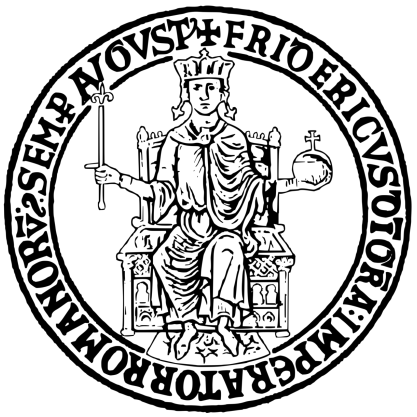


Progettazione e sviluppo di una Base di dati relazionale per la gestione di un laboratorio scientifico

Cardone Ugo Luca
N86004038

Roberta Beneduce
N86004273

pagina volontariamente lasciata vuota

Capitolo 1

DESCRIZIONE DEL PROGETTO

1.1 Analisi del problema.

Ci è stato chiesto di sviluppare e progettare una base di dati relazionale per la gestione di un laboratorio scientifico. La prima fase che abbiamo dovuto affrontare è stata la fase dell' **Analisi dei requisiti** che consiste nell'individuazione e nello studio delle proprietà e delle funzionalità che il sistema informativo dovrà avere. Una volta raccolti i requisiti, occorre analizzarli ottenendo una descrizione sintetica e precisa che isoli le entità e le relazioni che intercorrono tra di esse.

La traccia a noi assegnata è la seguente:

<< Si sviluppi un sistema informativo, composto da una base di dati relazionale e da un applicativo Java dotato di GUI (Swing o JavaFX), per la gestione di un laboratorio scientifico. Ogni laboratorio ha una descrizione ed una o più sedi, per ognuna delle quali c'è un responsabile, e un numero variabile di tecnici. >>

Da queste righe abbiamo individuato le entità LABORATORIO e SEDE che hanno una relazione di tipo molti a molti, in quanto nella traccia si evince che un laboratorio ha una o più sedi e noi abbiamo supposto che in una sede si trovino uno o più laboratori.

<< …quali c'è un responsabile, e un numero variabile di tecnici. Di tutti si conservano i dati anagrafici, la matricola e i contatti telefonici e di email. >>

Da qua abbiamo ricavato l'entità PERSONALE da cui derivano le entità TECNICO e RESPONSABILE entrambi descritti dagli attributi che descrivono l'entità padre PERSONALE. Questo tipo di rappresentazione prende il nome di GENERALIZZAZIONE di cui parleremo in seguito.

<< Ogni postazione ospita uno o più strumenti, e può essere occupata contemporaneamente da una o più persone che usano strumenti differenti. Per ogni strumento si conservano le caratteristiche tecniche e la descrizione.>>

Sono state individuate poi le entità POSTAZIONE, STRUMENTO e UTENTE. Le relazioni che intercorrono tra queste entità sono: dalla traccia si deduce che una postazione ospiti uno o più strumenti, una postazione può essere occupata da una o più persone (nel nostro caso abbiamo deciso di utilizzare l'entità UTENTE come persona che occupa la postazione) e inoltre questo utente può utilizzare più strumenti diversi e quindi uno strumento può essere utilizzato da più persone ma non contemporaneamente.

<< Deve essere possibile prenotare uno strumento, indicando un tempo che non può però essere superiore al massimo tempo d'uso consentito. L'utente deve poter selezionare lo strumento per descrizione o per sede, e vederne il calendario di prenotazione. Può effettuare una prenotazione, cancellarla o, eventualmente, modificarla. Per ogni strumento, inoltre, si vorrebbe un riepilogo, su base mensile ed annuale, di quanto è stato usato e di quale utente lo ha usato di più. >>

Da qui abbiamo individuato l'entità PRENOTAZIONE. Su cui possono essere fatte operazioni di inserimento, modifica e cancellazione da parte dell' UTENTE(in questo caso l'utente non è più la persona che occupa la postazione o utilizza lo strumento, ma è l'utente registrato che effettua la prenotazione).

Nelle pagine che seguono sono descritte in modo più approfondito le entità e le varie relazioni.

Capitolo 2

PROGETTAZIONE CONCETTUALE

2.1 Alcuni cenni di teoria

La fase che segue l'analisi dei requisiti è la **‘ Progettazione concettuale della basi di dati'.** L'obiettivo associato a questa fase è quello di formalizzare i requisiti della base di dati in uno **schema concettuale** indipendente sia dal tipo che dallo specifico DBMS che verrà utilizzato.
Il modello di dati più utilizzato è il modello **Entità-Relazione (ER)** che serve per definire gli aspetti statici del sistema, cioè i dati. Esso è un modello che descrive le entità da modellare (ad esempio, nel nostro contesto, *laboratorio* e *sede*), gli attributi delle entità (ad esempio descrizione per il *laboratorio* o caratteristiche tecniche per lo *strumento*), le relazioni che intercorrono tra le entità (ad esempio la relazione *ospita* che associa *strumenti* alla *postazione*) e le cardinalità delle relazioni (un *laboratorio* si può trovare in una o più sedi e una *sede* può avere uno o più laboratori).

Un altro modello utilizzato è **Unified Modeling Language (UML)** che è un linguaggio grafico per la modellazione di applicazioni software basate sulla programmazione ad oggetti.

Il modello ER descrive i dati con tre concetti fondamentali:

- **Entità:** rappresenta un oggetto o un concetto del mondo reale che è importante per il sistema di gestione dei dati
- **Attributi:** proprietà o caratteristiche dell'entità stessa
- **Relazione:** corrisponde a legami logici tra entità. Sono caratterizzate da un nome e una certa cardinalità.

Il modello UML descrive i dati con questi tre concetti:

- **Classi:** Una classe rappresenta la descrizione di un insieme di oggetti omogenei individuati da comportamenti e proprietà comuni.
- **Attributi:** sono proprietà definite da una classe e possiamo vedere che hanno un nome e un tipo
- **Associazioni:** esprimono una connessione logica tra classi. È possibile associare un nome ad un'associazione, e specificare il ruolo che una certa classe ricopre nell'associazione, anche in questo caso hanno una cardinalità.

2.2 Class diagram / Diagramma ER

CLASS DIAGRAM

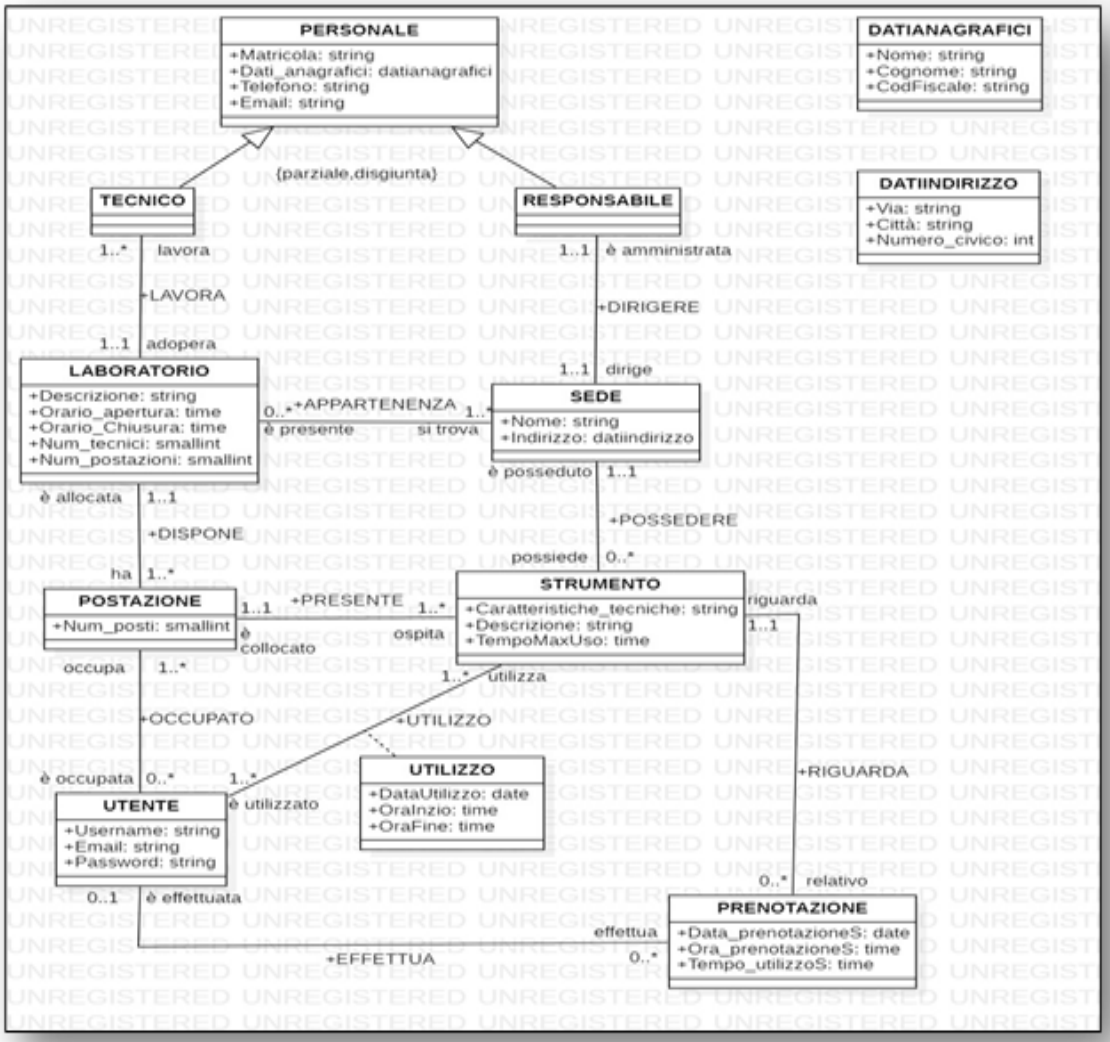
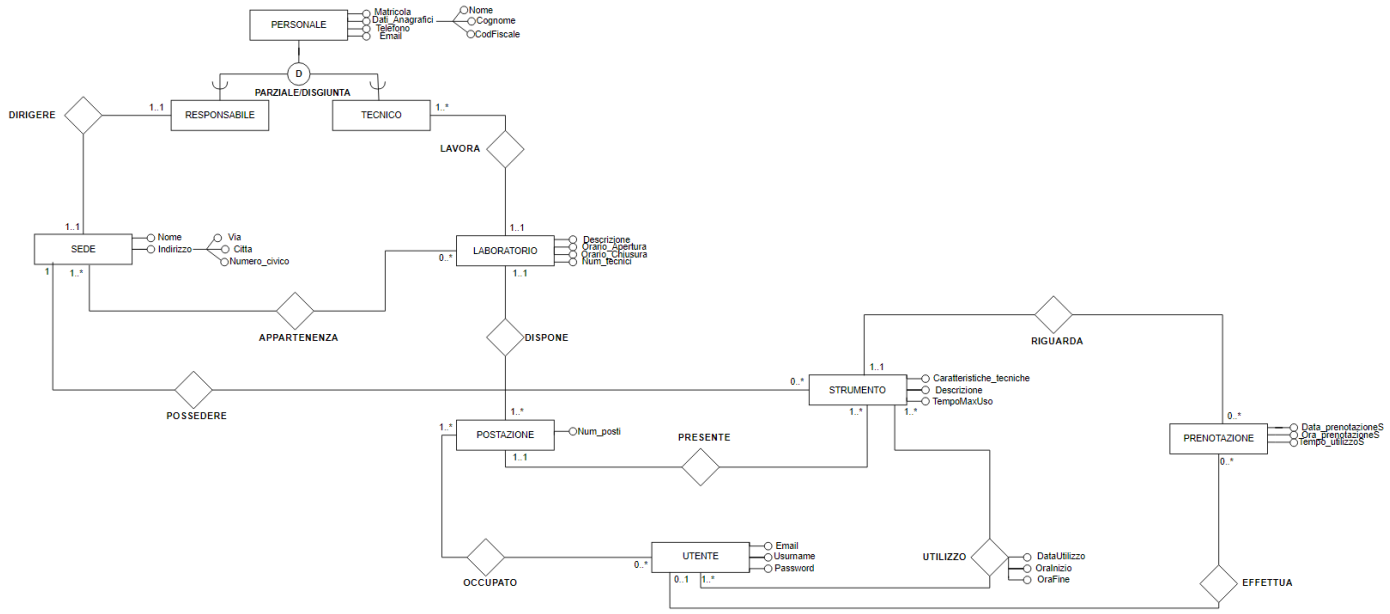


