazione.

Per quanto concerne l'entità *analisi*, si propone di rimuovere, in tutto il suo complesso, la relazione di generalizzazione associando piuttosto le differenti tipologie di analisi ai campioni su cui sono eseguite. Ognuna delle analisi è identificata in modo esclusivo da un codice e caratterizzata sia della data di esecuzione, proprietà ereditata dall'entità soppressa *analisi*, sia dagli attributi caratterizzanti di ciascuna specializzazione. Inoltre, pur non essendo del tutto rigoroso dal punto di vista scientifico, si propone di riassumere le tabelle rappresentative delle analisi *PCR qualitativa* e *PCR quantitativa* in un'unica soluzione denominata *analisi PCR*, che conserva informazioni di entrambe le nature sui campioni analizzati. Questa semplificazione è ritenuta lecita in quanto i risultati prodotti dalle due analisi sono intrinsecamente correlati e possono essere memorizzati in modo più efficiente all'interno di un'unica entità. Infatti la PCR qualitativa, che rileva la presenza del DNA di Legionella, è, nell'ipotesi in cui restituisce un risultato positivo, seguita dala PCR quantitativa, la quale misura la concentrazione del batterio nel campione. La soluzione proposta consente di alleggerire la struttura del database e di semplificare le operazioni di inserimento e consultazione dei dati, senza introdurre perdite di informazioni

né comportare un'aumento della complessità del sistema inteso come l'introduzione di vincoli di integrità. Si noti che non tutte le analisi sono eseguite su tutti i campioni, ma, talvolta, solo su una parte di essi. Ad esempio, come suggerito dalle linee guida per la prevenzione ed il controllo della legionellosi, «poiché la q-PCR è effettivamente vantaggiosa per molteplici aspetti ma non ancora validata a livello internazionale, essa può, ad oggi, essere solo consigliata per una rapida analisi di numerosi campioni prelevati da siti probabilmente associati ad un caso o ancor più a un cluster di legionellosi, potendo in tempi brevi escludere i siti negativi ed identificare quelli positivi»[21]. In altre parole, le stesse linee guida suggeriscono di eseguire l'analisi colturale solo in caso di risultato positivo alla q-PCR, senza tuttavia stabilire una convenzione. Tale mancanza consente diverse interpretazioni e, pertanto, si è deciso di definire la cardinalità della relazione tra *campione* e le diverse *analisi* come "(0,1) a (1,1)", stabilendo che un campione può essere associato a zero o una sola analisi specifica, mentre ogni analisi è sempre associata a un campione. Questa scelta garantisce anche la retrocompatibilità del sistema. Infatti non essendo possibile associare a campioni già esistenti le analisi aggiunte successivamente, ovvero *analisi del pH* e *analisi genomica*, è opportuno che i campioni non siano obbligatoriamente associati a tutte le tipologie di analisi.

Per quanto concerne la specializzazione relativa all'*analisi colturale*, ovvero l'*analisi colturale positiva*, se ne suggerisce la sostituzione con un attributo denominato sierogruppo, che è proprio dell'entità *analisi colturale*. Tale modifica permette di conservare le informazioni relative al sierogruppo di Legionella identificato nel campione, senza introdurre una nuova entità e risparmiando dunque spazio. Si noti che questa modifica implica l'introduzione di un vincolo di integrità che assicuri che solamente le analisi colturali positive siano associate a un sierogruppo. Maggiori dettagli sui vincoli di integrità saranno forniti nel capitolo successivo.

Un ulteriore elemento di riflessione riguarda la risoluzione degli attributi composti.

In riferimento alla relazione *simile a* che coinvolge le entità *gene* e *gene del genoma*, si propone di scomporre l'attributo similarità nelle tre componenti che lo costituiscono: percent identity, query-cover e e-value. Inoltre, la cardinalità della relazione suggerisce di ricollocare questi attributi in modo tale che siano riferiti all'entità *gene del genoma*.

In merito all'attributo composto e multivalotre *dati registrati*, afferente all'entità *stazione meteorologica*, la soluzione più adeguata consiste nella sua sostituzione con una nuova tabella denominata *dati meterelogici* associata tramite una relazione del tipo 1 a N all'entità *stazione meteorologica*. Questa nuova

[21] [1], *Linee guida per la prevenzione ed il controllo della legionellosi*, p. 21

entità è debole rispetto alla *stazione meteorologica* e conserva le seguenti informazioni: la data di acquisizione di ogni set di dati, che costituisce parte della chiave primaria, la temperatura, l'umidità e la pressione atmosferica.

In conclusione, relativamente agli attributi indirizzo e coordinate, si ritiene opportuno adottare la medesima soluzione per tutte le apparizioni nello schema: l'attributo coordinata è sostituito dalla coppia di latitudine e longitudine, che costituiscono la chiave primaria delle entità cui sono riferite, ovvero *sito* e *stazione meteorologica*; l'attributo indirizzo è risolto in modo analogo, sostituendolo con i campi via, numero civico, CAP e città. Questa soluzione è dettata dall'intenzione di sfruttare le informazioni geografiche per condurre analisi mirate sulla diffusione della Legionella e per identificare eventuali cluster. In particolare, la soluzione proposta consente di semplificare notevolmente la struttura delle interrogazioni necessarie per ottenere tali informazioni.

Per esempio, l'implementazione di un'interrogazione per la ricerca di tutti i campioni positivi raccolti in una determinata data, in una specifica via di una certa città, potrebbe restituire risultati non corretti qualora l'indirizzo fosse memorizzato come un'unica stringa di testo, poiché non sarebbe possibile fare una distinzione tra le componenti via e città. Al contrario, la soluzione proposta consente di ottenere risultati corretti e coerenti con le aspettative.

4.b. Diagramma E-R finale

Il diagramma E-R finale, risultante dalla rielaborazione dello schema proposto al paragrafo 3.b, è presentato di seguito.

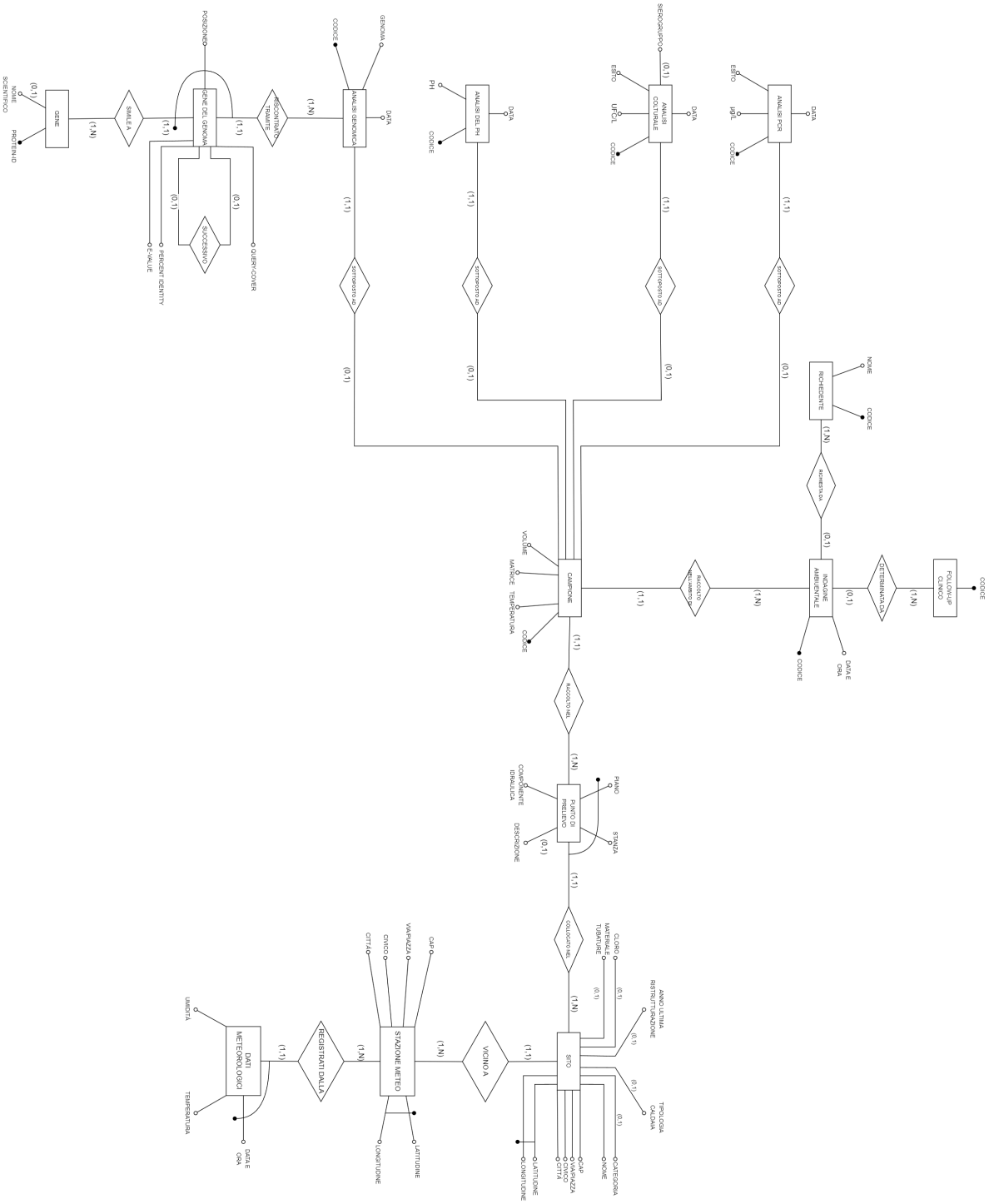

Figura 4: Diagramma ER

4.c. Illustrazione delle decisioni di progettazione nella trasformazione dal modello concettuale a quello logico

La trasposizione del modello concettuale in quello logico comporta la definizione delle tabelle, dei rispettivi campi, e quindi dei domini, delle chiavi primarie e delle modalità di associazione tra le tabelle. In questa sezione vengono presentate le scelte progettuali adottate.

