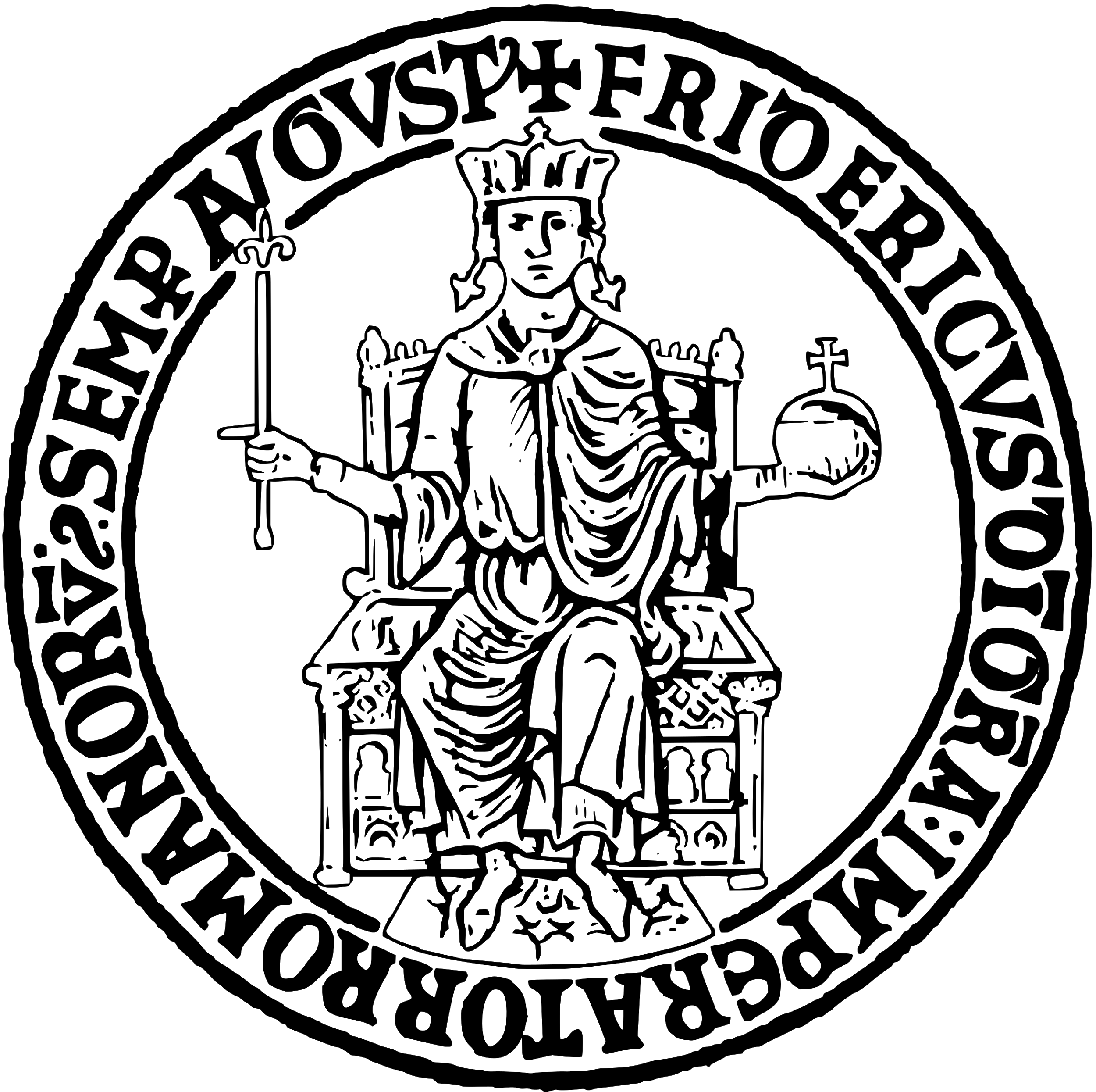
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL’INFORMAZIONE

Carmine Testa Dmytro Lozyak

N86002676 N86002756



Sunto

Base di dati relazionale per il tracciamento delle occasioni di contatto tra persone per recuperare informazioni sulla possibile diffusione del virus

Docenti

Adriano Peron   
Silvio Barra

Progettazione e sviluppo di una base di dati relazionale per monitoraggio pandemia