DIAGRAMMA ER



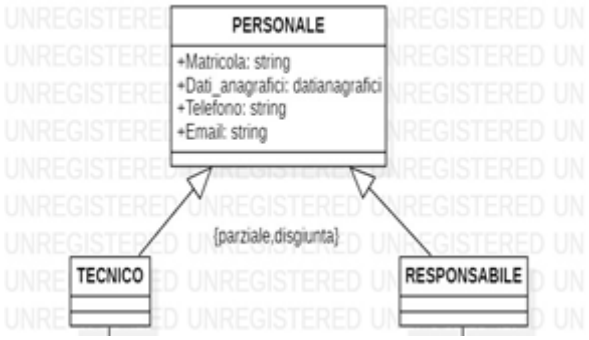
Possiamo notare che tra i due schemi cambia solo la rappresentazione grafica ma non l'approccio alla progettazione. Nelle pagine a seguire vengono usati i termini che descrivono il class diagram in UML ma il discorso è analogo anche per il diagramma ER

2.3 Ristrutturazione

La fase successiva alla progettazione concettuale è la progettazione logica che dà come output lo schema logico. Prima di passare allo schema logico però è necessario **ristrutturare il Class Diagram** (anche il diagramma ER) per semplificare la traduzione e ottimizzare il progetto. L'obiettivo è quello di creare uno schema logico che descrive tutte le informazioni contenute negli schemi concettuali. Poiché nello schema logico non è possibile rappresentare *attributi composti*, *multipli* e *gerarchie* e per questo motivo bisogna eliminarli.

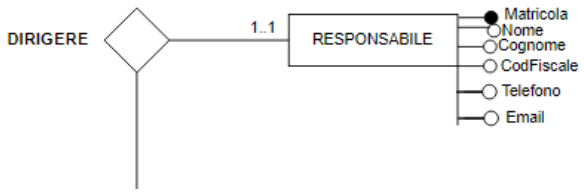
2.3.1 Generalizzazione

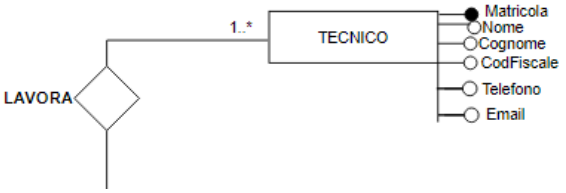
Come possiamo vedere è presente una generalizzazione:



Prendiamo la superclasse (o classe padre) PERSONALE e le sue due sottoclassi (o classi figlio) TECNICO e RESPONSABILE.
Ogni istanza membro di una sottoclasse eredita tutti gli attributi (come in questo caso dove tecnico e responsabile sono descritti dagli stessi attributi della loro classe padre {*matricola, dati_anagrafici, telefono, email*} e tutte le relazioni della classe padre.

Una gerarchia può essere totale o parziale, disgiunta o overlapped in questo caso si può anche notare che questa gerarchia è di tipo “parziale, disgiunta”.
Una gerarchia si dice *parziale* se una classe dell’insieme padre può non appartenere ad alcun insieme figlio (esiste personale che non è nè Tecnico nè Responsabile), si dice disgiunta se ogni occorrenza della classe padre è al più un’occorrenza di una classe figlia (il Personale o è un Tecnico o un Responsabile). In questo caso per ristrutturare questa gerarchia abbiamo deciso di accorpare il padre nei figli in quanto la classe padre non ha un numero significativo di associazioni correlate, infatti le associazioni riguardano le due classi figli, e accorpare i figli nel padre provocherebbe l’introduzione di vincoli che abbiamo ritenuto inutili inserire. Quindi la nostra gerarchia si riduce con le due entità figlio che ereditano gli attributi della classe padre e le rispettive associazioni.





2.3.2 Attributi composti

Nel class diagram sono presenti anche attributi composti *dati_anagrafici* e *indirizzo* che descrivono rispettivamente le classi TECNICO, RESPONSABILE e SEDE.
dati_anagrafci è un attributo composto da *nome*, *cognome*, *codfiscale* mentre indirizzo è composto da *via*, *città* e *numero civico*, poiché, come detto prima, non è possibile rappresentare attributi composti c’è quindi il bisogno di eliminarli.
Ci sono diversi approcci per affrontare il problema e noi abbiamo deciso che per l’attributo dati_anagrafici sono stati considerati singolarmente ogni attributo che compone l’attributo composto, cioè le classi tecnico e responsabile saranno descritte dagli attributi componenti.
Quindi la classe tecnico sarà descritta dagli attributi semplici che già aveva più quelli componenti:
TECNICO = {*Matricola, nome, cognome, codfiscale…ecc*}
Stesso discorso per la classe responsabile.

Al contrario l'attributo *indirizzo* è stato considerato come un'unica stringa perché nella traccia non ci sono richieste o operazioni da effettuare su di esso. Quindi la classe SEDE avrà gli attributi semplici che la descrivevano e questa unica stringa che rappresenta l'indirizzo:
SEDE = {*…, Via starza Napoli 54, …* }
Indirizzo (*via = via starza, città = Napoli, numero civico= 54*)

2.3.3 Informazioni ridondanti

Abbiamo constatato che un'informazione ridondante è l'entità RESPONSABILE.
Dalla traccia non occorre fare particolari operazioni su tale entità e quindi poiché l'associazione tra SEDE e RESPONSABILE è di tipo 1:1 (ad una SEDE corrisponde un unico RESPONSABILE) abbiamo deciso di trasformare l'entità RESPONSABILE in un attributo dell'entità SEDE.
L'entità RESPONSABILE era descritta dagli attributi della sua entità padre, che in fase di ristrutturazione ha ereditato, e nel momento in cui l'abbiamo trasformato in un attributo dell'entità SEDE è diventato un attributo composto da matricola, nome, cognome, codfiscale, telefono, email e dato che c'è il bisogno di eliminarlo essendo composto, abbiamo considerato tale attributo come un'unica stringa.

2.3.4 Analisi degli identificativi

Per la maggior parte delle classi si è deciso di utilizzare chiavi surrogate in modo da evitare l'impiego di chiavi primarie composte. Per convenzione le chiavi primarie inizieranno con il prefisso **Cod**.