La traduzione delle entità in tabelle è diretta e non comporta particolari difficoltà. Ogni entità, infatti, è rappresentata mediante una matrice in cui ogni attributo corrisponde a una colonna.

Per quanto riguarda le relazioni, è essenziale considerare tre tipologie fondamentali: le relazioni uno a uno, le relazioni uno a molti e le relazioni molti a molti. Le relazioni uno a uno sono le più complesse in quanto non è immediatamente chiaro quale entità debba essere scelta per mappare la relazione, ovvero in quale entità inserire la chiave esterna. Nel nostro contesto si presentano due situazioni principali: la relazione autoreferenziale tra i geni del genoma e la relazione tra i campioni e le analisi. Nel primo caso, la relazione è mappata sull'entità *gene del genoma* poiché, sebbene gli strumenti di analisi presentino alcune limitazioni, nella maggioranza dei genomi analizzati esiste un'effettiva sequenzialità tra i geni. Pertanto lo spazio sprecato a causa della mancanza di informazioni è limitato e dunque non giustifica l'introduzione di una nuova tabella che, pur limitando lo spazio utilizzato, porterebbe problemi di integrità e complessità.

Per quanto riguarda le relazioni tra campioni e analisi, si è deciso di mappare la relazione sull'entità *analisi*. Si osserva che la soluzione alternativa, ovvero quella di mappare la relazione sull'entità *campione*, potrebbe comportare perdite di spazio a causa della presenza di campioni non analizzati rispetto ad un esame specifico.

Le relazioni uno a molti, invece, sono più semplici da gestire, in quanto la chiave esterna è necessariamente inserita nell'entità che rappresenta il lato "uno" della relazione. Infine, le relazioni molti a molti sono gestite mediante l'introduzione di una tabella di associazione che contiene le chiavi esterne delle due entità coinvolte.

Per concludere, le entità deboli, come *dati meteorologici*, *punto di prelievo* e *gene del genoma*, la chiave primaria è composta dalla chiave primaria dell'entità forte, o delle entità forti, a cui sono associate e un attributo che ne identifica univocamente l'istanza all'interno dell'entità forte.

4.d. Schema relazionale

Sulla base delle considerazioni precedenti, si procede alla definizione dello schema relazionale, che rappresenta la struttura logica del database.

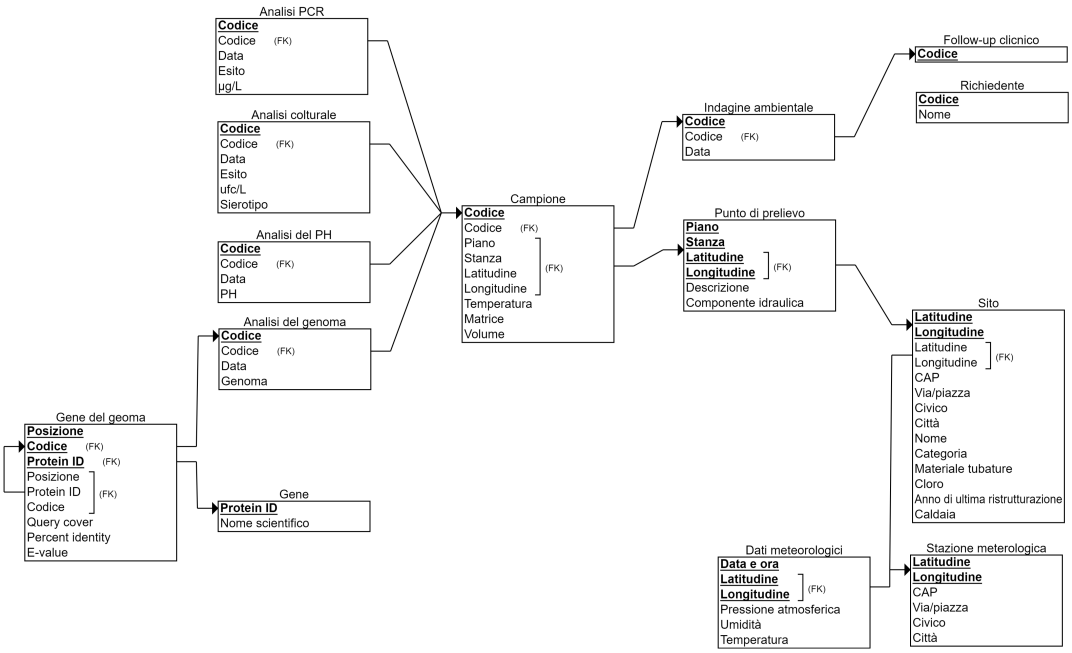

Figura 5: Schema relazionale

5. Progettazione fisica della base di dati

In questa sezione viene eseguita l'implementazione della base di dati, iniziando dalla definizione dei domini, ovvero l'insieme dei valori ammissibili per ciascuna colonna, e proseguendo con la creazione delle tabelle e delle funzioni che implementano i vincoli che assicurano l'integrità dei dati.

Il DBMS scelto per la realizzazione del database è PostgreSQL[22], un sistema di gestione di basi di dati relazionali rilasciato per la prima volta nel 1989. La scelta è motivata dalla flessibilità ed estendibilità del sistema, che consente di implementare vincoli di integrità complessi e di gestire grandi quantità di dati in modo efficiente.

5.a. Definizione dei domini

La maggior parte dei domini relativi alle colonne di ciascuna tabella è facilmente determinabile. Tuttavia, alcuni domini richiedono una definizione più dettagliata per garantire una corretta rappresentazione dei dati e facilitare l'esecuzione delle operazioni di interrogazione.

Primariamente, si considerino i domini relativi alla quantificazione della Legionella nei campioni, espressi in ufc/l e µg/l. Per entrambi è opportuno ridurre il dominio ai valori interi positivi, poiché non ha senso esprimere la presenza di Legionella con valori negativi. Un caso analogo si presenta per i valori che misurano il volume di un campione, l'umidità e la pressione atmosferica, per i quali si propone di definire il dominio come un numero decimale positivo, in quanto non ha senso avere valori negativi per queste grandezze fisiche.

```
-- Dominio per il valore intero non negativo
CREATE DOMAIN INT_POS AS INTEGER
    CONSTRAINT valore_non_negativo CHECK (VALUE >= 0);

-- Dominio per il valore reale non negativo
```

[22] [7] «PostgreSQL 16.4 documentation». [Online]. Disponibile su: https://www.postgresql.org/docs/current

```
CREATE DOMAIN FLOAT_POS AS FLOAT
    CONSTRAINT valore_non_negativo CHECK (VALUE >= 0);
```

In secondo luogo si analizzi il dominio relativo al parametro di misurazione del pH. Per il fatto che il range di valori ammissibili per il pH è compreso tra 0 e 14, si propone di definire il dominio del pH come un numero decimale rientrante in questo intervallo.

```
CREATE DOMAIN PH AS FLOAT
    CONSTRAINT ph_range CHECK (VALUE >= 0 AND VALUE <= 14);
```

Un ulteriore aspetto da considerare riguarda le colonne categoria e matrice relative rispettivamente alle tabelle *sito* e *campione*. Per quanto riguarda la colonna categoria, si propone di limitare il dominio a pochi vocaboli appartenenti ad un ristretto insieme semantico, come ad esempio "ospedaliero", "termale", "alberghiero", "pubblico" e "privato". Si precisa che il valore "pubblico" include tutti quegli edifici, non afferenti alle categorie specificate, destinati alla fruizione da parte di un'ampia e variegata utenza. Analogamente, per la colonna matrice, si propone di fissare un dominio che comprenda solo valori appartenenti a un insieme finito di matrici, come ad esempio "acqua", "aria" e "biofilm" e "sedimento".

```
-- Tipo enum per la categoria di un sito
CREATE TYPE CATEGORIA AS
    ENUM ('ospedaliero', 'termale', 'alberghiero', 'pubblico', 'privato');

-- Tipo enum per la matrice di un campione
CREATE TYPE MATRICE AS
    ENUM ('acqua',  'aria', 'biofilm', 'sedimento');
```

La colonna CAP delle tabelle *sito* e *stazione meteorologica* rappresenta un aggiuntivo aspetto notevole. Poiché il CAP è un codice numerico formato da cinque cifre, si suggerisce di definire un dominio di tipo integer che accetti esclusivamente valori numerici di tale lunghezza.

```
CREATE DOMAIN CAP AS INTEGER
    CONSTRAINT cap_range CHECK (VALUE >= 10000 AND VALUE <= 99999);
```

Inoltre, si propone di restringere l'intervallo dei valori ammessi per le colonne *percent-identity*, *query-cover* e *e-value* dell'entità *gene del genoma*. In termini pratici, si suggerisce di definire un dominio di tipo float compresi tra 0 e 100 per le colonne *percent-identity* e *query-cover*, in quanto rappresentano percentuali di similarità tra i geni noti e quelli individuati tramite l'analisi. Per quanto concerne l'*e-value*, invece, si propone di utilizzare il dominio FLOAT_POS definito in precedenza, in quanto si vuole rappresentare un valore numerico positivo.

```sql
CREATE DOMAIN PERCENT AS FLOAT
    CONSTRAINT percent_range CHECK (VALUE >= 0 AND VALUE <= 100);
```

Infine, per quanto riguarda gli attributi relativi alle coordinate geografiche, ovvero latitudine e longitudine, il dominio deve essere limitato a valori compresi tra -90 e 90 per la latitudine e tra -180 e 180 per la longitudine.

```sql
-- Dominio per latitudine
CREATE DOMAIN LATITUDINE AS REAL
    CONSTRAINT latitudine_range CHECK (VALUE >= -90 AND VALUE <= 90);

-- Dominio per longitudine
CREATE DOMAIN LONGITUDINE AS REAL
    CONSTRAINT longitudine_range CHECK (VALUE >= -180 AND VALUE <= 180);
```

5.A.I. Note

Sulla base delle osservazioni riportate in diversi articoli scientifici riguardanti lo studio degli aspetti genetici del batterio, come ad esempio *Draft genome sequences from 127 Legionella spp. strains isolated in water systems linked to legionellosis outbreaks*[23], è emerso che la lun ghezza media del genoma di Legionella pneumophila è di circa 3.500.000 coppie di basi, con una significativa variabilità tra i genomi sequenziati.