Corso di laurea in Informatica Anno Accademico 2020/2021

*Indice*

1. **DESCRIZIONE PROGETTO** Pag. 3
   1. Introduzione Pag. 3
   2. Analisi del problema Pag. 3
2. **PROGETTAZIONE CONCETTUALE** Pag. 4
   1. Introduzione alla progettazione concettuale Pag. 4
   2. Convenzioni adottate Pag. 4
   3. Class Diagram Pag. 5
   4. Ristrutturazione Pag. 6
      1. Valutazione delle associazioni 1 a 1 Pag. 6
      2. Valutazione delle informazioni ridondanti Pag. 6
      3. Valutazione degli attributi a valore multiplo Pag. 6
      4. Gerarchie di specializzazione Pag. 7
      5. Chiavi surrogate Pag. 7
   5. Class Diagram ristrutturato Pag. 8
   6. Dizionari Pag. 9
      1. Dizionari classi Pag. 9
      2. Dizionari associazioni Pag. 13
      3. Dizionari vincoli Pag. 16
3. **PROGETTAZIONE LOGICA Pag. 20**
   1. Schema Logico Pag. 20
4. **PROGETTAZIONE FISICA Pag. 23**
   1. Introduzione alla progettazione fisica Pag. 23
   2. Convenzioni adottate Pag. 23
   3. Definizione delle tabelle Pag. 23
      1. Tabella: Persona Pag. 24
      2. Tabella: StatoSalute Pag. 25
      3. Tabella: Parentela Pag. 27
      4. Tabella: Vaccino Pag. 28
      5. Tabella: Somministrazione Pag. 29
      6. Tabella: Affluenza Pag. 30
      7. Tabella: Compagni Pag. 31
      8. Tabella: Colleghi Pag. 32
      9. Tabella: Studente Pag. 33
      10. Tabella: Lavoratore Pag. 34
      11. Tabella: Lavoro Pag. 36
      12. Tabella: Luogo Pag. 38
      13. Tabella: AmbienteFamiliare Pag. 39
      14. Tabella: AmbienteEventoOccasionale Pag. 41
      15. Tabella: AmbienteLavoro Pag. 43
      16. Tabella: AmbienteDiStudio Pag. 45
      17. Tabella: Aula Pag. 47
      18. Tabella: ControlloMedico Pag. 48

*Indice*

* + 1. Tabella: Frequentazione Pag. 50
    2. Tabella: Gestione Pag. 51

1. **MANUALE D’USO Pag. 52**
   1. Introduzione al manuale d’uso Pag. 52
   2. Introduzione all’interfaccia:Login Pag. 53
   3. Introduzione all’interfaccia:Schermata primaria Pag. 53
   4. Schermata di selezione Pag. 54
   5. Aggiunta di un record Pag. 55
   6. Modifica di un record Pag. 56
   7. Ricerca Pag. 57

*Capitolo 1*

Descrizione progetto

## **1.1 Introduzione**

E’ richiesta la progettazione di una base di dati per tracciare le occasioni di contatto tra persone per recuperare informazioni sulla possibile diffusione del virus risalendo alle occasioni di contatto.

Degli utenti va memorizzato:

* La possibile relazione (parentela, essere colleghi di lavoro);
* I luoghi in cui sono stati presenti (ambienti familiari, ambienti di lavoro, eventi occasionali);
* Occasioni di contatto;
* Tamponi e stato sierologico;
* Informazioni sulla eventuale malattia.

Si possono integrare in modo autonomo le specifiche.

## **1.2 Analisi del problema**

Viene sviluppata una base di dati relazionale che memorizza informazioni riguardo gli spostamenti delle persone e i luoghi che queste ultime hanno visitato con annesse informazioni riguardo l’orario e la data della visita. Viene memorizzata la posizione, la durata e il distanziamento di un eventuale contatto tra individui. Ci sarà dunque una visione generale sullo stato di salute dei singoli utenti e test sierologici o tamponi effettuati da essi. Per quanto riguarda un vaccino futuro, oltre a salvare i dati del vaccino, andremo a memorizzare le somministrazioni effettuate con eventuali complicazioni annesse.