2.3.5 Class diagram / diagramma er ristrutturati

CLASS DIAGRAM

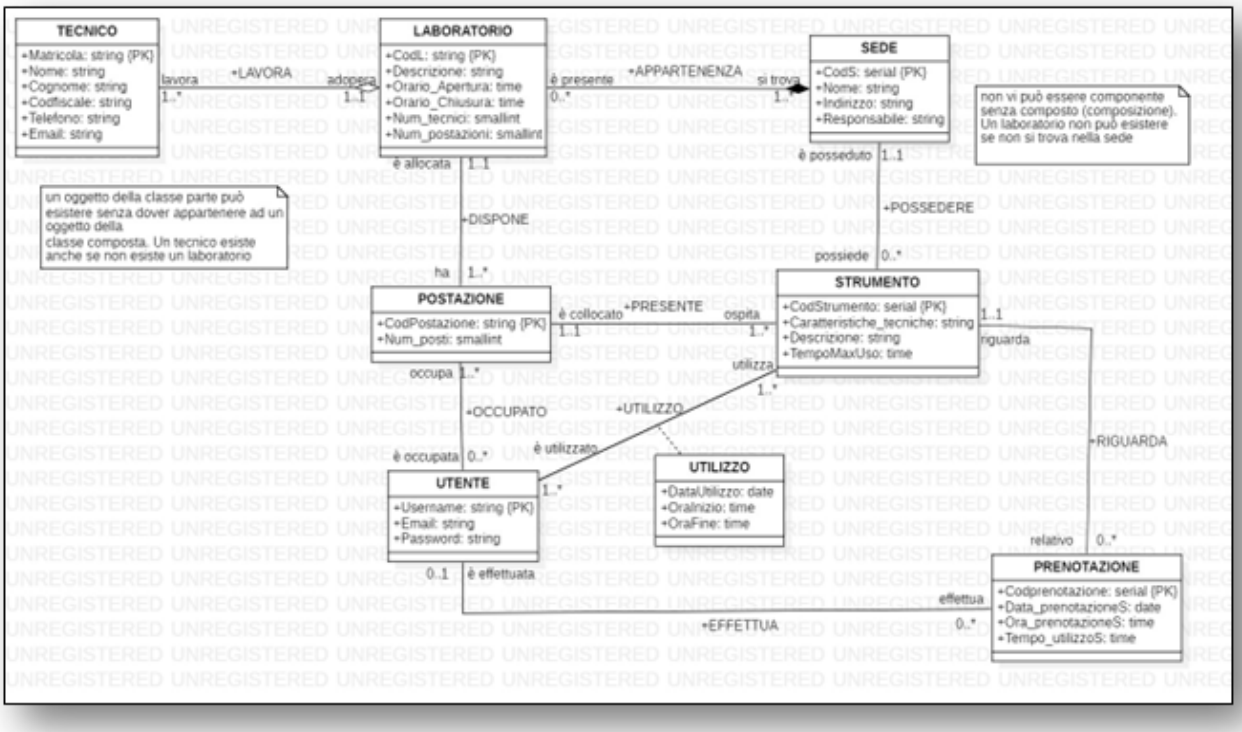
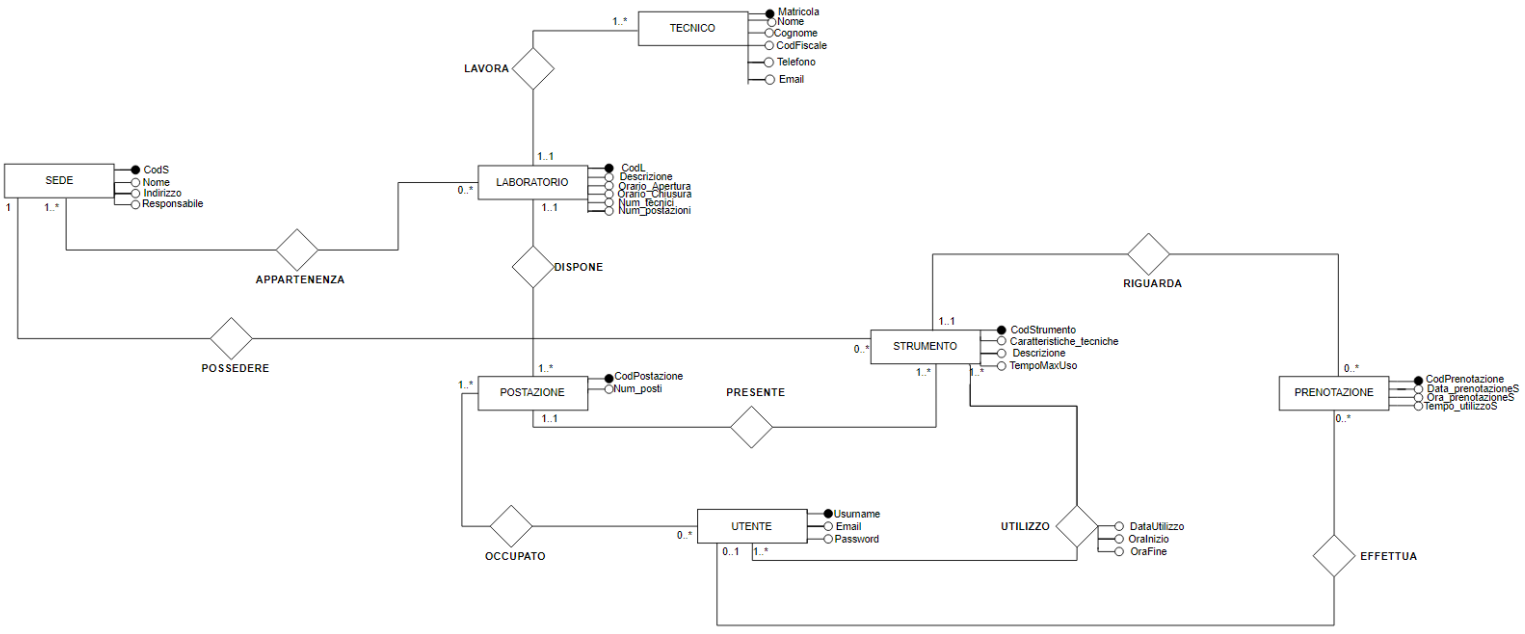


DIAGRAMMA ER



2.3.6 Precisazioni sull' associazione UTILIZZO (UTILIZZATO nel DB)

L'associazione utilizzo tra (UTENTE e STRUMENTO) è stata introdotta per avere un riepilogo degli strumenti utilizzati, come richiesto dalla traccia.
Non potendo, con i mezzi a nostra disposizione, tenere traccia dell'utilizzo effettivo dello strumento, questa associazione o più specificamente tabella ponte (dato che l'associazione in cui è coinvolta è del tipo molti a molti) viene vista come una tabella statica, che quindi non viene popolata dinamicamente ma tramite un trigger che la popola non appena viene inserita o aggiornata una nuova prenotazione.
Poiché tramite l'ausilio di un Trigger le prenotazioni che sono più vecchie di due settimane vengono eliminate dalla tabella PRENOTAZIONE, la tabella UTILIZZO preserva lo storico delle prenotazioni per poi ricavarne i report come richiesto dalla traccia.
Infine possiamo notare che sono presenti anche attributi dell'associazione che sono: *data_utilizzo*, *ora inizio* e *ora_fine.*
La data di utilizzo corrisponde alla data di prenotazione dello strumento, l'ora di inizio corrisponde all' ora di prenotazione e l'ora di fine è un attributo calcolato dalla somma tra l'ora di prenotazione e il tempo di utilizzo.

DIZIONARI

2.4.1 Dizionario delle classi

Classe	Descrizione	Attributi
Laboratorio	Descrittore di un laboratorio	**CodL** (*varchar(3)*): Chiave surrogata. Identifica unicamente ciascuna istanza di Laboratorio. **descrizione** (*varchar(2000)*): Descrizione associata al laboratorio. **orarioApertura** (*time*): Orario di apertura del laboratorio. **orarioChiusura** (*time*): Orario di chiusura del laboratorio. **num_tecnici** (*smallint*): Numero di tecnici presenti nel laboratorio. **num_postazioni** (*smallint*): Numero di postazioni presenti nel laboratorio.
Sede	Descrittore di una sede	**CodS** (*serial*): Chiave surrogata. Identifica unicamente ciascuna istanza di Sede. **nome** (*varchar(35)*): Nome della sede. **indirizzo** (varchar(60)): Indirizzo in cui si trova la sede. **responsabile** (*varchar(70)*): Nome del responsabile della sede.

Classe	Descrizione	Attributi
Appartenenza	Descrittore dell'associazione tra Laboratorio e Sede	**CodL _fk** (*varchar(3)*): Chiave esterna. Referenzia la chiave primaria CodL della classe Laboratorio. **CodS_fk** (Integer) : Chiave esterna. Referenzia la chiave primaria CodS della classe Sede.
Tecnico	Descrittore di un tecnico che lavora in un laboratorio	**Matricola** (*varchar(10)*): Chiave primaria. Identifica unicamente ciascuna istanza di Tecnico. **nome** (*varchar(20)*): Nome del tecnico. **cognome** (*varchar(20)*): Cognome del tecnico. **codfiscale** (varchar(16)): Chiave candidata. Identifica il codice fiscale del tecnico. **telefono** (*varchar(10)*): Numero di telefono del tecnico. **e-mail** (*varchar(30)*): Chiave candidata. Identifica l'e-mail del tecnico. **CodL_fk** (*varchar(3)*): Chiave esterna. Referenzia la chiave primaria CodL della classe Laboratorio.

Classe	Descrizione	Attributi
Postazione	Descrittore di una postazione presente in un laboratorio	**CodPostazione** (*varchar(3)*): Chiave surrogata. Indentifica unicamente ciascuna istanza di Postazione. **num_posti** (*smallint*): Descrizione associata al laboratorio. **CodL_fk** (*varchar(3)*): Chiave esterna. Referenzia la chiave primaria CodL della classe Laboratorio.
Strumento	Descrittore di uno strumento	**CodStrumento** (*serial*): Chiave surrogata. Identifica unicamente ciascuna istanza di Strumento. **caratteristiche_tecniche** (*varchar(1000)*): Caratteristiche tecniche relative allo strumento- **descrizione** (*varchar*(200)): descrizione associata allo strumento. **tempoMaxUso** (*time*): Tempo di utilizzo massimo per uno strumento. **CodPostazione_fk** (*varchar(3)*): Chiave esterna. Referenzia la chiave primaria CodPostazione della classe Postazione. **CodSede_fk** (*integer*): Chiave esterna. Referenzia la chiave primaria CodS della classe Sede.

Classe	Descrizione	Attributi
Utente	Descrittore di un utente	**Username** (*varchar(20)*): Chiave primaria. Indentifica unicamente ciascuna istanza di Utente. **e-mail** (*varchar(30)*): Chiave candidata. Identifica l'e-mail del tecnico. **password** (*varchar(30)*): Password per accedere al profilo utente
Prenotazione	Descrittore di una prenotazione per uno strumento	**CodPrenotazione** (*serial*): Chiave surrogata. Identifica unicamente ciascuna istanza di Prenotazione. **data_prenotazioneS** (*date*): Data di prenotazione dello strumento. **ora_prenotazioneS** (*time*): Ora di prenotazione dello strumento. **tempo_utilizzoS** (*time*): tempo di utilizzo dello strumento. **Username_fk** (*varchar(20)*): Chiave esterna. Referenzia la chiave primaria Username della classe Utente. **CodStrumento_fk** (*integer*): Chiave esterna. Referenzia la chiave primaria CodStrumento della classe Strumento.