In considerazione alle dimensioni dell'oggetto, si propone di assegnare un dominio di tipo text[24]. Questo tipo di dato consente la memorizzazione di stringhe di lunghezza arbitraria, risultando particolarmente adatto per la conservazione di sequenze di DNA.

Si evidenzia inoltre che, in termini di occupazione della memoria, la politica TOAST[25] tipica di PostgreSQL consente una gestione efficiente anche per attributi di grandi dimensioni, allocando i dati in pagine separate e comprimendoli per ridurre lo spazio complessivo occupato.

[23] [8] A. Colautti *et al.*, «Draft genome sequences from 127 Legionella spp. strains isolated in water systems linked to legionellosis outbreaks», *Microbiol Resour Announc*, vol. 13, fasc. 6, 2024, doi: 10.1128/mra.01154-23.

[24] https://www.postgresql.org/docs/current/datatype-character.html

[25] https://www.postgresql.org/docs/current/storage-toast.html

5.b. Creazione delle tabelle

Il codice per la creazione delle tabelle è banalmente ottenuto dal modello relazionale. Tuttavia, merita particolare attenzione la gestione dei vincoli di chiave esterna. In particolare, è necessario considerare il comportamento delle chiavi esterne nei casi di eliminazione o aggiornamento di una riga a cui queste fanno riferimento. In questo ambito vi sono tre principali opzioni, ovvero l'impedimento dell'operazione (RESTRICT), che comporta il rifiuto dell'operazione stessa, l'azione di cascata (CASCADE), che comporta l'aggiornamento o la cancellazione delle righe collegate alla riga interessata, e l'assegnazione di un valore nullo (SET NULL), che imposta il valore nullo nelle righe che fanno riferimento alla riga eliminata o modificata.

Per ragioni di spazio vengono forniti alcuni esempi di creazione delle tabelle, mentre il codice completo è riportato in appendice.

Analisi

Un aspetto rilevante riguarda la cancellazione di un campione. In generale, si ritiene opportuno di eliminare i dati associate al campione, poiché perderebbero di significato in sua assenza. Tuttavia, si propone di impedire l'operazione di cancellazione qualora il campione sia associato ad un'analisi del genoma. Tale decisione è finalizzata a interrompere la catena di eliminazione che coinvolgerebbe tutte le informazioni relative ai dati genomici osservati, al fine di evitare l'eliminazione accidentale di una grande quantità di dati. Si osserva che, per eliminare un campione, sarà sufficiente rimuovere preventivamente l'eventuale analisi genomica associata, dopodiché sarà possibile procedere con la cancellazione del campione stesso. A titolo di esempio si riportano le tabelle *analisi PCR* e *analisi del genoma*

```sql
-- Analisi PCR
CREATE TABLE Analisi_PCR (
    codice CHAR(6) NOT NULL,
    codice_campione CHAR(6) NOT NULL,
    data_ora DATE NOT NULL,
    esito BOOLEAN NOT NULL,
    µg_l INT_POS NOT NULL,

    PRIMARY KEY (codice),

    FOREIGN KEY (codice_campione) REFERENCES Campione(codice)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

-- Analisi genomica
```

```
CREATE TABLE Analisi_genomica (
    codice CHAR(6) NOT NULL,
    codice_campione CHAR(6) NOT NULL,
    data_ora DATE NOT NULL,
    genoma TEXT NOT NULL,

    PRIMARY KEY (codice),

    FOREIGN KEY (codice_campione) REFERENCES Campione(codice)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);
```

Gene del genoma

Un secondo caso di particolare rilevanza riguarda la tabella *gene del genoma*. Per quanto concerne la relazione con i geni noti di Legionella, si propone di propagare l'aggiornamento della chiave protein-ID a cascata e di impedire la cancellazione dei geni noti per i quali esistano geni del genoma associati. Relativamente alla relazione con i genomi di Legionella sequenziati, invece, è opportuno eseguire sia le operazioni di aggiornamento che di cancellazione a cascata. Tale comportamento è motivato dal fatto che, in caso di modifica di un genoma, è necessario aggiornare i geni del genoma associati; mentre, in caso di cancellazione del genoma, risulta opportuno eliminare anche i geni del genoma collegati, per evitare inconsistenze dovute alla presenza di dati orfani. Infine, riguardo alla relazione di sequenzialità tra i geni del genoma, la soluzione è immediatamente determinata dalla cardinalità della relazione: sostituire i dati registrati con il valore NULL.

```
CREATE TABLE Gene_del_genoma (
    posizione INTEGER NOT NULL,
    codice_genoma CHAR(6) NOT NULL,
    protein_ID CHAR(6) NOT NULL,
    posizione_predecessore INTEGER,
    codice_genoma_predecessore CHAR(6),
    protein_ID_predecessore CHAR(6),
    query_cover PERCENT NOT NULL,
    percent_identity PERCENT NOT NULL,
    e_value FLOAT_POS NOT NULL,

    PRIMARY KEY (posizione, codice_genoma, protein_ID),

    FOREIGN KEY (codice_genoma) REFERENCES Analisi_genomica(codice)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (protein_ID) REFERENCES Gene(protein_ID)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    FOREIGN KEY (posizione_predecessore, codice_genoma_predecessore,
protein_ID_predecessore)
```

```
        REFERENCES Gene_genoma(posizione, codice_genoma, protein_ID)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);
```

Sito

In ultimo, si riporta una nota riguardante la tabella sito. Poiché si desidera collegare ciascun sito alla stazione meteorologica più vicina, le operazioni di aggiornamento e cancellazione delle stazioni sono risolte mediante l'applicazione di un trigger, che associa automaticamente tutti i siti legati alla stazione interessata alla stazione meteorologica più vicina. Di conseguenza, non è necessario definire un vincolo di chiave esterna per tale tabella. Una trattazione più approfondita sarà fornita nei capitoli successuvi.]

Per ragioni di efficienza è consigliabile definire un indice spaziale per le colonne latitudine e longitudine delle tabelle *sito* e *stazione meteorologica*. Questo indice consente di migliorare le prestazioni delle interrogazioni che utilizzano operazioni di ricerca basate sulla distanza tra due punti sulla superficie terrestre. In termini implementativi, si introduce un campo di tipo geometry[26] del pacchetto PostGis[27], corrispondente alle coordinate geografiche. Successivamente, si definisce un indice spaziale su questo campo che permette di ottimizzare le operazioni di ricerca spaziale.

```
CREATE TABLE Sito (
    latitudine LATITUDINE NOT NULL,
    longitudine LONGITUDINE NOT NULL,
    latitudine_stazione LATITUDINE NOT NULL,
    longitudine_stazione LONGITUDINE NOT NULL,
    CAP CAP NOT NULL,
    via_piazza VARCHAR(25) NOT NULL,
    civico INTEGER NOT NULL,
    città VARCHAR(25) NOT NULL,
    nome VARCHAR(25),
    categoria CATEGORIA NOT NULL,
    materiale_tubature VARCHAR(25),
    cloro BOOLEAN NOT NULL,
    anno_ultima_ristrutturazione DATE,
    caldaia VARCHAR(25),
    geom GEOMETRY(Point, 4326) NOT NULL,


    PRIMARY KEY (latitudine, longitudine),

    FOREIGN KEY (latitudine_stazione, longitudine_stazione)
```

[26] https://postgis.net/docs/geometry.html

[27] [9] «PostGIS 3.5.0Dev Manual». [Online]. Disponibile su: https://postgis.net/docs/manual-dev/en

```
        REFERENCES Stazione_meterologica(latitudine, longitudine)
);
```

Si noti che che il codice 4326 è relativo al sistema di riferimento spaziale WGS 84, che rappresenta un sistema di coordinate geografiche utilizzato comunemente per la rappresentazione di dati geografici sulla superficie terrestre.

Inoltre, l'operazione di aggiunta di una nuova *stazione meteorologica* o *sito* deve essere strutturata in modo tale da garantire la coerenza delle informazioni relative a latitudine e longitudine e il punto geografico associato. In tal senso è opportuno ridefinire le funzioni di inserimento per le tabelle coinvolte, piuttosto che introdurre vincoli di integrità che appesentirebbero solamente il codice senza apportare miglioramenti in termini prestazionali. Al contrario, le funzioni di inserimento specializzate garantiscono che i dati vengano gestiti correttamente sin dal momento dell'inserimento a costo di un leggero aumento della complessità del codice. La definizione di tali funzioni è riportata nella sezione 5.e.