Riguardo l’entità principale del sistema, ovvero la persona, andremo a specificare la categoria della stessa che può essere Studente: in questo modo conosciamo la frequenza abituale dei luoghi di studio o meno e quindi poter tenere traccia degli spostamenti che gli studenti compiono; Lavoratore: conoscendo l’ambiente lavorativo con orari di ingresso e uscita ed eventuali colleghi possiamo stabilire con più facilità gli eventuali contatti e possibili rischi di diffusione.; una Persona Comune che non si identifica nelle precedenti due categorie verrà studiata come Persona Generica di cui verranno salvate le parentele e il grado, così come avviene per Studenti e Lavoratori.

Un generico luogo avrà attributi che lo identificano (Città, Provincia, CAP, ecc.) e potrà essere descritto tramite le seguenti categorie: ambiente di lavoro, evento occasionale, ambiente familiare, ambiente di studio.

*Capitolo 2*

Progettazione concettuale

## **2.1 Introduzione alla progettazione concettuale**

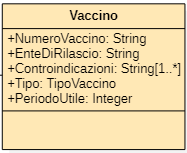
In questo capitolo inizia la progettazione della base di dati, seguendo tale ordine:

1. Class Diagram UML
2. Analisi di ristrutturazione
3. Class Diagram UML ristrutturato
4. Dizionari
   1. Classi
   2. Associazioni
   3. Vincoli

## **2.2 Convenzioni adottate**

Questo paragrafo serve a specificare un elenco di convenzioni adottate per facilitare la comprensione dei successivi schemi.

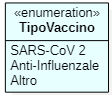
* Tutti gli attributi in assenza di cardinalità hanno una cardinalità implicita [1].
* Nelle descrizioni delle associazioni verticali il nome è situato a sinistra mentre la cardinalità a destra della linea.
* Nelle descrizioni delle associazioni orizzontali il nome è situato al di sopra della linea mentre la cardinalità in basso.
* Nelle descrizioni delle associazioni ricorsive il nome è situato all’esterno mentre la cardinalità all’interno.
* Le classi di enumerazione sono sempre situate nei pressi delle classi che contengono gli attributi con tipo corrispondente.
* Ogni entità ha un colore identificativo e sono i seguenti:
  + Classe (Colore giallo)



* + Classe d’associazione (Colore grigio)