Classe	Descrizione	Attributi
Occupato	Descrittore dell'associazione tra Utente e Postazione	**CodPostazione_fk** (*varchar(3)*): Chiave esterna. Referenzia la chiave primaria CodPostazione della classe Postazione. **Username_fk** (*varchar(20)*): Chiave esterna. Referenzia la chiave primaria Username della classe Utente.
Utilizzato	Descrittore dell'associazione tra Utente e Strumento	**Username_fk** (*varchar(20)*): Chiave esterna. Referenzia la chiave primaria Username della classe Utente. **CodStrumento_fk** (*integer*): Chiave esterna. Referenzia la chiave primaria CodStrumento della classe Strumento. **dataUtilizzo** (*date*): Data di utilizzo dello strumento prenotato. **oraInizio** (*time*): Ora di inizio di utilizzo dello strumento. **oraFine**(*time*): Ora di fine di utilizzo dello strumento prenotato.

2.4.2 Dizionario delle associazioni

Associazione	Descrizione	Classi coinvolte
Lavora	Esprime la presenza di un tecnico che lavora in un laboratorio. In un laboratorio lavorano uno o più tecnici. Un tecnico lavora in un solo laboratorio	**Laboratorio** [1..1]: ruolo (lavora) indica un laboratorio **Tecnico** [1..*]: ruolo (adopera) indica un tecnico che lavora nel laboratorio
Appartenenza	Esprime la relazione di appartenenza di una sede con un laboratorio. In una sede ci possono essere zero o più laboratori. Un laboratorio si trova in una o più sedi.	**Laboratorio** [0..*]: ruolo (si trova) indica un laboratorio che si trova in una sede **Sede** [1..*]: ruolo (è presente) indica la sede in cui si trova il laboratorio
Possedere	Esprime la relazione di presenza di uno strumento in una sede. Una sede può possedere zero o più strumenti. Uno strumento si trova in una sede.	**Sede** [1..1]: ruolo (possiede) indica la sede in cui si trova lo strumento **Strumento** [0..*]: ruolo (è posseduto) indica lo strumento posseduto nella sede
Dispone	Esprime la presenza delle postazioni in un laboratorio. Un laboratorio ha una o più postazioni, mentre una postazione si trova in un solo laboratorio	**Laboratorio** [1..1]: ruolo (ha) indica il laboratorio **Postazione** [1..*]: ruolo (è allocata) indica la postazione presente nel laboratorio
Occupato	Esprime l'occupazione di una postazione da vari utenti. Una postazione può essere occupata da zero o più utenti. Un utente può occupare una o più postazioni.	**Postazione** [1..*]: ruolo (è occupata) indica la postazione occupata da vari utenti **Utente** [0..*]: ruolo (occupa) indica l'utente che occupa la postazione

Presente	Esprime la presenza di vari strumenti in una postazione. Una postazione dispone di uno o più strumenti. Uno strumento è presente in una postazione	**Postazione** [1..1]: ruolo (ospita) indica la postazione in cui si trovano gli strumenti **Strumento** [1..*]: ruolo (è collocato) indica lo strumento presente in una postazione
Riguarda	Esprime la prenotazione relativa allo strumento. Uno strumento è relativo a zero o più prenotazioni. Una prenotazione riguarda un solo strumento	**Prenotazione** [0..*]: ruolo (riguarda) indica la prenotazione ad uno strumento **Strumento** [1..1]: (relativo) indica lo strumento prenotato
Effettua	Esprime l'operazione di prenotazione da parte di un utente. Un utente può effettuare zero o più prenotazioni, mentre una prenotazione può essere effettuata da zero o un utente.	**Utente** [0..1]: ruolo (effettua) indica l'utente che ha effettuato la prenotazione **Prenotazione** [0..*]: ruolo (è effettuata) indica la prenotazione effettuata dall'utente
Utilizzo	Esprime l'utilizzo effettivo di uno strumento a seguito di una prenotazione da parte dell'utente. Un utente utilizza uno o più strumenti. Uno strumento è utilizzato da uno o più utenti.	**Strumento** [1..*]: ruolo (utilizza) indica lo strumento utilizzato dall'utente **Utente** [1..*]: ruolo (è utilizzato) indica l'utente che ha utilizzato lo strumento

2.4.1 Dizionario dei vincoli

Sono stati anche individuati dei vincoli di integrità sui dati, che rappresentano le condizioni che i dati devono rispettare per essere consistenti, validi.

I vincoli di integrità individuati sono:

- Il vincolo di integrità implicito della chiave primaria il quale stabilisce che in ogni relazione deve esistere una chiave primaria. Tale vincolo viene implementato tramite la parola chiave **PRIMARY KEY** che troviamo dopo il tipo dell'attributo ritenuto idoneo ad essere chiave primaria.

- Il vincolo di integrità referenziale così definito in quanto legato alla presenza di chiavi primarie di relazione in altre come chiavi esterne. Tale vincolo è implementato tramite la parola chiave **FOREIGN KEY REFERENCES.**

- Il vincolo di chiave candidata (che è una chiave appunto ad essere candidata come chiave primaria e come tale è unica) che è implementato tramite la parola chiave **UNIQUE** è posta dopo il tipo dell'attributo.

- E poi ci sono vincoli espliciti definiti da noi che sono:

Nome vincolo	**Descrizione**
Orariol	Vincolo che permette ai laboratori di aprire alle 8:00 e chiudere alle 20:00
num_tecnici_valido	Vincolo che permette di avere per ogni laboratorio un numero di tecnici compreso tra 0 e 100.
num_postazioni_valido	Vincolo che permette di avere per ogni laboratorio un numero di postazioni compreso tra 0 e 50
codicefiscale_valido	Un codice fiscale per essere valido deve contenere 16 caratteri e tale vincolo verifica ciò.

telefono_valido	IL numero di telefono è solitamente composto da 10 numeri e tale vincolo verifica che il numero di telefono di ogni tecnico sia valido.
username_valido	Vincolo che verifica che l'username dell'utente sia valido. Per valido si intende che non contenga caratteri speciali come '- ', '?' e che la lunghezza sia almeno di 10.
num_posti_valido	Il numero di posti di una postazione deve essere ovviamente positivo. Non deve essere possibile inserire postazioni con numero di posti negativi, e tale vincolo rispetta ciò.
ora_valida	Vincolo che impedisce che ci sia una sovrapposizione temporale e che quindi l'ora di fine di utilizzo di uno strumento sia precedente alla sua ora di inizio.
email_valido/email_validot	Quando viene inserita una mail da parte di un utente o di un tecnico questo vincolo controllo se essa è valida, ovvero se prima del carattere @ ci sia almeno un altro carattere.

Capitolo 3

PROGETTAZIONE LOGICA

3.1 Schema logico

Dopo la ristrutturazione possiamo passare alla fase di **progettazione logica** e quindi creare il nostro **schema logico**. Il modello utilizzato per modellare la realtà è il modello relazionale che prevede un unico meccanismo di strutturazione dei dati, che come dice il nome, è la **relazione**, essa serve per rappresentare sia le classi (insiemi di entità) sia le associazioni che intercorrono tra esse.

3.1.1 Traduzione delle classi

La rappresentazione delle classi con i rispettivi attributi *a1,a2,…an* avviene tramite una relazione R(*a1,a2,..an*) che per convenzione ha lo stesso nome dell'entità.

3.1.2 Traduzione delle associazioni

Per rappresentare le associazioni bisogna tener conto della cardinalità, infatti sono stati seguiti i seguenti punti:

- Per le associazioni di tipo 1:N (uno a molti) o N:1 (molti ad uno) si inserisce come chiave esterna la chiave primaria dell'insieme di entità del lato a uno nella relazione che rappresenta l'insieme di entità del lato a molti
- Per le associazioni di tipo 1:1 (uno a uno) basta inserire come chiave esterna la chiave primaria di uno qualunque dei due insiemi di entità.
- Per le associazioni di tipo N:N (molti a molti) si crea una nuova relazione avente come attributi le chiavi primarie delle due entità coinvolte dall'associazione ed eventuali attributi dell'associazione vengono inseriti in questa nuova relazione.

3.1.2 Schema logico

1. **Laboratorio** (CodL, descrizione, orario_Apertura, orario_chiusura, num_tecnici, num_postazioni)
   Sede (CodS, nome, indirizzo, responsabile)
2. **Appartenenza** (***CodL_fk, CodS_fk***)
3. **Tecnico** (Matricola, nome, cognome, codfiscale, telefono, emai, ***CodL_fk***)
4. **Postazione** (CodPostazione, num_posti, ***CodL_fk***)
5. **Strumento** (CodStrumento, caratteristiche_tecniche, descrizione, tempoMaxUso, ***CodPostazione_fk, CodSede_fk***)
6. **Utente** (Username, email, pw)
7. **Prenotazione** (CodPrenotazione, data_prenotazioneS, ora_prenotazioneS, tempo_utilizzoS, ***Username_fk***, ***CodStrumento_fk***)
8. **Occupato** (***CodPostazione_fk***, ***Username_fk***)
9. **Utilizzato** (***Username_fk***, ***CodStrumento_fk***, datautilizzo, oraInizio, oraFine)

Le chiavi primarie sono sottolineate mentre le chiavi esterne sono identificate _fk e sono in grassetto.

Capitolo 4

PROGETTAZIONE FISICA

4.1 Definizione delle tabelle e dei vincoli

L'ultima fase è la progettazione fisica. Quest'ultima fase consiste nella traduzione dello schema logico dei dati in uno **schema fisico dei dati** contenente le definizioni delle tabelle e dei relativi vincoli di integrità.