5.c. Definizione dei vincoli

A questo punto si dispone di una visione completa e definitiva della struttura del database, che rende possibile analizzare le criticità non risolte dallo schema attuale. In questa sezione sono presentati i vincoli di integrità necessari per garantire la consistenza dei dati all'interno del database, insieme alle motivazioni che ne determinano l'introduzione.

5.c.I. Gestione della chiave esterna relativa alla tabella *Sito*

In questo paragrafo si affrontano le problematiche relative alla cancellazione e all'aggiornamento di una stazione meteorologica, come accennato nel capitolo precedente. Poiché si desidera associare ciascun sito alla stazione meteorologica più vicina, si propone di implementare un trigger che, in caso di cancellazione o aggiornamento di una stazione meteorologica, assegni automaticamente a tutti i siti precedentemente collegati alla stazione interessata la stazione meteorologica più vicina. Questa soluzione evita l'utilizzo di un vincolo RESTRICT, il quale renderebbe più complessa la gestione delle operazioni di cancellazione e aggiornamento.

In dettaglio, il trigger di aggiornamento si occupa di aggiornare la stazione meteorologica associata a ciascun sito, sostituendola con quella più vicina in seguito alla modifica delle coordinate o all'inserimento di una stazione meteorologica. Il trigger di cancellazione, invece, impedisce l'eliminazione totale delle stazioni meteorologiche, assicurando che almeno una stazione meteorologica sia

associata a ciascun sito. Se la cancellazione è possibile, il trigger aggiorna le coordinate riferite alle stazioni meteorologiche dei siti coinvolti, sostituendole con quelle degli osservatori più vicini. Per garantire precisione ed efficienza, l'implementazione di questi trigger si avvale della funzione ST_Distance[28] del pacchetto PostGis che calcola la distanza tra due punti sulla superficie terrestre. Inoltre, si utilizza la funzione e ST_DWithin[29], che opera sugli indici spaziali, per determinare se due punti si trovano entro una certa distanza l'uno dall'altro e ridurre il numero di confronti necessari.

Di seguito è presentato solo il codice relativo alla definizione del trigger per le operazioni di aggiornamento e inserimento di una nuovo centro meteorologico. Il codice per la gestione della cancellazione, simile a quello mostrato, è riportato in appendice.

```
CREATE OR REPLACE FUNCTION update_stazione_meteorologica_on_delete()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    -- Verifica se ci sono più di una stazione meteorologica
    IF (SELECT COUNT(*) FROM Stazione_meteorologica) = 1 THEN
        RAISE EXCEPTION 'Non è possibile eliminare tutte le stazioni
meteorologiche.';
    END IF;

    -- Trova la stazione meteorologica più vicina per ciascun sito che aveva come
più vicina la stazione cancellata
    UPDATE Sito
    SET latitudine_stazione_meteorologica = stazione.latitudine,
        longitudine_stazione_meteorologica = stazione.longitudine
    FROM (
        SELECT sito.latitudine AS sito_latitudine, sito.longitudine AS
sito_longitudine,
               stazione.latitudine, stazione.longitudine,
               ST_Distance(sito.geom, stazione.geom) AS distance,
               ROW_NUMBER() OVER (PARTITION BY sito.latitudine, sito.longitudine
ORDER BY ST_Distance(sito.geom, stazione.geom)) AS rn

        FROM Sito sito
        JOIN Stazione_meteorologica stazione
        ON ST_DWithin(
            geography(sito.geom),
            geography(stazione.geom),
            100000 -- 100 km
        )
        WHERE (stazione.latitudine != OLD.latitudine OR stazione.longitudine !=
OLD.longitudine)

    ) AS stazione
    WHERE Sito.latitudine = stazione.sito_latitudine
      AND Sito.longitudine = stazione.sito_longitudine
```

[28]https://postgis.net/docs/ST_Distance.html
[29]https://postgis.net/docs/ST_DWithin.html

```
      AND rn = 1;

    RETURN NULL;
END;
$$;

CREATE TRIGGER update_stazione_meteorologica_on_delete_trigger
BEFORE DELETE ON Stazione_meteorologica
FOR EACH ROW
EXECUTE FUNCTION update_stazione_meteorologica_on_delete();
```

5.c.II. Vincoli relativi ai dati

Analisi

Per quanto riguarda le analisi, si propone di introdurre vincoli che assicurino la coerenza dei dati registrati.

Si consideri, in primo luogo, l'entità analisi colturale. La corretta formazione dei dati registrati a seguito di ciascuna analisi richiede l'applicazione dei seguenti vincoli di integrità relativi ai casi di positività del campione: ad ogni campione negativo non deve essere associato alcun sierogruppo; ad ogni campione positivo deve essere associato un valore di unità formanti colonia per litro (ufc/l) maggiore di zero, mentre ad ogni campione negativo deve essere associato il valore pari a zero.

Per quanto riguarda l'entità analisi PCR, invece, si applica il seguente vincolo: ad ogni campione positivo deve essere associato un valore di microgrammi per litro (µg/l) maggiore di zero, mentre ad ogni campione negativo deve essere associato il valore pari a zero.

Per ragioni di spazio, si riporta esclusivamente il codice relativo ai vincoli riguardanti l'*analisi colturale*. Il codice che implementa la funzione di controllo per la tabella *analisi PCR* è analogo, con l'eccezione che non prevede condizioni relative al sierotipo, ed è consultabile in appendice.

```
CREATE OR REPLACE FUNCTION check_esito_Colturale()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    IF NEW.esito = TRUE AND NEW.ufc_l = 0 THEN
        RAISE EXCEPTION 'Il valore ufc/l deve essere maggiore di 0 quando l''esito è
positivo.';
    ELSIF NEW.esito = FALSE AND NEW.ufc_l > 0 THEN
        RAISE EXCEPTION 'Il valore ufc/l deve essere 0 quando l''esito è negativo.';
    ELSIF NEW.esito = FALSE AND NEW.sierotipo IS NOT NULL THEN
        RAISE EXCEPTION 'Il sierotipo non può essere specificato quando l''esito è
negativo.';
    END IF;
    RETURN NEW;
END;
$$;
```

```
CREATE TRIGGER check_esito_Colturale
BEFORE INSERT OR UPDATE ON Analisi_culturale
FOR EACH ROW
EXECUTE FUNCTION check_esito_Colturale();
```

Campioni

Un ulteriore accorgimento deve essere impiegato nel caso dei campioni. Infatti, come già accenntato, poiché un'indagine ambientale è una collezione di campioni raccolti in una stessa data, in un sito specifico, è necessario garantire che tutti i campioni associati a un'indagine siano prelevati nello stesso sito. In termini pratici si propone di introdurre un trigger che, in caso di inserimento o aggiornamento di un campione, verifichi che il sito associato al campione sia lo stesso per tutti i campioni associati all'indagine a cui il campione appartiene.

```
CREATE OR REPLACE FUNCTION check_campione_indagine()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM Indagine_ambientale JOIN Campione ON codice = codice_indagine
        WHERE codice = NEW.codice_indagine
        AND (latitudine_sito != NEW.latitudine_sito OR longitudine_sito !=
NEW.longitudine_sito)
    ) THEN
        RAISE EXCEPTION 'I campioni raccolti nell''ambito di una stessa indagine
devono essere prelevati nel medesimo sito.';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER check_campione_indagine
BEFORE INSERT OR UPDATE ON Campione
FOR EACH ROW
EXECUTE FUNCTION check_campione_indagine();
```

Geni

Infine, per quanto riguarda l'entità *gene del genoma* è necessario considerare attentamente la relazione di sequenzialità tra i genii è consigliabile l'introduzione di vincoli che garantiscano che a un gene di un genoma non possa essere associato un gene di un genoma diverso, né se stesso, né possa essere associato a un altro gene dello stesso genoma, qualora esistano altri geni con posizione assoluta maggiore rispetto a quello con cui si intende stabilire la relazione, ma minore rispetto al gene considerato. Questo vincolo è necessario per garantire

la corretta rappresentazione della sequenza genetica di Legionella e prevenire eventuali situazioni di inconsistenza.

```
CREATE OR REPLACE FUNCTION check_genoma()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    IF NEW.codice_genoma != NEW.codice_genoma_predecessore THEN
        RAISE EXCEPTION 'Non è possibile associare un gene a un genoma differente.';
    END IF;
    IF NEW.posizione = NEW.posizione_predecessore AND NEW.codice_genoma =
NEW.codice_genoma_predecessore AND NEW.protein_ID = NEW.protein_ID_predecessore THEN
        RAISE EXCEPTION 'Non è possibile associare un gene a se stesso.';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER check_genoma
BEFORE INSERT OR UPDATE ON Gene_genoma
FOR EACH ROW
EXECUTE FUNCTION check_genoma();

CREATE OR REPLACE FUNCTION check_predecessore()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM Gene_genoma
        WHERE codice_genoma = NEW.codice_genoma_predecessore AND posizione >
NEW.posizione_predecessore AND posizione < NEW.posizione
    ) THEN
        RAISE EXCEPTION 'Il gene predecessore non è corretto: esiste un gene con
posizione compresa tra la posizione del gene e quella del gene predecessore.';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER check_predecessore
BEFORE INSERT OR UPDATE ON Gene_genoma
FOR EACH ROW
EXECUTE FUNCTION check_predecessore();
```

5.d. Definizione dei trigger per la gestione delle istanze divenute isolate a seguito di operazioni di cancellazione o aggiornamento

Prima di procedere con l'implementazione di alcune operazioni notevoli, è essenziale definire opportuni trigger che consentano di gestire, in modo conforme a una politica ben definita, le operazioni di cancellazione e aggiornamento relative a determinate tabelle. Questi trigger servono a prevenire situazioni in cui alcune entry risultino "prive" di significato a seguito di modifiche o cancellazioni di altre entry a cui erano precedentemente collegate.