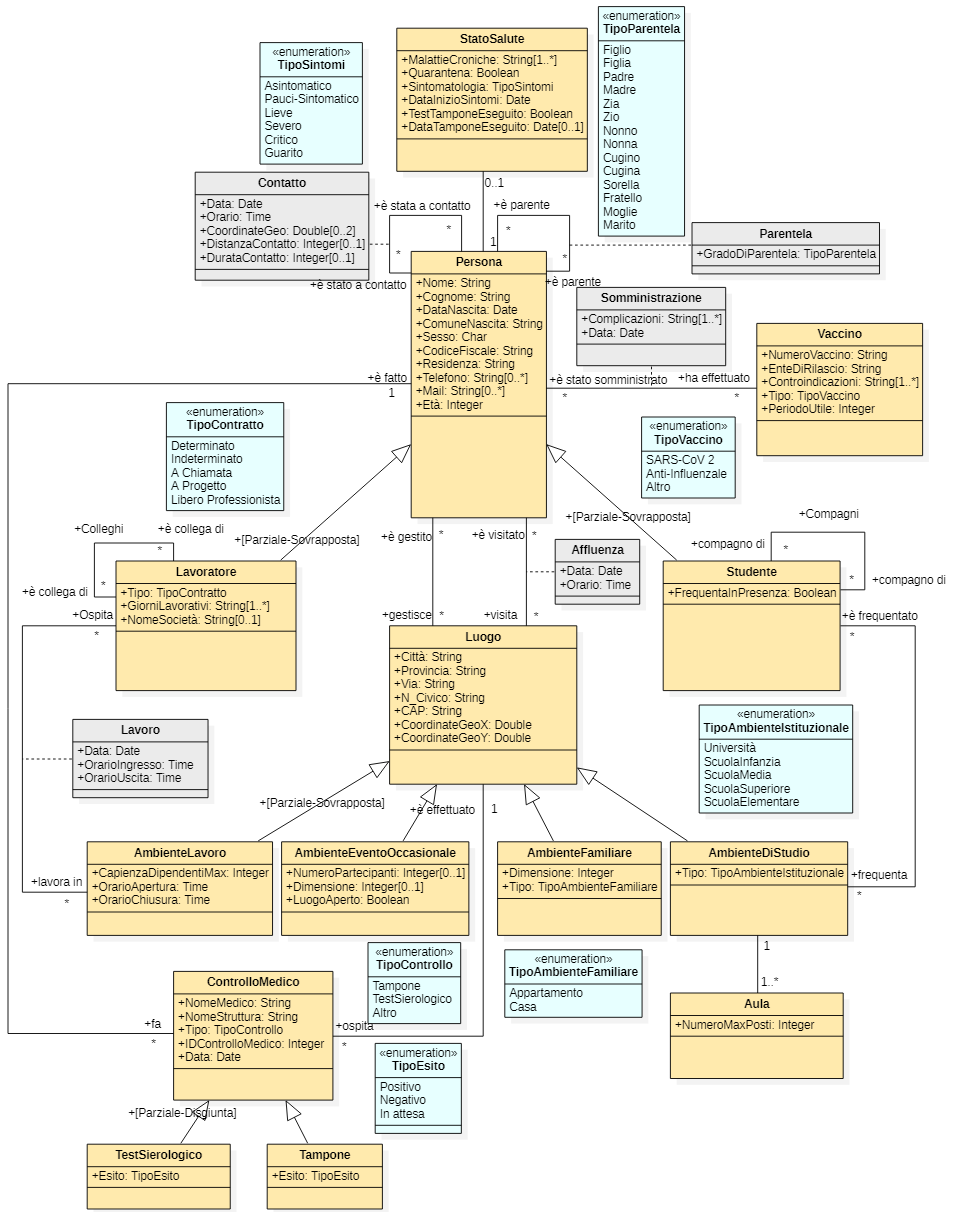


* + Classe enumerazione (Colore azzurro)



*Capitolo 2 Progettazione concettuale*

# **2.3 Class Diagram**



*Capitolo 2 Progettazione concettuale*

# **2.4 Ristrutturazione**

Nel seguente paragrafo ci occuperemo dell’analisi del class diagram per la successiva ristrutturazione, i parametri che andremo ad analizzare sono descritti successivamente.

# **2.4.1 Valutazione delle associazioni 1 a 1**

Nessuna associazione 1 a 1 è presente nel nostro Class Diagram.

# **2.4.2 Valutazione delle informazioni ridondanti**

Le informazioni ridondati rilevate riguardano i seguenti attributi:

Età (Persona): lasciamo l'attributo memorizzato essendo uno degli attributi fondamentali per studiare l'età media dei contagiati. Calcolare giorno per giorno l'età sarebbe svantaggioso rispetto alla memorizzazione.

DataTamponeEseguito (StatoSalute): lasciamo l’attributo memorizzate anche se potremmo calcolarlo percorrendo l'associazione "Controllo", avendolo memorizzato possiamo ricavare il numero di tamponi effettuati nell’ultimo periodo senza percorrere tutte le associazioni tra “Persona” e “ControlloMedico”.

NumeroPartecipanti (EventoOccasionale): lasciamo l’attributo memorizzato anche se potremmo calcolarlo andando a contare il numero di classi che partecipano all'associazione "Affluenza" con il ruolo di "è visitato" ma lasciamo memorizzato in modo da sapere in tempo reale eventuali assembramenti.

Contatto (*Classe*): Le occasioni di contratto vengono scoperte tramite la coincidenza delle persone nel medesimo luogo e allo stesso orario. Quindi dediciamo di eliminare la classe Contatto.

# **2.4.3 Valutazione degli attributi a valore multiplo**

I attributi mostrati successivamente hanno valore multiplo e quindi saranno gestiti in uno dei seguenti modi:

1. Dichiarare il tipo dell’attributo come String (METODO 1).
2. Introdurre un secondo attributo con il medesimo nome che ne identifica un secondo possibile valore (METODO 2).
3. Creare una classe ausiliaria che rappresenta l’attributo multiplo, associata alla classe di partenza (METODO 3). (Questo metodo non è stata applicato nell’attuale ristrutturazione).