4.1.1 Definizione della tabella LABORATORIO

```
CREATE TABLE IF NOT EXISTS Laboratorio
(
   CodL VARCHAR (3) PRIMARY KEY,
   descrizione VARCHAR (2000) NOT NULL,
   orario_Apertura TIME NOT NULL,
   orario_Chiusura TIME NOT NULL,
   num_tecnici SMALLINT NOT NULL,
   num_postazioni SMALLINT NOT NULL
);
```

4.1.2 Definizione della tabella SEDE

```
CREATE TABLE IF NOT EXISTS Sede
(
   CodS SERIAL PRIMARY KEY,
   nome VARCHAR (35) NOT NULL,
   indirizzo VARCHAR (60) NOT NULL,
   responsabile VARCHAR (70) NOT NULL
);
```

4.1.3 Definizione della tabella APPARTENENZA

```
CREATE TABLE IF NOT EXISTS Appartenenza
(
   CodL_fk VARCHAR (3) REFERENCES Laboratorio (CodL)
      on delete cascade
      on update cascade,
   CodS_fk INTEGER REFERENCES Sede (CodS)
      on delete cascade
      on update cascade
);
```

4.1.4 Definizione della tabella TECNICO

```
CREATE TABLE IF NOT EXISTS Tecnico
(
   Matricola VARCHAR (10) PRIMARY KEY,
   nome VARCHAR(20) NOT NULL,
   cognome VARCHAR(20) NOT NULL,
   codfiscale VARCHAR(16) UNIQUE,
   telefono VARCHAR(10),
   email VARCHAR(30) UNIQUE,
   CodL_fk VARCHAR(3) REFERENCES Laboratorio(CodL)
      On delete cascade
      On update cascade
);
```

4.1.5 Definizione della tabella POSTAZIONE

```
CREATE TABLE IF NOT EXISTS Postazione
(
   CodPostazione VARCHAR(3) PRIMARY KEY,
   num_posti SMALLINT NOT NULL,
   CodL_fk VARCHAR(3) REFERENCES Laboratorio(CodL)
      On delete cascade
      On update cascade
);
```

4.1.6 Definizione della tabella STRUMENTO

```
CREATE TABLE IF NOT EXISTS Strumento
(
   CodStrumento SERIAL PRIMARY KEY,
   caratteristiche_tecniche VARCHAR(1000) NOT NULL,
   descrizione VARCHAR(200) NOT NULL,
   tempoMaxUso TIME NOT NULL,
   CodPostazione_fk VARCHAR(3) REFERENCES Postazione(CodPostazione)
      On delete cascade
      On update cascade,
   CodSede_fk INTEGER REFERENCES Sede(CodS)
      on delete cascade
      on update cascade
);
```

4.1.7 Definizione della tabella UTENTE

```
CREATE TABLE IF NOT EXISTS Utente
(
   Username VARCHAR(20) PRIMARY KEY,
   email VARCHAR(30) UNIQUE,
   pw VARCHAR(30) NOT NULL
);
```

4.1.8 Definizione della tabella PRENOTAZIONE

```
CREATE TABLE IF NOT EXISTS Prenotazione
(
   CodPrenotazione SERIAL PRIMARY KEY,
   data_prenotazioneS DATE NOT NULL,
   ora_prenotazioneS TIME NOT NULL,
   tempo_utilizzoS TIME NOT NULL,
   Username_fk VARCHAR(20) REFERENCES Utente(Username)
      On delete cascade
      On update cascade,
   CodStrumento_fk INTEGER REFERENCES Strumento(CodStrumento)
      On delete cascade
      On update cascade
);
```

4.1.9 Definizione della tabella OCCUPATO

```
CREATE TABLE IF NOT EXISTS Occupato
(
   Username_fk VARCHAR(20) REFERENCES Utente(Username)
      On delete cascade
      On update cascade,
   CodPostazione_fk VARCHAR(3) REFERENCES Postazione(CodPostazione)
      On delete cascade
      On update cascade
);
```

4.2.0 Definizione della tabella UTILIZZATO

```
CREATE TABLE IF NOT EXISTS Utilizzato
(
   Username_fk VARCHAR(20) REFERENCES Utente(Username)
      On delete cascade
      On update cascade,
   CodStrumento_fk INTEGER REFERENCES Strumento(CodStrumento)
      On delete cascade
      On update cascade,
   datautilizzo DATE,
   oraInizio TIME,
   oraFine TIME
);
```

4.2.1 Definizione dei vincoli relativi alla tabella UTILIZZATO

```
ALTER TABLE utilizzato
ADD CONSTRAINT ora_valida
CHECK (orafine >= orainizio)
```

👁

*Si può notare che ci sono dei vincoli riguardanti l'operazione di cancellazione e di modifica da una tabella Y referenziata da una tabella X di un elemento. I tipi di vincoli da noi implementati **ON DELETE CASCADE** e **ON UPDATE CASCADE** consentono rispettivamente la cancellazione e la modifica a cascata di tutti gli elementi di X in cui è presente come valore di chiave esterna la chiave dell'elemento di Y che si vuole cancellare o modificare.*

4.3 Definizione delle Funzioni, Procedure e Trigger

4.3.1 Verifica della disponibilità dello strumento scelto prima di una prenotazione.

Il trigger "CHECK_PRENOTAZIONE" chiamato dalla funzione controlla se uno strumento è già stato prenotato per l'ora specificata nella nuova riga che sta per essere inserita nella tabella "PRENOTAZIONE". Se lo strumento è già stato prenotato, la funzione "gia_prenotato()" genera un'eccezione con un messaggio di errore. Altrimenti, la nuova riga viene inserita nella tabella.

```
CREATE OR REPLACE FUNCTION gia_prenotato ()
   RETURNS TRIGGER as $$
  DECLARE
   tempo_utilizzo_sec integer;
   contatore INTEGER;
BEGIN
   tempo_utilizzo_sec = extract(epoch from NEW.tempo_utilizzoS);

   SELECT count(*) INTO contatore
   FROM prenotazione

   -- LE DATE DI PRENOTAZIONE E LO STRUMENTO DEVONO CORRISPONDERE
   WHERE data_prenotazioneS = NEW.data_prenotazioneS
         AND codstrumento_fk = NEW.codstrumento_fk AND
         (ora_prenotazioneS, ora_prenotazioneS + interval '1 second' * extract(epoch from
prenotazione.tempo_utilizzoS))
          OVERLAPS (NEW.ora_prenotazioneS, NEW.ora_prenotazioneS + interval '1
second' * tempo_utilizzo_sec);
   IF (contatore > 0) THEN
      RAISE EXCEPTION 'Strumento già prenotato per questo orario';
   END IF;

            RETURN NEW;
END

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER CHECK_PRENOTAZIONE
BEFORE INSERT on PRENOTAZIONE
FOR EACH ROW
EXECUTE FUNCTION gia_prenotato();
```

4.3.2 Controllo che la prenotazione non sia effettuata su date passate

Il trigger controlla che la data e l'ora di una nuova prenotazione non siano passate e che la prenotazione sia per l'anno corrente. In caso contrario, viene generata un'eccezione.

```
CREATE OR REPLACE FUNCTION no_prenotazioni_date_passate()
RETURNS TRIGGER AS $$

DECLARE

data_di_oggi DATE = current_date;
ora_di_oggi TIME = current_time;
anno_corrente INTEGER = extract(year from data_di_oggi);
anno_prenotazione INTEGER = extract(year from NEW.data_prenotazioneS);

BEGIN

IF(NEW.data_prenotazioneS < data_di_oggi) THEN
RAISE EXCEPTION 'Errore, stai provando a prenotare uno strumento per una data
passata';
END IF;

IF(NEW.data_prenotazioneS = data_di_oggi AND NEW.ora_prenotazioneS < ora_di_oggi)
THEN
RAISE EXCEPTION 'Errore, stai provando a prenotare uno strumento per un''ora passata';
END IF;

IF((anno_prenotazione - anno_corrente) > 1 ) THEN
RAISE EXCEPTION 'Non è possibile prenotare uno strumento per un anno il quale non sia
quello corrente';
END IF;
RETURN NEW;

END

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER CONTROLLO_DATE_PASSATE
BEFORE INSERT on PRENOTAZIONE
FOR EACH ROW
EXECUTE FUNCTION no_prenotazioni_date_passate()
```

4.3.2 Incremento numero di tecnici all'inserimento di un nuovo tecnico

Il trigger incrementa num_tecnici nella tabella Laboratorio dopo che è stato inserito un nuovo tecnico.

```
CREATE OR REPLACE FUNCTION incremento_num_tecnici()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE Laboratorio SET num_tecnici = num_tecnici + 1
   WHERE CodL = NEW.CodL_fk;
   RETURN NEW;
END

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER incremento_tecnici
AFTER INSERT ON Tecnico
FOR EACH ROW
EXECUTE FUNCTION incremento_num_tecnici();
```

4.3.2 Controllo disponibilità posti nella postazione

Questo trigger controlla se è possibile effettuare una prenotazione per uno strumento in una determinata postazione verificando se il numero di prenotazioni che si sovrappongono allo stesso intervallo di tempo (cioè quelle che utilizzano la stessa postazione e lo stesso strumento) è inferiore al numero massimo di posti disponibili nella postazione. Se il numero di prenotazioni supera la capienza massima della postazione, viene sollevata un'eccezione con un messaggio di errore.

```
CREATE OR REPLACE FUNCTION max_postazioni()
RETURNS TRIGGER AS $$
DECLARE

lim_post INTEGER;
num_prenotaz INTEGER;
postaz_pren VARCHAR(10);

BEGIN

/*Selezioniamo il numero massimo di posti per la postazione in cui si trova
lo strumento selezionato nella prenotazione*/
SELECT p.num_posti
INTO lim_post
FROM postazione as p JOIN strumento as s ON p.codpostazione = s.codpostazione_fk
WHERE p.codpostazione = s.codpostazione_fk AND s.codstrumento =
NEW.codstrumento_fk;

/*Selezioniamo la postazione in cui è presente lo strumento selezionato
```

```
nella prenotazione*/

SELECT p.codpostazione
INTO postaz_pren
FROM postazione as p JOIN strumento as s ON p.codpostazione = s.codpostazione_fk
WHERE s.codstrumento = NEW.codstrumento_fk;

/*Utilizzando count ci ricaviamo il numero di prenotazioni relative a quella
postazione che si sovrappongono al tempo di utilizzo dello strumento, scelto
nella prenotazione. Per verificare ciò utilizziamo la funzione overlap*/

SELECT count(pr.codprenotazione)
 INTO num_prenotaz
 FROM prenotazione as pr JOIN strumento as s ON s.codStrumento = pr.codstrumento_fk
      JOIN postazione as p ON p.codpostazione = s.codpostazione_fk
      WHERE pr.data_prenotaziones = NEW.data_prenotaziones
      AND (pr.ora_prenotaziones, pr.ora_prenotaziones +
      interval '1 second' * extract(epoch from pr.tempo_utilizzos))
      OVERLAPS
      (NEW.ora_prenotaziones, NEW.ora_prenotaziones
      + interval '1 second' * extract(epoch from NEW.tempo_utilizzos))
      AND p.codpostazione = postaz_pren;

-- Verifichiamo che le prenotazioni presenti siano inferiori della capienza
IF (lim_post <= num_prenotaz) THEN

-- Messaggio di errore nel caso il numero venga superato
RAISE EXCEPTION 'La postazione per l'ora scelta è piena,
provare con un altro orario o giorno. Postazione: %, Capienza: %', postaz_pren, lim_post;
END IF;

RETURN NEW;

END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER max_post
BEFORE INSERT ON prenotazione
FOR EACH ROW
EXECUTE FUNCTION max_postazioni();
```