Indagine ambientale

Un primo caso riguarda la gestione delle indagini ambientali, definite come una collezione di campioni raccolti in una data specifica e in un sito particolare. A seguito della modifica o della cancellazione delle entry relative ai campioni afferenti a un'indagine, questa risulterebbe priva di collegamenti con alcun campione, ovvero, di fatto, priva di significato. Per evitare tale situazione, si propone di definire un trigger che, in caso di cancellazione di tutti i campioni associati a un'indagine, elimini automaticamente anche l'indagine stessa.

Richiedente e Follow-up Clinico

Un altro aspetto fondamentale riguarda la gestione delle entry relative ai richiedenti e ai follow-up clinici. È opportuno che, qualora tutte le indagini associate a un determinato richiedente o follow-up clinico vengano cancellate, anche il richiedente o il follow-up clinico vengano rimossi. Anche in questo caso, l'introduzione di un trigger automatizza il processo di cancellazione.

Per quanto concerne le altre tabelle del database, si ritiene non necessario implementare trigger per gestire le operazioni di cancellazione o aggiornamento. Queste tabelle, infatti, contengono entità sostanzialmente stabili che mantengono la loro validità anche in assenza di entry collegate, poiché potrebbero essere riutilizzate in futuro.

Si riporta solamente il codice relativo alla definizione del trigger per la gestione delle operazioni di cancellazione di un'indagine ambientale. Altri trigger, analoghi a quello mostrato, sono stati implementati per le tabelle *Richiedente* e *FollowUp_clinico* e sono consultabili in appendice.

```
CREATE OR REPLACE FUNCTION delete_indagine()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
```

```
    IF NOT EXISTS (
        SELECT 1
        FROM Campione
        WHERE codice_indagine = OLD.codice
    ) THEN
        DELETE FROM Indagine_ambientale
        WHERE codice = OLD.codice;
    END IF;
END;
$$;

CREATE TRIGGER delete_indagine
AFTER DELETE ON Campione
FOR EACH ROW
EXECUTE FUNCTION delete_indagine();
```

5.e. Operazioni di inserimento e aggiornamento

L'ultima sezione di questo elaborato è dedicata alla ridefinizione di alcune operazioni di inserimento o aggiornamento, con l'obbiettivo di aggevolare la fruizione della base di dati per gli utenti.

5.e.i. Inserimento e aggiornamento di una nuova stazione meteorologica

Una prima operazione di rilievo riguarda l'inserimento di una nuova stazione meteo. Per questa operzione è opportuno definire una funzione che, a partire dalle coordinate geografiche inserite dall'utente, calcoli il punto geografico associato. Questa soluzione è particolarmente utile per garantire la coerenza dei dati: infatti, evita che le coordinate geografiche e il punto geografico associato siano inseriti in modo non corrispondente e semplifica l'operazione per l'utente. Si sottolinea che l'operazione di aggiornamento è del tutto analoga a quella di inserimento; pertanto, di seguito, si riporta solamente il codice per l'inserimento di una nuova stazione meteo, mentre quello relativo all'aggiornamento è consultabile in appendice.

```
CREATE OR REPLACE FUNCTION insert_stazione_meteorologica(
    latitudine FLOAT,
    longitudine FLOAT,
    via_piazza VARCHAR(25),
    numero_civico INTEGER,
    CAP INTEGER,
    città VARCHAR(25)
)
RETURNS VOID LANGUAGE plpgsql AS
$$
```

```
DECLARE
    point GEOMETRY;
BEGIN
    -- Crea un punto geometrico a partire da latitudine e longitudine
    point := ST_SetSRID(ST_MakePoint(longitudine, latitudine), 4326);

    -- Inserisci i valori nella tabella Stazione_meteorologica
    INSERT INTO Stazione_meteorologica (latitudine, longitudine, via_piazza, civico,
CAP, città, geom)
    VALUES (latitudine, longitudine, via_piazza, numero_civico, CAP, città, point);

    RETURN;
END;
$$;
```

5.e.II. Inserimento e aggiornamento di un nuovo sito

Un secondo aspetto di rilievo riguarda l'inserimento o l'aggiornamento di sito. Anche in questo caso, è opportuno definire una funzione che, a partire dalle coordinate geografiche inserite dall'utente, calcoli il punto geografico associato. Inoltre, è necessario referenziare automaticamente il sito alla stazione meteorologica più vicina. Tale collegamento al momento della creazione di una nuova entry nella tabella sito è garantita dall'impiego, nella funzione di inserimento, di una user define function quasi del tutto analoga a quelle viste nel paragrafo 5.c.I che calcola la distanza tra il sito e le stazioni meteorologiche presenti nel database e associa il sito alla stazione più vicina. Di seguito si riporta il codice per l'operazione di aggiornamento. Per la consoltazione del codice relativo all'inserimento si rimanda all'appendice.

```
CREATE OR REPLACE FUNCTION update_sito(
    old_latitudine LATITUDINE,
    old_longitudine LONGITUDINE,
    new_latitudine LATITUDINE,
    new_longitudine LONGITUDINE,
    new_via_piazza VARCHAR(25),
    new_numero_civico INTEGER,
    new_CAP CAP,
    new_città VARCHAR(25),
    new_nome VARCHAR(25),
    new_categoria CATEGORIA,
    new_materiale_tubature VARCHAR(25),
    new_cloro BOOLEAN,
    new_anno_ultima_ristrutturazione DATE,
    new_caldaia VARCHAR(25)
)
RETURNS VOID LANGUAGE plpgsql AS
$$
DECLARE
    point GEOMETRY;
    new_latitudine_stazione_meteorologica LATITUDINE;
```

```
    new_longitudine_stazione_meteorologica LONGITUDINE;
BEGIN
    point := ST_SetSRID(ST_MakePoint(new_longitudine, new_latitudine), 4326);

    SELECT stazione.latitudine, stazione.longitudine
    INTO new_latitudine_stazione_meteorologica,
new_longitudine_stazione_meteorologica
    FROM Stazione_meteorologica stazione
    WHERE ST_DWithin(
        geography(point),
        geography(stazione.geom),
        100000 -- 100 km
    )
    ORDER BY ST_Distance(point, stazione.geom)
    LIMIT 1;

    UPDATE Sito
    SET latitudine = new_latitudine,
        longitudine = new_longitudine,
        latitudine_stazione_meteorologica = new_latitudine_stazione_meteorologica,
        longitudine_stazione_meteorologica = new_longitudine_stazione_meteorologica,
        via_piazza = new_via_piazza,
        civico = new_numero_civico,
        CAP = new_CAP,
        città = new_città,
        nome = new_nome,
        categoria = new_categoria,
        materiale_tubature = new_materiale_tubature,
        cloro = new_cloro,
        anno_ultima_ristrutturazione = new_anno_ultima_ristrutturazione,
        caldaia = new_caldaia,
        geom = point
    WHERE latitudine = old_latitudine
      AND longitudine = old_longitudine;

    RETURN;
END;
$$;
```

6. Conclusioni

Questa tesi ha illustrato l'intero processo di sviluppo di un database relazionale per la gestione dei dati relativi al monitoraggio del batterio Legionella raccolti nell'ambito delle indagini svolte dall'ARPA FVG. In particolare, nei primi capitoli è stata condotta un'analisi approfondita di un progetto pre-esistente, evidenziandone i punti di forza e le criticità, e proponendo un nuovo modello concettuale in grado di risolvere le problematiche riscontrate e integrare nuove funzionalità richieste dagli stakeholders. Successivamente, nel capitolo 4 è stato definito lo schema logico del database, seguendo i principi del modello relazionale descritti nel volume "Database Systems[30]". Infine, nella seizione 5 è stato presentato il codice SQL per la creazione delle tabelle, la definizione degli indici spaziali e dei vincoli di integrità, oltre che per la gestione di alcune operazioni di inserimento e aggiornamento, corredato delle motivazioni che ne hanno guidato l'implementazione.

Il prodotto di questo elaborato costituisce una risorsa iniziale per l'ARPA FVG per il monitoraggio della Legionella nella nostra regione e consente di costruire analisi per monitorare e prevedere la diffusione del batterio permettendo di definire le misure di prevenzione e controllo maggiormente efficaci. Tuttavia, il lavoro svolto lascia spazio a ulteriori sviluppi e miglioramenti, come, per esempio, l'introduzione di applicativi software per la raccolta e l'inserimento dei dati, attualmente eseguiti manualmente attraverso schede cartacee o fogli elettronici senza l'impiego di procedure standardizzate, e per la consultazione del sistema, che potrebbe risultare complessa per utenti non esperti.