Gli attributi a valore multipli sono i seguenti:

* Telefono (Persona) risolto con METODO 2
* Mail (Persona) risolto con METODO 2
* Controindicazioni (Vaccino) risolto con METODO 1
* Complicazioni (Somministrazione) risolto con METODO 1
* MalattieCroniche (StatoSalute) risolto con METODO 1
* GiorniLavorativi (Lavoratore) risolto con METODO 1

*Capitolo 2 Progettazione concettuale*

# **2.4.4 Gerarchie di specializzazione**

Le gerarchie individuate sono state gestite in uno dei seguenti modi:

1. Classi specializzate accorpate nella classe generale (METODO 1).
2. Classe generale tramandata alle classi specializzate (METODO 2). (Questo metodo non è stato applicato nell’attuale ristrutturazione)
3. Trasformazione della gerarchia in associazione (METODO 3).

Di seguito sono riportate le gerarchie di specializzazione e i metodi applicati:

* Specializzazione di “Persona” in “Studente” e “Lavoratore” è una specializzazione Parziale-Sovrapposta e, data la sua natura, è stato applicato il METODO 3 senza aggiunta di attributi e vincoli.
* Specializzazione di “Luogo” in “AmbienteLavoro”, “AmbienteEventoOccasionale”, “AmbienteFamiliare” e “AmbienteDiStudio”, è una specializzazione Parziale-Sovrapposta e data la sua natura è stato applicato il METODO 3 senza aggiunta di attributi e vincoli.
* Specializzazione di “ControlloMedico” in “Tampone” e “TestSierologico” è una specializzazione Parziale-Disgiunta ed è stato applicato il METODO 1 con l’attributo Tipo che identifica il tipo di ControlloMedico effettuato. Il vincolo di consistenza che andrà inserito, controllerà che quando l’attributo Tipo corrisponde ad altro l’attributo esito abbia valore NULL.

# **2.4.5 Chiavi surrogate**

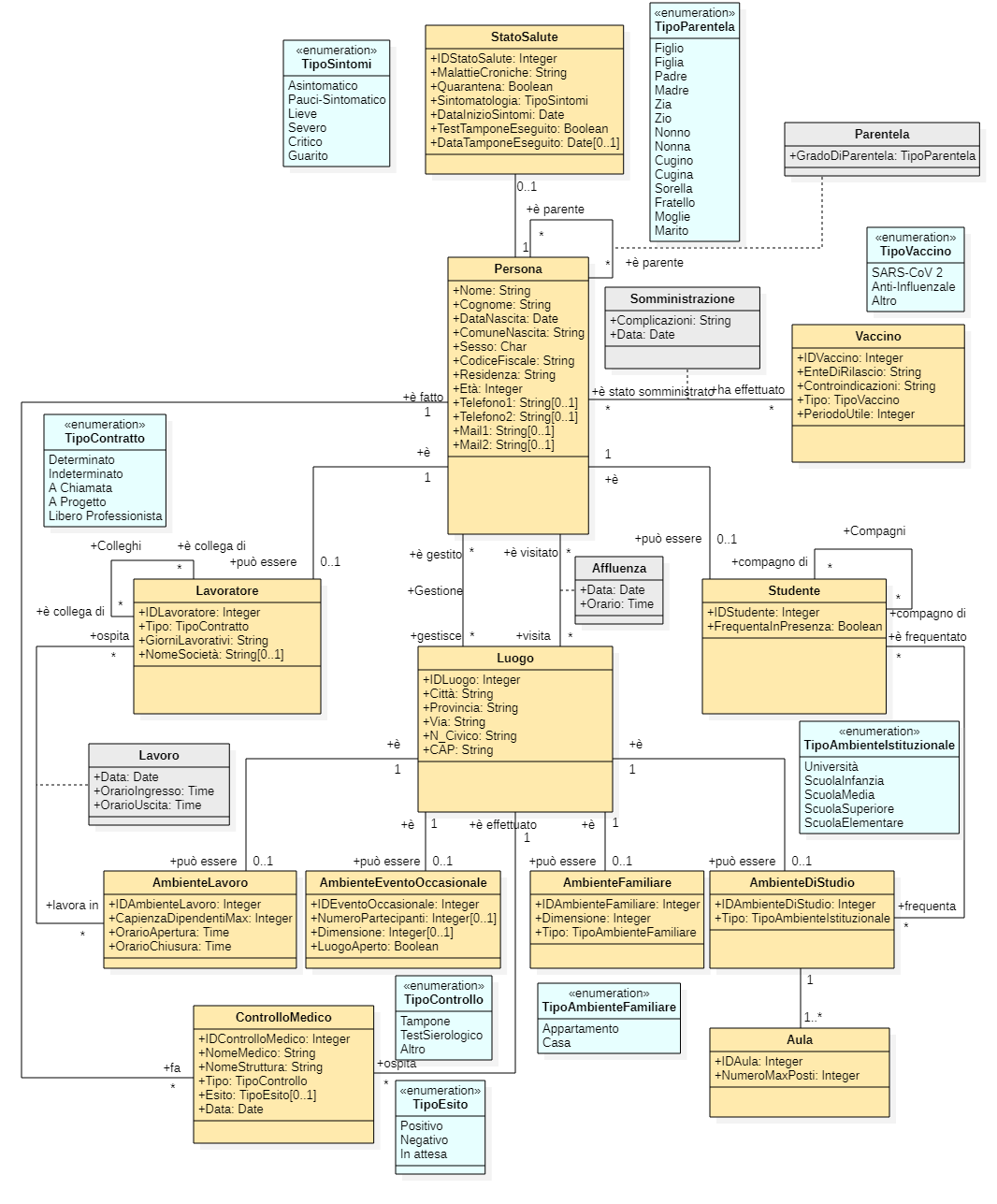
Di seguito sono riportate le chiavi surrogate inserite al fine di rendere leggibile l’identificato di ogni singola classe:

* IDLavoratore Lavoratore
* IDStudente Studente
* IDLuogo Luogo
* IDAmbienteLavoro AmbienteLavoro
* IDEventoOccasionale AmbienteEventoOccasionale
* IDAmbienteFamiliare AmbienteFamiliare
* IDAmbienteDiStudio AmbienteDiStudio
* IDAula Aula
* IDStatoSalute StatoSalute
* IDLuogo Luogo

Nota: Tutte le chiavi surrogate avranno come tipo Integer.

*Capitolo 2 Progettazione concettuale*

# **2.5 Class diagram ristrutturato**



*Capitolo 2 Progettazione concettuale*

# **2.6 Dizionario delle classi**

|  |  |  |
| --- | --- | --- |
| Classe | Descrizione | Attributi |
| Persona | Descrittore di una persona generica. | **Nome** (*String*): nome della persona  **Cognome** (*Sring*): cognome della persona.  **DataNascita** (*Date*): data di nascita della persona.  **ComuneNascita** (*String*): il comune di nascita.  **Sesso** (*Char*): sesso della persona.  **CodiceFiscale** (*String*): codice fiscale della persona.  **Residenza** (*String*): Comune di residenza della persona.  **Età** (*Integer*): età della persona.  **Telefono1** (*String*): primo numero di telefono della persona.  **Telefono2** (*String*): secondo numero di telefono della persona.  **Mail1** (*String*): primo Mail della persona.  **Mail2** (*String*): secondo Mail della persona. |
| Luogo | Descrittore di un luogo generico. | **IDLuogo** (*Integer*): Chiave identificativa per un luogo.  **Città** (*String*): nome della città.  **Provincia** (*String*): nome provincia.  **Via** (*String*): nome della via.  **N\_Civico** (*String*): numero civico, con eventuali lettere.  **CAP** (*String*): codice postale del luogo. |
| StatoSalute | Descrittore della cartella clinica di una persona. | **IDStatoSalute** (*Integer*): Chiave identificativa per lo stato di salute di una persona.  **MalattieCroniche** (*String*): le malattie croniche di cui soffre la persona.  **Quarantena** (*Boolean*): indica se la persona è attualmente è in uno stato di quarantena.  **Sintomatologia** (*TipoSintomi*): indica la tipologia dei sintomi per SARS-CoV 2.  **DataInizioSintomi** (*Date*): indica la data in cui sono comparsi i sintomi.  **TestTamponeEseguito** (*Boolean*): indica se la persona ha fatto un tampone.  **DataTamponeEseguito** (*Date*): indica la data dell’ultimo tampone eseguito. |