4.3.3 Inserimento automatico della prenotazione nella tabella Utilizzato

Questo trigger viene attivato dopo l'inserimento di una nuova prenotazione e ha lo scopo di inserire, aggiornare o eliminare una prenotazione dello strumento nella tabella Utilizzato. Vengono inseriti i dati dell'utente, del codice dello strumento, della data e dell'ora di inizio e fine dell'utilizzo, che vengono ricavati dalle informazioni della prenotazione appena inserita.

```
CREATE OR REPLACE FUNCTION inserisci_utilizzo()
RETURNS TRIGGER AS $$
BEGIN

IF (TG_OP= 'INSERT') THEN
  INSERT INTO Utilizzato (Username_fk, CodStrumento_fk, datautilizzo, oraInizio, oraFine)
  VALUES (NEW.Username_fk, NEW.CodStrumento_fk, NEW.data_prenotazioneS,
NEW.ora_prenotaziones, NEW.ora_prenotaziones + interval'1 second'* extract(epoch from
new.tempo_utilizzos));
  RETURN NEW;

ELSE IF(TG_OP='UPDATE') THEN
    UPDATE Utilizzato SET
    datautilizzo = NEW.data_prenotazioneS,
    oraInizio = NEW.ora_prenotaziones,
    oraFine = NEW.ora_prenotaziones + interval '1 second' * extract(epoch from
NEW.tempo_utilizzos)
  WHERE Username_fk = NEW.Username_fk AND CodStrumento_fk =
NEW.CodStrumento_fk;
  RETURN NEW;

ELSE IF (TG_OP = 'DELETE') THEN
    DELETE FROM Utilizzato
    WHERE Username_fk = OLD.Username_fk AND CodStrumento_fk =
OLD.CodStrumento_fk;
    RETURN OLD;
    END IF;
  END IF;
 END IF;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER inserisci_utilizzo_trigger
AFTER INSERT OR UPDATE OR DELETE  ON Prenotazione
FOR EACH ROW
 -- Può essere attivato solo da una query esterna e non da un altro trigger
WHEN (pg_trigger_depth() = 0)
      EXECUTE FUNCTION inserisci_utilizzo();
```

4.3.4 Controllo limite di prenotazioni consentite da un utente

Questo è un trigger che esegue una funzione chiamata "verifica_limite_prenotazioni" ogni volta che viene inserita o aggiornata una riga nella tabella "Prenotazione".
La funzione dichiara un cursore che seleziona tutte le prenotazioni con la stessa data e lo stesso utente della nuova riga inserita o aggiornata.
In un ciclo "LOOP", la funzione conta il numero di prenotazioni trovate e se questo numero supera 5, viene generata un'eccezione che impedisce l'ulteriore inserimento o aggiornamento, in caso contrario restituisce la nuova riga.

```
CREATE OR REPLACE FUNCTION verifica_limite_prenotazioni()
RETURNS TRIGGER AS $$
DECLARE
  cur CURSOR FOR SELECT CodPrenotazione, data_prenotazioneS, Username_fk
  FROM Prenotazione
  WHERE data_prenotazioneS = NEW.data_prenotazioneS AND Username_fk =
NEW.Username_fk;
  contatore INTEGER;
  prenotazioneU cur%TYPE;
BEGIN
  contatore := 0;
  OPEN cur;
  LOOP
    FETCH cur INTO prenotazioneU ;
    IF NOT FOUND THEN
      EXIT;
    END IF;
    contatore := contatore + 1;
    IF contatore >= 5 THEN
      RAISE EXCEPTION  'Non è possibile effettuare più di 5 prenotazioni in un giorno';
    END IF;
  END LOOP;
  CLOSE cur;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER verifica_limite_prenotazioni_trigger
AFTER INSERT OR UPDATE ON Prenotazione
FOR EACH ROW
      EXECUTE FUNCTION verifica_limite_prenotazioni();
```

4.3.5 Controllo orari del laboratorio prima di effettuare una prenotazione

Il trigger controlla se l'orario della nuova prenotazione è compreso tra l'orario di apertura e chiusura del laboratorio. Se l'orario non rientra in questi limiti, viene generata un'eccezione con un messaggio di errore, ovvero il laboratorio è chiuso.

```
CREATE OR REPLACE FUNCTION check_ora_prenotazione()
RETURNS TRIGGER AS $$
DECLARE
   ora_apertural TIME;
   ora_chiusural TIME;
BEGIN
   SELECT orario_apertura,orario_chiusura
   INTO ora_apertural,ora_chiusural
   FROM laboratorio;

IF(ora_apertural > NEW.ora_prenotazioneS OR ora_chiusural <NEW.ora_prenotazioneS)
THEN
  RAISE EXCEPTION 'Lo strumento non può essere prenotato per quell''ora.Il laboratorio è
chiuso.';
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER ora
BEFORE INSERT ON prenotazione
FOR EACH ROW
EXECUTE FUNCTION check_ora_prenotazione();
```

4.3.6 Controllo tempo massimo di utilizzo per lo strumento prenotato

Questo trigger controlla se il tempo di utilizzo selezionato per una prenotazione è maggiore del tempo massimo consentito per lo strumento selezionato. Se il tempo di utilizzo supera il tempo massimo consentito, viene generata un'eccezione che indica il nome dello strumento e il tempo massimo consentito. Il trigger viene attivato dopo l'inserimento o l'aggiornamento di una prenotazione.

```
CREATE OR REPLACE FUNCTION tempo_superato()
RETURNS TRIGGER AS $$
DECLARE
   tempo_max TIME;
   strumentod varchar(200);
BEGIN

   -- selezione descrizione strumento per codice errore.
SELECT s.descrizione
   INTO strumentod
```

```
FROM strumento AS s JOIN prenotazione AS pr ON
s.codStrumento=NEW.codStrumento_fk;

 SELECT tempoMaxUso
   INTO tempo_max
 FROM strumento as s JOIN prenotazione JOIN pr ON s.codStrumento=pr.codStrumento_fk
 WHERE s.codStrumento=NEW.codStrumento_fk;

   IF tempo_max< NEW.tempo_utilizzoS THEN
      RAISE EXCEPTION 'Il tempo scelto è MAGGIORE del tempo massimo consentito per
questo strumento % che è %', strumentod, tempo_max;
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER controllo_tempo
AFTER INSERT OR UPDATE ON prenotazione
FOR EACH ROW
EXECUTE FUNCTION tempo_superato();
```

4.3.7 Cancellazione automatica prenotazione passate

Questo Trigger viene azionato ogni volta che viene inserita una nuova prenotazione ed elimina dalla tabella prenotazione, le prenotazioni più vecchie di due settimane.

```
CREATE OR REPLACE FUNCTION cancella_prenotazioni_scadute()
RETURNS TRIGGER AS $$
DECLARE
   CURSOR cur FOR SELECT codprenotazione FROM prenotazione
WHERE data_prenotaziones < NOW() - INTERVAL '2 WEEK';
codcorrente INTEGER;

BEGIN
   OPEN cur;
     LOOP
       FETCH FROM cur INTO codcorrente;
       EXIT WHEN NOT FOUND;
       DELETE FROM prenotazione WHERE codprenotazione=codcorrente;
   END LOOP;
   CLOSE cur;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER elimina_prenotazioni_scadute
```

```
AFTER INSERT ON prenotazione
FOR EACH ROW
      EXECUTE FUNCTION cancella_prenotazioni_scadute();
```

4.3.8 Controllo della durata minima relativa ad uno strumento

Questo trigger è stato creato per controllare che la durata minima di utilizzo per lo strumento inserita nella prenotazione sia superiore ad 1 minuto. Se la durata è inferiore a 1 minuto, il trigger lancerà un'eccezione con un messaggio di errore. Il trigger viene attivato prima che la nuova prenotazione venga inserita nel database.

```
CREATE OR REPLACE FUNCTION check_minuti()
RETURNS TRIGGER AS $$
BEGIN
  IF NEW.tempo_utilizzoS <= INTERVAL '1 minute' THEN
   RAISE EXCEPTION 'Durata minima di utilizzo deve essere superiore a 1 minuto.';
   END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER minuti
BEFORE INSERT ON prenotazione
FOR EACH ROW
EXECUTE FUNCTION check_minuti();
```

4.3.9 Funzione per riepilogo in base mensile ed annuale dello strumento

Questa funzione riceve tre parametri in ingresso: il codice di uno strumento, il mese e l'anno di interesse. In base a questi parametri, la funzione restituisce una tabella con il riepilogo dell'utilizzo dello strumento nel periodo specificato, come la durata totale di utilizzo e l'utente che l'ha utilizzato di più.

```
-- Creiamo una VIEW riepilogo per ottenere una tabella di appoggio con le informazioni di cui necessitiamo
CREATE OR REPLACE VIEW riepilogo AS
   SELECT codStrumento, Anno, Mese,
          sum(durata) as Durata_totale, Utente

-- Raggruppiamo le informazioni per codice dello strumento, mese e anno

FROM (SELECT codstrumento_fk as codStrumento, extract('year' from datautilizzo) as Anno, extract('month' from datautilizzo) as Mese, orafine - interval '1 second'*extract (epoch from orainizio) AS Durata, username_fk as Utente
FROM utilizzato) as T
GROUP BY codStrumento, anno, mese, utente;
```

```
CREATE OR REPLACE FUNCTION riepilogo_strumenti(codS IN INTEGER, mesein IN
INTEGER, annoin IN INTEGER)

-- La funzione ci deve ritornare una tabella che ha la struttura della tabella riepilogo

RETURNS TABLE(codstrumento riepilogo.codstrumento%type,anno riepilogo.anno%type,
mese riepilogo.mese%type,durata_totale riepilogo.durata_totale%type,utentemax
riepilogo.utente%type ) AS $$

DECLARE
    utente_max riepilogo.utente%TYPE;

BEGIN
/*Questa select recupera l'utente che ha utilizzato di più uno specifico strumento
in un determinato mese e anno, o in generale se non viene specificato il parametro del mese
e/o anno.

SELECT utente
   INTO utente_max
FROM riepilogo as r
WHERE r.codstrumento = cods AND (r.mese=mesein OR mesein IS NULL) AND
(r.anno=annoin OR annoin IS NULL)
GROUP BY utente
ORDER BY MAX(r.durata_totale) DESC;

SELECT *
   INTO codStrumento, anno, mese, durata_totale, utentemax
FROM riepilogo AS r
WHERE r.utente=utente_max AND r.codstrumento = cods AND (r.mese=mesein OR mesein
IS NULL) AND (r.anno=annoin OR annoin IS NULL);


IF NOT FOUND THEN
    RAISE EXCEPTION 'Lo strumento % non è stato utilizzato il mese % dell''anno %',
cods,mesein,annoin;
ELSE
       RETURN NEXT;
END IF;
END;
$$ LANGUAGE plpgsql;
```