[30] [5] «*Database Systems: Concepts, Languages & Architectures*»

Appendice

Creazione delle tabelle e la definizione degli indici spaziali

```
-- DEFINIZIONE DELLE TABELLE

-- Stazione meteorologica
CREATE TABLE Stazione_meteorologica (
    latitudine LATITUDINE NOT NULL,
    longitudine LONGITUDINE NOT NULL,
    via_piazza VARCHAR(25) NOT NULL,
    civico INTEGER NOT NULL,
    CAP CAP NOT NULL,
    città VARCHAR(25) NOT NULL,
    geom GEOMETRY(Point, 4326) NOT NULL,

    PRIMARY KEY (latitudine, longitudine)
);

-- Dati meteorologici
CREATE TABLE Dati_meteorologici (
    data_ora TIMESTAMP NOT NULL,
    latitudine_stazione LATITUDINE NOT NULL,
    longitudine_stazione LONGITUDINE NOT NULL,
    temperatura FLOAT NOT NULL,
    umidità FLOAT_POS NOT NULL,
    pressione_atmosferica FLOAT NOT NULL,

    PRIMARY KEY (data_ora, latitudine_stazione, longitudine_stazione),

    FOREIGN KEY (latitudine_stazione, longitudine_stazione) REFERENCES
Stazione_meteorologica(latitudine, longitudine)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE Sito (
    latitudine LATITUDINE NOT NULL,
    longitudine LONGITUDINE NOT NULL,
    latitudine_stazione_meteorologica LATITUDINE NOT NULL,
    longitudine_stazione_meteorologica LONGITUDINE NOT NULL,
```

```
    CAP CAP NOT NULL,
    via_piazza VARCHAR(25) NOT NULL,
    civico INTEGER NOT NULL,
    città VARCHAR(25) NOT NULL,
    nome VARCHAR(25),
    categoria CATEGORIA NOT NULL,
    materiale_tubature VARCHAR(25),
    cloro BOOLEAN NOT NULL,
    anno_ultima_ristrutturazione DATE,
    caldaia VARCHAR(25),
    geom GEOMETRY(Point, 4326) NOT NULL,

    PRIMARY KEY (latitudine, longitudine),

    FOREIGN KEY (latitudine_stazione_meteorologica,
longitudine_stazione_meteorologica) REFERENCES Stazione_meteorologica(latitudine,
longitudine)
);

-- Punto di prelievo
CREATE TABLE Punto_di_prelievo (
    piano INTEGER NOT NULL,
    stanza VARCHAR(15) NOT NULL,
    latitudine_sito LATITUDINE NOT NULL,
    longitudine_sito LONGITUDINE NOT NULL,
    descrizione VARCHAR(100),
    componente_idraulica VARCHAR(25) NOT NULL,

    PRIMARY KEY (latitudine_sito, longitudine_sito, piano, stanza),

    FOREIGN KEY (latitudine_sito, longitudine_sito) REFERENCES Sito(latitudine,
longitudine)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);

-- FollowUp clinico
CREATE TABLE FollowUp_clinico (
    codice CHAR(6) NOT NULL,

    PRIMARY KEY (codice)
);

-- Richiedente
CREATE TABLE Richiedente (
    codice CHAR(6) NOT NULL,
    nome VARCHAR(25),

    PRIMARY KEY (codice)
);

-- Indagine ambientale
CREATE TABLE Indagine_ambientale (
    codice CHAR(6) NOT NULL,
    codice_FollowUp CHAR(6),
    codice_Richiedente CHAR(6),
    data DATE NOT NULL,
```

```
    PRIMARY KEY (codice),

    FOREIGN KEY (codice_FollowUp) REFERENCES FollowUp_clinico(codice)
        ON DELETE SET NULL
        ON UPDATE CASCADE,
    FOREIGN KEY (codice_Richiedente) REFERENCES Richiedente(codice)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);

-- Campione
CREATE TABLE Campione (
    codice CHAR(6) NOT NULL,
    longitudine_sito LONGITUDINE NOT NULL,
    latitudine_sito LATITUDINE NOT NULL,
    piano_punto_prelievo INTEGER NOT NULL,
    stanza_punto_prelievo VARCHAR(15) NOT NULL,
    codice_indagine CHAR(6) NOT NULL,
    temperatura FLOAT NOT NULL,
    matrice MATRICE NOT NULL,
    volume FLOAT_POS NOT NULL,

    PRIMARY KEY (codice),

    FOREIGN KEY (longitudine_sito, latitudine_sito, piano_punto_prelievo,
stanza_punto_prelievo) REFERENCES Punto_di_prelievo(longitudine_sito,
latitudine_sito, piano, stanza)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    FOREIGN KEY (codice_indagine) REFERENCES Indagine_ambientale(codice)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);

-- Analisi PCR
CREATE TABLE Analisi_PCR (
    codice CHAR(6) NOT NULL,
    codice_campione CHAR(6) NOT NULL,
    data_ora DATE NOT NULL,
    esito BOOLEAN NOT NULL,
    µg_l INT_POS NOT NULL,

    PRIMARY KEY (codice),

    FOREIGN KEY (codice_campione) REFERENCES Campione(codice)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

-- Analisi colturale
CREATE TABLE Analisi_culturale (
    codice CHAR(6) NOT NULL,
    codice_campione CHAR(6) NOT NULL,
    data_ora DATE NOT NULL,
    esito BOOLEAN NOT NULL,
    ufc_l INT_POS NOT NULL,
    sierotipo VARCHAR(50),
```

```
    PRIMARY KEY (codice),

    FOREIGN KEY (codice_campione) REFERENCES Campione(codice)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

-- Analisi del pH
CREATE TABLE Analisi_pH (
    codice CHAR(6) NOT NULL,
    codice_campione CHAR(6) NOT NULL,
    data_ora DATE NOT NULL,
    ph PH NOT NULL,

    PRIMARY KEY (codice),

    FOREIGN KEY (codice_campione) REFERENCES Campione(codice)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

-- Analisi genomica
CREATE TABLE Analisi_genomica (
    codice CHAR(6) NOT NULL,
    codice_campione CHAR(6) NOT NULL,
    data_ora DATE NOT NULL,
    genoma TEXT NOT NULL,

    PRIMARY KEY (codice),

    FOREIGN KEY (codice_campione) REFERENCES Campione(codice)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);

-- Gene
CREATE TABLE Gene (
    protein_ID CHAR(6) NOT NULL,
    nome VARCHAR(75),

    PRIMARY KEY (protein_ID)
);

-- Gene del genoma
CREATE TABLE Gene_genoma (
    posizione INTEGER NOT NULL,
    codice_genoma CHAR(6) NOT NULL,
    protein_ID CHAR(6) NOT NULL,
    posizione_predecessore INTEGER,
    codice_genoma_predecessore CHAR(6),
    protein_ID_predecessore CHAR(6),
    query_cover PERCENT NOT NULL,
    percent_identity PERCENT NOT NULL,
    e_value FLOAT_POS NOT NULL,

    PRIMARY KEY (posizione, codice_genoma, protein_ID),

    FOREIGN KEY (codice_genoma) REFERENCES Analisi_genomica(codice)
```

```
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (protein_ID) REFERENCES Gene(protein_ID)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    FOREIGN KEY (posizione_predecessore, codice_genoma_predecessore,
protein_ID_predecessore) REFERENCES Gene_genoma(posizione, codice_genoma,
protein_ID)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);

-- DEFINIZIONE DEGLI INDICI

-- Indice per la tabella Stazione_meteorologica
CREATE INDEX idx_stazione_geom ON Stazione_meteorologica USING GIST (geom);

-- Indice per la tabella Sito
CREATE INDEX idx_sito_geom ON Sito USING GIST (geom);
```