*Capitolo 2 Progettazione concettuale*

|  |  |  |
| --- | --- | --- |
| Parentela | Descrittore del tipo di parentela tra persone. | GradoDiParentela (*TipoParentela*): specifica il grado di parentela tra le persone. |
| Vaccino | Descrittore delle informazioni sul vaccino. | **IDVaccino** (*Integer*): indica la chiave identificativa per un vaccino.  **EnteDiRilascio** (*String*): indica il nome dell’azienda che ha prodotto il vaccino.  **Controindicazioni** (*String*): contiene informazioni sulle eventuali controindicazioni.  **Tipo** (*TipoVaccino*): indica la tipologia del vaccino.  **PeriodoUtile** (*Integer*): indica il tempo di efficacia del vaccino espresso in giorni. |
| Somministrazione | Descrittore dell’avvenuta somministrazione di un vaccino. | **Complicazioni** (*String*): indica gli eventuali complicazioni a seguito dell’avvenuta somministrazione un di vaccino.  **Data** (*Date*): indica la data dell’avvenuta somministrazione. |
| Lavoratore | Descrittore di una persona lavoratore. | **IDLavoratore** (*Integer*): indica la chiave identificativa per un lavoratore.  **Tipo** (*TipoContratto*): indica la tipologia del contratto sottoscritto dalla persona.  **GiorniLavorativi** (*String*): indica i giorni lavorativi in settimana definiti dal contratto.  **NomeSocietà** (*String*): nome della società per la quale lavorano le persone lavoratori. |
| AmbienteLavoro | Descrittore del luogo di lavoro | **IDAmbienteLavoro** (*Integer*): indica la chiave identificativa per un ambiente lavorativo.  **CapienzaDipendentiMax** (*Integer*): specifica la capienza massima indicata dalle norme di distanziamento anti SARS-CoV 2.  **OrarioApertura** (*Time*): indica l’ora in cui i dipendenti possono effettuare ingresso nell’ambiente di lavoro.  **OrarioChiusura** (*Time*): indica l’ora massima, entro cui i dipendenti devono lasciare il posto di lavoro. |

*Capitolo 2 Progettazione concettuale*

|  |  |  |
| --- | --- | --- |
| AmbienteEventoOccasionale | Descrittore dell’ambiente di un evento occasionale | IDEventoOccassionale (*Integer*): indica la chiave identificativa per un evento.  NumeroPartecipanti (*Integer*): indica il numero totale delle persone che hanno partecipato all’evento.  Dimensione (*Integer*): indica la dimensione (m2) dell’ambiente nel quale viene svolto l’evento.  LuogoAperto (*Boolean*): indentifica se evento viene svolto all’aperto o in un ambiente chiuso. |
| AmbienteFamiliare | Descrittore dell’ambiente famigliare di una persona. | **IDAmbienteFamiliare** (*Integer*): indica la chiave identificativa per un ambiente familiare.  **Dimensione** (*Integer*): indica la dimensione (m2) dell’ambiente familiare.  **Tipo** (*TipoAmbienteFamiliare*): specifica la tipologia dell’ambiente familiare (Appartamento o Casa). |
| Studente | Descrittore di una persona studente. | **IDStudente** (*Integer*): indica la chiave identificativa per uno studente.  **FrequentaInPresenza** (*Boolean*): indica se lo studente frequenta il suo ambiente di studio. |
| AmabienteDiStudio | Descrittore dell’ambiente istituzionale | **IDAmbienteDiStudio** (*Integer*): indica la chiave identificativa per un ambiente di studio.  **Tipo** (*TipoAmbienteIstituzionale*): indica la tipologia dell’ambiente di studio (scuola media, scuola superiore, università ecc). |
| Aula | Descrittore dell’aula in cui vengono svolte le lezioni. | **IDAula** (*Integer*): indica la chiave identificativa per un’aula.  NumeroMaxPosti (*Integer*): specifica la capienza massima indicata dalle norme anti SARS-CoV 2. |
| Lavoro | Descrittore degli orari di ingresso e uscita dall’ambiente di lavoro di un lavoratore | **Data** (*Date*): Indica la data di ingresso del lavoratore nell’ambiente di lavoro.  **OrarioIngresso** (*Time*): Indica l’orario di ingresso nell’ambiente di lavoro.  **OrarioUscita** (*Time*):Indica l’orario di uscita dall’ambiente di lavoro. |