4.3.10 Funzione per riepilogo calendario prenotazioni

La funzione "calendario_prenotazioni_s" restituisce una tabella di righe contenenti informazioni sul calendario delle prenotazioni. La funzione accetta due argomenti: "descrizione_in" e "sede_in".
La funzione utilizza una struttura LOOP per iterare attraverso la vista "Calendario" e selezionare le righe che corrispondono ai criteri di ricerca specificati dagli argomenti della funzione.
Infine, la funzione restituisce una tabella di righe che mostra il calendario in base agli argomenti passati alla funzione.

```
-- Creiamo una VIEW per ottenere la struttura del calendario richiesta
CREATE OR REPLACE VIEW calendario
(codice_strumento,descrizione_strumento,nome_sede,data_prenotazione,ora_inizio,ora_fin
e) AS
(
   SELECT s.codStrumento, s.descrizione, se.nome, pr.data_prenotazioneS,
pr.ora_prenotazioneS,
      (pr.ora_prenotazioneS + interval '1 second' * extract(epoch from pr.tempo_utilizzoS))
      FROM strumento AS s    JOIN sede AS se ON s.codSede_fk = se.cods
          LEFT JOIN prenotazione AS pr ON pr.codstrumento_fk = s.codstrumento
)

--LEFT JOIN per includere tutte le informazioni sullo strumento, anche se non è stato
prenotato.
--In questo modo, le colonne della tabella Calendario che riguardano la prenotazione
avranno un valore NULL se lo strumento non è stato prenotato.

CREATE OR REPLACE FUNCTION calendario_prenotazioni_s (IN descrizione_in
VARCHAR(200), IN sede_in VARCHAR(50))
RETURNS TABLE (codice_strumento INTEGER, descrizione_strumento
VARCHAR(200),nome_sede VARCHAR(50), data_prenotazione DATE, ora_inizio TIME,
ora_fine TIME) AS $$

DECLARE
   riga_tab_ritorno calendario%ROWTYPE;
BEGIN
   FOR riga_tab_ritorno IN
      SELECT *
      FROM calendario as ca
      -- Con ILIKE l'istruzione non è casesensitive
      WHERE  (ca.descrizione_strumento ILIKE '%' ||descrizione_in|| '%' OR
descrizione_in='') AND (ca.nome_sede ILIKE '%' ||sede_in|| '%' OR sede_in = '')
      ORDER BY ca.data_prenotazione DESC

   LOOP
         codice_strumento := riga_tab_ritorno.codice_strumento;
         descrizione_strumento := riga_tab_ritorno.descrizione_strumento;
```

```
            nome_sede := riga_tab_ritorno.nome_sede;
            data_prenotazione := riga_tab_ritorno.data_prenotazione;
            ora_inizio := riga_tab_ritorno.ora_inizio;
            ora_fine := riga_tab_ritorno.ora_fine;
            RETURN NEXT; --restituisce riga per riga il risultato di una query
    END LOOP;
END;
    $$ LANGUAGE plpgsql;
```

4.4 Implementazione dei Vincoli

4.4.1 Implementazione del vincolo *orariol*

```
ALTER TABLE laboratorio
ADD CONSTRAINT orariol
CHECK (orario_apertura='08:00:00' AND orario_chiusura='20:00:00');
```

4.4.2 Implementazione del vincolo *codicefiscale_valido*

```
ALTER TABLE TECNICO
ADD CONSTRAINT codicefiscale_valido
CHECK(length (codfiscale)=16);
```

4.4.3 Implementazione del vincolo *telefono_valido*

```
ALTER TABLE TECNICO
ADD CONSTRAINT telefono_valido
CHECK (length (telefono)=10);
```

4.4.4 Implementazione del vincolo *username_valido*

```
ALTER TABLE utente
ADD CONSTRAINT username_valido
CHECK (username NOT LIKE '%-%' AND username NOT LIKE '%"%' AND username NOT
LIKE '%?%' AND LENGTH(username) >= 10);
```

4.4.5 Implementazione del vincolo *num_tecnici_valido*

```
ALTER TABLE laboratorio
ADD CONSTRAINT num_tecnici_valido
CHECK (num_tecnici BETWEEN 0 AND 100);
```

4.4.6 Implementazione del vincolo *num_postazioni_valido*

```
ALTER TABLE laboratorio
ADD CONSTRAINT num_postazioni_valido
CHECK (num_postazioni BETWEEN 0 and 50);
```

4.4.7 Implementazione del vincolo *email_validot*

```
ALTER TABLE tecnico
ADD CONSTRAINT email_validot
CHECK ( position('@' IN email) > 1);
```

4.4.8 Implementazione del vincolo *username_valido*

```
ALTER TABLE utente
ADD CONSTRAINT username_valido
CHECK (username NOT LIKE '%-%' AND username NOT LIKE '%"%' AND username NOT
LIKE '%?%' AND LENGTH(username) >= 10);
```

4.4.9 Implementazione del vincolo *email_valido*

```
ALTER TABLE utente
ADD CONSTRAINT email_valido
CHECK ( position('@' IN email) > 1);
```

4.5.0 Implementazione del vincolo *num_posti_valido*

```
ALTER TABLE postazione
ADD CONSTRAINT num_posti_valido
CHECK (num_posti>=0);
```

4.5.1 Implementazione del vincolo *utilizzato*

```
ALTER TABLE utilizzato
ADD CONSTRAINT ora_valida
CHECK (orafine>=orainizio)
```

CAPITOLO 5

MANUALE D’USO

5.1 Popolamento con Dati di Esempio

Questo script popola il database con dati fittizi, utilizzare su un database vuoto per evitare conflitti, è necessario conoscere la progettazione fisica al Capitolo 4 per ottenere un inserimento coerente.

```
-- Popolamento di LABORATORIO

INSERT INTO laboratorio
VALUES
('L1','Nel Laboratorio di Informatica collocato nell'ex aula 20 al primo della Palazzina Uomini sono installati 16 personal computer ed un pc come postazione docente. Si tratta di 5 Asus EEepc e di 12 HP 6200 Pro.
I PC operano in ambiente Windows 7. In tutti i PC sono installati Microsoft Office Professional 2010, Mozilla Firefox, Google Chrome, Microsoft Security Essentials',
'08:00','20:00',20,12),
('L2','Il laboratorio chimico è un locale adibito a indagini chimiche di tipo analitico, e in genere a preparazioni a scala di laboratorio di sostanze chimiche e miscele di esse.','08:00','20:00',2,4),
('L3','Laboratorio multidisciplinare (Biologia,Scienze della Terra) dove si possono osservare oggetti naturali ed eseguire, lavorando in piccoli gruppi, esperimenti di semplice attuazione che permettono la verifica di alcuni fenomeni naturali chimici o biologici.','08:00','20:00',17,22),
('L4','Laboratorio medico: una struttura dove vengono eseguite analisi e test diagnostici su campioni di sangue, urine, tessuti o altri materiali biologici per aiutare nella diagnosi e nel trattamento delle malattie.','08:00','20:00',8,4),
('L5','Laboratorio Ematologix: un laboratorio ematologico moderno e all avanguardia.','08:00','20:00',10,6),
('L6','Allergie & Analisi: un laboratorio specializzato in allergologia.','08:00','20:00',34,30),
('L7','Città del Lab: Un idea creativa e moderna per un centro in cui sono riunite diverse tipologie di laboratorio.','08:00','20:00',2,1),
('L8','Mortaio Tech: Una proposta eccentrica per il nome di un laboratorio moderno che prende spunto dallo strumento più tradizionale, il mortaio.','08:00','20:00',45,50),
('L9','Vitaedoc Laboratori: Un bel nome altisonante per un laboratorio medico. La parola "vitae" in latino aggiunge un tocco di classe.','08:00','20:00',18,9),
('L10','Triglives Laboratorio: Una parola inventata che unisce due degli esami che di solito vengono fatti nei laboratori: i trigliceridi e la Ves.','08:00','20:00',11,32)
```

-- Popolamento di SEDE

```
INSERT INTO Sede (nome,indirizzo,responsabile)
VALUES
('CERN','Via regina Napoli 78','Luca Rossi'),
('Lawrence Berkeley','Via Giulio Caserta 23','Marco Bianchi'),
('Max Planck','Via piccioli Palermo 44','Alessandro Verdi'),
('Institut Pasteur','Via Augusteo Roma 30','Francesca Neri'),
('Nazionale Argonne','Via Giovanni II Roma 56','Giovanni Ferrari'),
('Brookhaven','Via Paolo Torino 99','Claudio Romano'),
('Oak Ridge','Vai Rosane Milano 103','Antonio Esposito'),
('Synchrotron Light Source II','Via Trento Napoli 9','Stefano Ricci'),
('Fraunhofer','Via Starza Genova 11','Daniela Moretti'),
('Salk Institute','Via Rollo Ostia 22','Matteo Marino'),
('Scripps Research','Via Gianturco Roma 98','Rosa Nuvola'),
('RIKEN','Via Luigi Beneduce Sant Anastasia 68','Marco Gialli'),
('Weizmann Institute','Via Ombra Milano 49','Giulio Quaranta'),
('Commonwealth','Via Claudio Napoli 21','Marta Rossi'),
('Turing','Via Est Liguria 14','Roberta Beneduce'),
('Ronaldo','Via Colle Somma 4','Vincenzo Capasso'),
('Von Neumann','Via Bakù¹ Scampia 33','Luca Cardone');
```