Implementazione dei trigger

```
-- TRIGGER

--1. Trigger per aggiornamento automatico della stazione meteorologica più vicina
-- 1.A : On Insert or Update
SET search_path TO legionella;

CREATE OR REPLACE FUNCTION update_stazione_meteorologica()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    UPDATE Sito
    SET latitudine_stazione_meteorologica = stazione.latitudine,
        longitudine_stazione_meteorologica = stazione.longitudine
    FROM (
        SELECT sito.latitudine AS sito_latitudine,
               sito.longitudine AS sito_longitudine,
               stazione.latitudine,
               stazione.longitudine,
               ST_Distance(sito.geom, stazione.geom) AS distance,
               ROW_NUMBER() OVER (PARTITION BY sito.latitudine, sito.longitudine
ORDER BY ST_Distance(sito.geom, stazione.geom)) AS rn
        FROM Sito sito
        JOIN Stazione_meteorologica stazione
        ON ST_DWithin(
            geography(sito.geom),
            geography(stazione.geom),
            100000 -- 100 km
        )
    ) AS stazione
    WHERE Sito.latitudine = stazione.sito_latitudine
      AND Sito.longitudine = stazione.sito_longitudine
```

```
      AND rn = 1;
    RETURN NULL;
END;
$$;
-- si utilizza ROW_NUMBER per cercare la stazione meteorologica più vicina per
ciascun sito
-- partition by per raggruppare i siti con la stessa latitudine e longitudine

CREATE TRIGGER update_stazione_meteorologica_trigger
AFTER INSERT OR UPDATE ON Stazione_meteorologica
FOR EACH ROW
EXECUTE FUNCTION update_stazione_meteorologica();

-- 1.B : On Delete
CREATE OR REPLACE FUNCTION update_stazione_meteorologica_on_delete()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    -- Verifica se ci sono più di una stazione meteorologica
    IF (SELECT COUNT(*) FROM Stazione_meteorologica) = 1 THEN
        RAISE EXCEPTION 'Non è possibile eliminare tutte le stazioni
meteorologiche.';
    END IF;

    -- Trova la stazione meteorologica più vicina per ciascun sito che aveva come
più vicina la stazione cancellata
    UPDATE Sito
    SET latitudine_stazione_meteorologica = stazione.latitudine,
        longitudine_stazione_meteorologica = stazione.longitudine
    FROM (
        SELECT sito.latitudine AS sito_latitudine, sito.longitudine AS
sito_longitudine,
               stazione.latitudine, stazione.longitudine,
               ST_Distance(sito.geom, stazione.geom) AS distance,
               ROW_NUMBER() OVER (PARTITION BY sito.latitudine, sito.longitudine
ORDER BY ST_Distance(sito.geom, stazione.geom)) AS rn

        FROM Sito sito
        JOIN Stazione_meteorologica stazione
        ON ST_DWithin(
            geography(sito.geom),
            geography(stazione.geom),
            100000 -- 100 km
        )
        WHERE (stazione.latitudine != OLD.latitudine OR stazione.longitudine !=
OLD.longitudine)

    ) AS stazione
    WHERE Sito.latitudine = stazione.sito_latitudine
      AND Sito.longitudine = stazione.sito_longitudine
      AND rn = 1;

    RETURN NULL;
END;
$$;

CREATE TRIGGER update_stazione_meteorologica_on_delete_trigger
BEFORE DELETE ON Stazione_meteorologica
```

```
FOR EACH ROW
EXECUTE FUNCTION update_stazione_meteorologica_on_delete();


--2. Analisi PCR
CREATE OR REPLACE FUNCTION check_esito_PCR()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    IF NEW.esito = TRUE AND NEW.µg_l = 0 THEN
        RAISE EXCEPTION 'Il valore µg/l deve essere maggiore di 0 quando l''esito è
positivo.';
    ELSIF NEW.esito = FALSE AND NEW.µg_l > 0 THEN
        RAISE EXCEPTION 'Il valore µg/l deve essere 0 quando l''esito è negativo.';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER check_esito_PCR
BEFORE INSERT OR UPDATE ON Analisi_PCR
FOR EACH ROW
EXECUTE FUNCTION check_esito_PCR();


--3. Analisi colturale
CREATE OR REPLACE FUNCTION check_esito_Colturale()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    IF NEW.esito = TRUE AND NEW.ufc_l = 0 THEN
        RAISE EXCEPTION 'Il valore ufc/l deve essere maggiore di 0 quando l''esito è
positivo.';
    ELSIF NEW.esito = FALSE AND NEW.ufc_l > 0 THEN
        RAISE EXCEPTION 'Il valore ufc/l deve essere 0 quando l''esito è negativo.';
    ELSIF NEW.esito = FALSE AND NEW.sierotipo IS NOT NULL THEN
        RAISE EXCEPTION 'Il sierotipo non può essere specificato quando l''esito è
negativo.';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER check_esito_Colturale
BEFORE INSERT OR UPDATE ON Analisi_culturale
FOR EACH ROW
EXECUTE FUNCTION check_esito_Colturale();


--4. Sito dei campioni di un'indagine
CREATE OR REPLACE FUNCTION check_campione_indagine()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM Indagine_ambientale JOIN Campione ON codice = codice_indagine
        WHERE codice = NEW.codice_indagine
```

```
        AND (latitudine_sito != NEW.latitudine_sito OR longitudine_sito !=
NEW.longitudine_sito)
    ) THEN
        RAISE EXCEPTION 'I campioni raccolti nell''ambito di una stessa indagine
devono essere prelevati nel medesimo sito.';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER check_campione_indagine
BEFORE INSERT OR UPDATE ON Campione
FOR EACH ROW
EXECUTE FUNCTION check_campione_indagine();

-- 5. Sequenzialità geni del genoma

--5.A : Trigger per controllare che un gene non venga associato a se stesso o a un
genoma diverso
CREATE OR REPLACE FUNCTION check_genoma()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    IF NEW.codice_genoma != NEW.codice_genoma_predecessore THEN
        RAISE EXCEPTION 'Non è possibile associare un gene a un genoma differente.';
    END IF;
    IF NEW.posizione = NEW.posizione_predecessore AND NEW.codice_genoma =
NEW.codice_genoma_predecessore AND NEW.protein_ID = NEW.protein_ID_predecessore THEN
        RAISE EXCEPTION 'Non è possibile associare un gene a se stesso.';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER check_genoma
BEFORE INSERT OR UPDATE ON Gene_genoma
FOR EACH ROW
EXECUTE FUNCTION check_genoma();

--5.B : Trigger per controllare che il gene predecessore sia corretto
CREATE OR REPLACE FUNCTION check_predecessore()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM Gene_genoma
        WHERE codice_genoma = NEW.codice_genoma_predecessore AND posizione >
NEW.posizione_predecessore AND posizione < NEW.posizione
    ) THEN
        RAISE EXCEPTION 'Il gene predecessore non è corretto: esiste un gene con
posizione compresa tra la posizione del gene e quella del gene predecessore.';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER check_predecessore
```

```
BEFORE INSERT OR UPDATE ON Gene_genoma
FOR EACH ROW
EXECUTE FUNCTION check_predecessore();

--6. Eliminazione di un'indagine se non ci sono più campioni associati
CREATE OR REPLACE FUNCTION delete_indagine()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    IF NOT EXISTS (
        SELECT 1
        FROM Campione
        WHERE codice_indagine = OLD.codice
    ) THEN
        DELETE FROM Indagine_ambientale
        WHERE codice = OLD.codice;
    END IF;
END;
$$;

CREATE TRIGGER delete_indagine
AFTER DELETE ON Campione
FOR EACH ROW
EXECUTE FUNCTION delete_indagine();

--7. Eliminazione di un richiedente o di un follow-up se non ci sono più indagini
associate

CREATE OR REPLACE FUNCTION delete_richiedente_follow_up()
RETURNS TRIGGER LANGUAGE plpgsql AS
$$
BEGIN
    IF NOT EXISTS (
        SELECT 1
        FROM Indagine_ambientale
        WHERE codice_richiedente = OLD.codice
    ) THEN
        DELETE FROM Richiedente
        WHERE codice = OLD.codice;
    END IF;
    IF NOT EXISTS (
        SELECT 1
        FROM Indagine_ambientale
        WHERE codice_follow_up = OLD.codice
    ) THEN
        DELETE FROM Follow_up
        WHERE codice = OLD.codice;
    END IF;
END;
$$;

CREATE TRIGGER delete_richiedente_follow_up
AFTER DELETE ON Indagine_ambientale
FOR EACH ROW
EXECUTE FUNCTION delete_richiedente_follow_up();
```

Creazione di user define function per inserimento e aggiornamento

```
-- 1. Inserimento di una stazione meteorologica
CREATE OR REPLACE FUNCTION insert_stazione_meteorologica(
    latitudine FLOAT,
    longitudine FLOAT,
    via_piazza VARCHAR(25),
    numero_civico INTEGER,
    CAP INTEGER,
    città VARCHAR(25)
)
RETURNS VOID LANGUAGE plpgsql AS
$$
DECLARE
    point GEOMETRY;
BEGIN
    -- Crea un punto geometrico a partire da latitudine e longitudine
    point := ST_SetSRID(ST_MakePoint(longitudine, latitudine), 4326);

    -- Inserisci i valori nella tabella Stazione_meteorologica
    INSERT INTO Stazione_meteorologica (latitudine, longitudine, via_piazza, civico,
CAP, città, geom)
    VALUES (latitudine, longitudine, via_piazza, numero_civico, CAP, città, point);

    RETURN;
END;
$$;

-- 2. Aggiornamento di una stazione meteorologica
CREATE OR REPLACE FUNCTION update_stazione_meteorologica(
    old_latitudine FLOAT,
    old_longitudine FLOAT,
    new_latitudine FLOAT,
    new_longitudine FLOAT,
    new_via_piazza VARCHAR(25),
    new_numero_civico INTEGER,
    new_CAP CAP,
    new_città VARCHAR(25)
)
RETURNS VOID LANGUAGE plpgsql AS
$$
DECLARE
    point GEOMETRY;
BEGIN
    -- Crea il punto geometrico a partire dalle nuove coordinate
    point := ST_SetSRID(ST_MakePoint(new_longitudine, new_latitudine), 4326);

    -- Esegui l'aggiornamento della stazione meteorologica
    UPDATE Stazione_meteorologica
    SET latitudine = new_latitudine,
        longitudine = new_longitudine,
        via_piazza = new_via_piazza,
        civico = new_numero_civico,
        CAP = new_CAP,
```

```
        città = new_città,
        geom = point
    WHERE latitudine = old_latitudine
      AND longitudine = old_longitudine;

    RETURN;
END;
$$;

-- 3. Inserimento di un sito

CREATE OR REPLACE FUNCTION insert_sito(
    latitudine LATITUDINE,
    longitudine LONGITUDINE,
    via_piazza VARCHAR(25),
    numero_civico INTEGER,
    CAP CAP,
    città VARCHAR(25),
    nome VARCHAR(25),
    categoria CATEGORIA,
    materiale_tubature VARCHAR(25),
    cloro BOOLEAN,
    anno_ultima_ristrutturazione DATE,
    caldaia VARCHAR(25)
)

RETURNS VOID LANGUAGE plpgsql AS
$$
DECLARE
    point GEOMETRY;
    latitudine_stazione_meteorologica LATITUDINE;
    longitudine_stazione_meteorologica LONGITUDINE;
BEGIN
    -- Crea un punto geometrico per il sito
    point := ST_SetSRID(ST_MakePoint(longitudine, latitudine), 4326);

    SELECT stazione.latitudine, stazione.longitudine
    INTO latitudine_stazione_meteorologica, longitudine_stazione_meteorologica
    FROM Stazione_meteorologica stazione
    WHERE ST_DWithin(
        geography(point),
        geography(stazione.geom),
        100000 -- 100 km
    )
    ORDER BY ST_Distance(point, stazione.geom)
    LIMIT 1;

    -- Inserisci i valori nella tabella Sito
    INSERT INTO Sito (latitudine, longitudine, latitudine_stazione_meteorologica,
longitudine_stazione_meteorologica, via_piazza, civico, CAP, città, nome, categoria,
materiale_tubature, cloro, anno_ultima_ristrutturazione, caldaia, geom)

    VALUES (latitudine, longitudine, latitudine_stazione_meteorologica,
longitudine_stazione_meteorologica, via_piazza, numero_civico, CAP, città, nome,
categoria, materiale_tubature, cloro, anno_ultima_ristrutturazione, caldaia, point);

    RETURN;
END;
```

```
$$;

-- 4. Aggiornamento di un sito
CREATE OR REPLACE FUNCTION update_sito(
    old_latitudine LATITUDINE,
    old_longitudine LONGITUDINE,
    new_latitudine LATITUDINE,
    new_longitudine LONGITUDINE,
    new_via_piazza VARCHAR(25),
    new_numero_civico INTEGER,
    new_CAP CAP,
    new_città VARCHAR(25),
    new_nome VARCHAR(25),
    new_categoria CATEGORIA,
    new_materiale_tubature VARCHAR(25),
    new_cloro BOOLEAN,
    new_anno_ultima_ristrutturazione DATE,
    new_caldaia VARCHAR(25)
)
RETURNS VOID LANGUAGE plpgsql AS
$$
DECLARE
    point GEOMETRY;
    new_latitudine_stazione_meteorologica LATITUDINE;
    new_longitudine_stazione_meteorologica LONGITUDINE;
BEGIN
    point := ST_SetSRID(ST_MakePoint(new_longitudine, new_latitudine), 4326);

    SELECT stazione.latitudine, stazione.longitudine
    INTO new_latitudine_stazione_meteorologica,
new_longitudine_stazione_meteorologica
    FROM Stazione_meteorologica stazione
    WHERE ST_DWithin(
        geography(point),
        geography(stazione.geom),
        100000 -- 100 km
    )
    ORDER BY ST_Distance(point, stazione.geom)
    LIMIT 1;

    UPDATE Sito
    SET latitudine = new_latitudine,
        longitudine = new_longitudine,
        latitudine_stazione_meteorologica = new_latitudine_stazione_meteorologica,
        longitudine_stazione_meteorologica = new_longitudine_stazione_meteorologica,
        via_piazza = new_via_piazza,
        civico = new_numero_civico,
        CAP = new_CAP,
        città = new_città,
        nome = new_nome,
        categoria = new_categoria,
        materiale_tubature = new_materiale_tubature,
        cloro = new_cloro,
        anno_ultima_ristrutturazione = new_anno_ultima_ristrutturazione,
        caldaia = new_caldaia,
        geom = point
    WHERE latitudine = old_latitudine
      AND longitudine = old_longitudine;
```

```
    RETURN;
END;
$$;
```

Bibliografia

[1] M. della Salute, «Linee guida per la prevenzione e il controllo della legionellosi», 2015, [Online]. Disponibile su: https://www.salute.gov.it/portale/malattieInfettive/dettaglioPubblicazioniMalattieInfettive.jsp?id=2362

[2] A. Felice, M. Franchi, S. De Martin, N. Vitacolonna, L. Iacumin, e M. Civilini, «Environmental surveillance and spatio-temporal analysis of Legionella spp. in a region of northeastern Italy (2002–2017)», *PLOS ONE*, vol. 14, fasc. 7, p. e218687, 2019, doi: 10.1371/journal.pone.0218687.

[3] E. F. Codd, «A relational model of data for large shared data banks», *Communications of the ACM*, vol. 13, fasc. 6, pp. 377–387, 1970, doi: 10.1145/362384.362685.

[4] D. Garlatti, «Base di dati e applicazione web per il monitoraggio del batterio della legionella», 2020.

[5] P. Atzeni, S. Ceri, S. Paraboschi, e R. Torlone, *Database Systems: Concepts, Languages & Architectures*. 1999.

[6] «BLAST Glossary». [Online]. Disponibile su: https://www.ncbi.nlm.nih.gov/books/NBK62051

[7] «PostgreSQL 16.4 documentation». [Online]. Disponibile su: https://www.postgresql.org/docs/current

[8] A. Colautti *et al.*, «Draft genome sequences from 127 Legionella spp. strains isolated in water systems linked to legionellosis outbreaks», *Microbiol Resour Announc*, vol. 13, fasc. 6, 2024, doi: 10.1128/mra.01154-23.

[9] «PostGIS 3.5.0Dev Manual». [Online]. Disponibile su: https://postgis.net/docs/manual-dev/en

[10] «Polymerase chain Reaction (PCR)». [Online]. Disponibile su: https://www.genome.gov/genetics-glossary/Polymerase-Chain-Reaction

Glossario

Al fine di facilitare la comprensione dell'elaborato, è redatto il seguente glossario contenente le definizioni dei termini tecnici utilizzati.

Termine	**Definizione**
Aerosol	Particelle sospese nell'aria, contenenti gocce d'acqua, che possono trasportare il batterio Legionella.
Analisi	Esame di laboratorio effettuato su campioni di acqua prelevati durante un'indagine ambientale.
Analisi Colturale	Esame di laboratorio che permette di isolare e identificare le unità formanti colonia (UFC_L) di Legionella in un campione di acqua.
Attributo	Concetto che descrive una proprietà o una componente di una entità o di una relazione (*i.e.* campo).
Attributo composto	Attributo dalla struttura complessa, costituito da diversi sotto-attributi.
Attributo multivalore	Attributo che, per ogni istanza dell'entità cui è associato, può assumere più di un valore.
Campione	Piccola quantità di acqua da sottoporre a esame.
Categoria	Classificazione di un sito, o più specificamente di un edificio, in base alla sua destinazione d'uso, come ad esempio ospedaliero, termale o alberghiero.
Chiave primaria	Attributo o insieme di attributi che identifica univocamente ogni istanza di un'entità.

Glossario

Termine	Definizione
Componente idraulica	Componente di un sistema idraulico da cui viene prelevato un campione di acqua, come un rubinetto o un filtro di un impianto di condizionamento.
Entità	In riferimento allo schema E-R, descrive una classe di oggetti con esistenza autonoma, con particolare significato nel contesto in esame (*i.e.* tabella).
Entità debole	Entità che non ha una chiave primaria propria, ma dipende da un'altra entità per la sua identificazione.
Generalizzazione	In riferimento al modello E-R, relazione che associa ad un'entità genitore una o più entità figlie, che ereditano le proprietà del genitore (*i.e.* specializzazione).
FollowUp Clinico	Indagine ambientale, o indagini ambientali, condotte a seguito di uno o più casi di legionellosi. Tali indagini non si limitano al domicilio del paziente, ma possono estendersi a tutti i luoghi frequentati dal malato nei dieci giorni precedenti l'insorgenza dei sintomi. La decisione di effettuare tali indagini è lasciata al competente servizio territoriale che «deve valutare di volta in volta l'opportunità di effettuare o meno dei campionamenti ambientali, sulla base della valutazione dei rischio»[31].
Indagine Ambientale	Collezione di campioni prelevati da un sito specifico in una data specifica.

Glossario

[31] [1], «Linee guida per la prevenzione ed il controllo della legionellosi», p. 30

Termine	Definizione
PCR	Polymerase Chain Reaction, è una «tecnica di laboratorio per produrre rapidamente (amplificare) milioni o miliardi di copie di uno specifico segmento di DNA, che può poi essere studiato in modo più dettagliato. La PCR prevede l'uso di brevi frammenti di DNA sintetico chiamati primer per selezionare un segmento del genoma da amplificare, e quindi più cicli di sintesi del DNA per amplificare quel segmento»[32].
PCR Qualitativa	Esame di laboratorio che fornisce un'informazione dicotomica sulla presenza di Legionella in un campione.
PCR Quantitativa	Esame di laboratorio rapido che rileva e quantifica il DNA o l'RNA di Legionella presenti in un campione (*i.e.* Real-Time PCR o qPCR).
Relazione	In riferimento allo schema E-R, legame che rappresenta la connessione logica e significativa per la realtà modellata, tra due o più entità.
Relazione Ricorsiva	Relazione che associa una entità a se stessa (*i.e.* relazione autoreferenziale).
Richiedente	Ente o istituzione che richiede un'indagine ambientale.
Sierotipo	Livello di classificazione di batteri di Legionella inferiore a quello specie. Il laboratorio ARPA distingue tre sierotipi: sierotipo 1, sierotipo 2-15 e sierotipo sp (*i.e.* sierogruppo).

Glossario

[32] [10] «Polymerase chain Reaction (PCR)». [Online]. Disponibile su: https://www.genome.gov/genetics-glossary/Polymerase-Chain-Reaction

Termine	Definizione
Sito	Edificio presso il quale è condotta un'indagine ambientale.
UFC_L	Unità formanti colonie per litro: ovvero unità di misura utilizzata per indicare la concentrazione di Legionella in un campione d'acqua destinato all'analisi colturale.
UG_L	Microgrammi per litro: ovvero unità di misura utilizzata per determinare la concentrazione di Legionella in un campione d'acqua mediante PCR quantitativa.

Glossario