-- Popolamento di TECNICO

```
INSERT INTO Tecnico (Matricola, nome, cognome, codFiscale, telefono, email, CodL_fk)
VALUES
('N860001', 'Andrea', 'Rossi', 'RSSAND99A01H501F', '3331234567',
'andrea.rossi@email.com', 'L1'),
('N860002', 'Luca', 'Cardone', 'CRDGLC01E21F839Y', '3896567353',
'luca.cardone@email.com', 'L1'),
('N860003', 'Marco', 'Chimenti', 'CMTMRC99A01H501F', '3331435567',
'marco.chimenti@email.com', 'L1'),
('C340001', 'Luca', 'Bianchi', 'BNCLCU99L01H501G', '3332305678',
'luca.bianchi@email.com', 'L2'),
('C340002', 'Patrizio', 'Coda', 'PTRCOD99L01H501G', '3332345670',
'patrizio.coda@email.com', 'L2'),
('M180001', 'Giovanni', 'Verdi', 'VRDGNN99A01H501H', '3333456789',
'giovanni.verdi@email.com', 'L3'),
('M180002', 'Giovanni', 'Rossi', 'RSSGNN99A01H501H', '3333436789',
'giovanni.rossi@email.com', 'L3'),
('M180003', 'Ludovica', 'Palla', 'PLLLDV99A01H501H', '3233456789',
'ludovica.palla@email.com', 'L3'),
('M180004', 'Pasquale', 'Turo', 'TROPSL99A01H501H', '3331456789',
'pasquale.turo@email.com', 'L3'),
('M300001', 'Francesca', 'Neri', 'NRIFRA99A01H501I', '3334678901',
'francesca.neri@email.com', 'L4'),
```

('M300002', 'Checco', 'Zalone', 'ZALCCO99A01H501I', '3334578901',
'checco.zalone@email.com', 'L4'),
('E820001', 'Giovanni', 'Ferrari', 'FRRGNN99A01H501J', '3335678901',
'giovanni.ferrari@email.com', 'L5'),
('E820002', 'Giovanna', 'Lupini', 'LPNGNA99A01H501J', '3335678901',
'giovanna.lupini@email.com', 'L5'),
('A610001', 'Claudio', 'Romano', 'RMNCLD99A01H501K', '3336789012',
'claudio.romano@email.com', 'L6'),
('C700001', 'Antonio', 'Esposito', 'SPOANT99A01H501L', '3337890123',
'antonio.esposito@email.com', 'L7'),
('C700002', 'Marco', 'Totti', 'TTIMRC99A01H501L', '3334890123', 'marco.totti@email.com',
'L7'),
('M000001', 'Stefano', 'Ricci', 'RCCSTF99A01H501M', '3338901234',
'stefano.ricci@email.com', 'L8'),
('V100001', 'Daniela', 'Moretti', 'MRTDNL99A01H501N', '3339567890',
'daniela.moretti@email.com', 'L9'),
('T100001', 'Matteo', 'Marino', 'MRNMTO99A01H501O', '3340789012',
'matteo.marino@email.com', 'L10');

-- Popolamento di STRUMENTO

INSERT INTO Strumento (caratt_tecniche, descrizione, TempoMaxUso, CodPostazione_fk)
VALUES
('Intel Core i7, 16 GB RAM, 1 TB SSD', 'Computer Desktop', 01:30:00, 'P1'),
('Dell U2415', 'Monitor 24 pollici', 01:30:00, 'P1'),
('Apple Magic Mouse 2', 'Mouse wireless', 01:30:00, 'P1'),
('Apple Keyboard', 'Tastiera Wireless', 01:30:00, 'P11'),
('Raspberry Pi 4', 'Mini computer single-board', 01:30:00, 'P11'),
('Capacità 500 mL, Precisione ± 0,5 mL', 'Pipetta automatica', 00:30:00, 'P2'),
('Intervallo di misura 0-100 °C, Precisione ± 0,1 °C', 'Termometro digitale', 00:30:00, 'P15'),
('Intervallo di misura pH 0-14, Precisione ± 0,01 pH', 'pH-metro', 00:30:00, 'P16'),
('Volume max 100 mL, Precisione ± 0,5 mL', 'Buretta' ,00:30:00, 'P16'),
('Ingrandimento 40-1600x, Illuminazione a LED', 'Microscopio elettronico', 00:50:00, 'P3'),
('Ingrandimento 40-400x, Illuminazione a LED', 'Microscopio ottico', 00:50:00, 'P17'),
('Risoluzione 2D/4D, Lunghezza onda 2-13 MHz', 'Ecografo', 02:00:00, 'P4'),
('Risoluzione 16-slice, Tecnologia multidetettore', 'Tomografo a raggi X', 02:00:00, 'P14'),
('Velocità di scansione 20 campioni/min, Precisione ± 2%', 'Analizzatore di conta dei globuli
rossi', 0:10:00, 'P5'),
('Precisione ± 1 µL, Intervallo di misura 0-100 µL', 'Microlitro', 0:10:00, 'P5'),
('Ingrandimento 40-1600x, Illuminazione a LED', 'Microscopio elettronico', 0:10:00, 'P12'),
('Intervallo di misura 0-100 °C, Precisione ± 0,1 °C', 0:10:00, 'P12'),
('Ingrandimento 40-400x, Illuminazione a LED', 'Microscopio ottico', 0:10:00, 'P13'),
('Intervallo di misura 0-100 °C, Precisione ± 0,1 °C', 0:10:00, 'P13'),
('Precisione ± 0,1 µg, Intervallo di misura 0-100 µg', 'Sonda ELISA', 00:50:00, 'P6'),
('Precisione ± 0,5 mL, Volume max 10 mL', 'Pipetta a volume variabile', 00:50:00, 'P6'),
('Formato max A3, Stampa a colori', 'Stampante a getto di inchiostro', 03:30:00, 'P7'),
('Risoluzione 1440 dpi, 8 testine di stampa', 'Stampante a sublimazione', 03:30:00, 'P7'),
('Capacità 250 ml, Materiale ceramica', 'Mortaio in ceramica', 01:30:00, 'P8'),

```
('Capacità 100 ml, Materiale acciaio inox', 'Mortaio in acciaio inox', 01:30:00, 'P8'),
('Capacità 50 ml, Materiale agate', 'Mortaio in agate', 01:30:00, 'P18'),
('Capacità 1000 ml, Materiale vetro', 'Mortaio in vetro', 01:30:00, 'P20'),
('Capacità 200 ml, Materiale metallo pesante', 'Mortaio in metallo pesante', 01:30:00, 'P20'),
('range di temperatura regolabile', 'Incubatore cellulare', 0:20:00, 'P9'),
('alte prestazioni', 'Centrifuga', 0:20:00, 'P9'),
('sensibilità di rilevamento di 5 µl', 'Analizzatore di trigliceridi', 0:20:00, 'P10'),
('precisione del 5%', 'Analizzatore di VES', 0:20:00, 'P10'),
('schermo OLED da 8 pollici', 'Monitor per visualizzazione risultati', 0:20:00, 'P19'),
('alte prestazioni', 'Centrifuga', 0:20:00, 'P19');

 -- Popolamento di PRENOTAZIONE

INSERT INTO prenotazione
(data_prenotazioneS,ora_prenotazioneS,tempo_utilizzoS,Username_fk,CodStrumento_fk)
 VALUES
 ('2023-02-02', '12:30:00', '01:00:00','lucacardone', 1),
 ('2023-01-05', '15:35:00', '00:05:00', 'robertaben', 18),
 ('2023-01-19', '16:30:00', '00:30:00','lucacardone', 5),
 ('2023-02-01', '10:30:00', '00:15:00', 'robertaben', 7);

 -- Popolamento di POSTAZIONE

INSERT INTO Postazione (CodPostazione,num_posti,CodL_fk)
VALUES
  ('P1',4,'L1'),('P11',2,'L1'),
  ('P2',6,'L2'),('P15',3,'L2'),('P16',1,'L2'),
  ('P3',2,'L3'),('P17',8,'L3'),
  ('P4',3,'L4'),('P14',1,'L4'),
  ('P5',4,'L5'),('P12',2,'L5'),('P13',1,'L5'),
  ('P6',11,'L6'),
  ('P7',13,'L7'),
  ('P8',4,'L8'), ('P18',4,'L8'), ('P20',1,'L8'),
  ('P9',14,'L9'),
  ('P10',2,'L10'),('P19',3,'L10');


 -- Popolamento di UTENTE

INSERT INTO Utente (username, email, pw)
VALUES
  ('robertaben', 'roberta.beneduce@gmail.com', 'roberta'),
  ('lucacardone', 'cardoneluca001@gmail.com', 'luca');
```

```
-- Popolamento di APPARTENENZA

INSERT INTO Appartenenza
VALUES
('L1',1),
('L2',2),('L2',4),('L2',5),
('L3',3),('L3',10),
('L4',11),
('L5',9),('L5',8),('L5',7),('L5',12),
('L6',8),('L6',17),('L6',15),
('L7',2),('L7',16),('L7',3),('L7',11),('L7',1),
('L8',14),
('L9',4),('L9',7),('L9',9),
('L10',11),('L10',12),('L10',1),('L10',14),('L10',17);

--sede 6 e 13 non hanno laboratori


-- Popolamento di OCCUPATO

INSERT INTO occupato
VALUES
('robertaben','P1'),('robertaben','P2'),('robertaben','P11'),
('lucacardone','P2'),('lucacardone','P17'),('lucacardone','P3'),('lucacardone','P14');




-- Popolamento di UTILIZZATO

INSERT INTO utilizzato (username_fk,codStrumento_fk,datautilizzo,oraInizio,oraFine)
VALUES
('robertaben',1,'2023-11-12','16:00','16:30'),
('lucacardone',1,'2023-11-12','09:00','10:00'),
('lucacardone',2,'2023-01-23','09:00','10:00'),
('robertaben',12,'2023-02-12','08:20','08:35'),
('robertaben',17,'2023-01-24','17:00','17:05');
```