*Capitolo 2 Progettazione concettuale*

|  |  |  |
| --- | --- | --- |
| ControlloMedico | Descrittore del controllo medico | IDControlloMedico (*Integer*): indica la chiave identificativa per il controllo medico.  NomeMedico (*String*): indica il nome e cognome del medico che ha effettuato il controllo.  NomeStruttura (*String*): indicai il nome dell’ospedale nel quale è stato effettuato il controllo medico.  Tipo (*TipoControllo*): specifica che tipo di controllo è stato effettuato (Tampone, TestSierologico o Altro).  Esito (*TipoEsito*): indica l’esito (Positivo, Negativo o In Attesa) del controllo medico di tipo Tampone o TestSierologico.  Data (*Date*): indica la data in cui è stato effettuato il controllo medico. |
| Affluenza | Descrittore degli orari in cui è stato visitato un luogo | **Data** (*Date*):Indica la data nella quale un luogo è stato visitato da una persona.  **Orario** (*Time*):Indica l’orario nel quale un luogo è stato visitato da una persona. |

*Capitolo 2 Progettazione concettuale*

# **2.7 Dizionario delle associazioni**

|  |  |  |
| --- | --- | --- |
| Nome associazione | Descrizione | Partecipanti |
| Gestione | Esprime la relazione tra un luogo e chi lo gestisce. | **Persona** [\*] ruolo (*gestisce*): indica le persone che gestiscono un luogo.  **Luogo** [\*] ruolo (*è* *gestito*): indica i luoghi gestiti da una persona. |
| Affluenza | Esprime l’affluenza delle persone in dei luoghi. | **Persona** [\*] ruolo (*visita*): indica le persone che visitano un luogo.  **Luogo** [\*] ruolo (*è* *visitato*): indica i luoghi visitati da una persona. |
| Parentela | Esprime la parentela che ci può essere tra persone. | **Persona** [\*] ruolo (*è parente*): indica le persone apparentante con una persona.  **Persona** [\*] ruolo (*è parente*): indica le persone apparentante con una persona. |
| Somministrazione | Esprime la relazione tra persona e un eventuale vaccino. | **Persona** [\*] ruolo (*ha effettuato*): indica le persone che hanno effettuato un vaccino.  **Vaccino** [\*] ruolo (*è stato somministrato*): indica i vaccini somministrati ad una persona. |
| Compagni | Esprime la relazione tra compagni di classe o di studio. | **Studente** [\*] ruolo (*compagno di*): indica gli studenti compagni di classe o studio di uno studente.  **Studente** [\*] ruolo (*compagno di*): indica gli studenti compagni di classe o studio di uno studente. |
| Lavoro | Esprime le abitudini che ha un lavoratore con il suo ambiente di lavoro. | **Lavoratore** [\*]ruolo (*lavora* *in*): indica i lavorati che lavorano in un ambiente di lavoro.  **AmbienteLavoro** [\*] ruolo (*ospita*): indica gli ambienti di lavoro che ospitano un lavoratore. |

*Capitolo 2 Progettazione concettuale*

|  |  |  |
| --- | --- | --- |
| Colleghi | Esprime la relazione che intercorre tra un lavoratore e i suoi colleghi di lavoro. | Lavoratore [\*] ruolo (*è* *collega* *di*): indica i lavoratori colleghi di un lavoratore.  Lavoratore [\*] ruolo (*è* *collega* *di*): indica i lavoratori colleghi di un lavoratore. |
| Visita medica | Esprime la relazione tra persone e i controlli medici che effettuano. | **Persona** [\*] ruolo (*fa*): indica le persone che si sottopongono ad un controllo medico.  **ControlloMedico** [1] ruolo (*è* *fatto*): indica il controllo medico sottoposto ad una persona. |
| Presidio medico | Esprime la relazione che c’è tra un controllo medico e il luogo dove viene effettuato. | **Luogo** [\*] ruolo (*ospita*): indica il luogo che ospita un controllo medico.  **ControlloMedico** [1] ruolo (*è* *effettuato*): indica il controllo medico effettuato in un luogo specfico. |
| Frequentazione | Esprime il legame che c’è tra uno studente e il luogo di studio che frequenta. | **Studente** [\*] ruolo (*frequenta*): indica gli studenti che frequentano un Ambiente di studio.  **AmbienteDiStudio** [\*] ruolo (*è* *frequentato*): indica gli ambienti di studio frequentati da uno studente. |
| Composizione | Esprime le aule che ci sono in un Ambiente di studio | **AmbienteDiStudio** [1..\*] ruolo (*è* *composto*): indica l’ambiente di studio composto da aule.  **Aula** [1] ruolo (*compone*): indica le aule che compongono un ambiente di studio. |
| Condizione clinica | Esprime il legame che c’è tra una persona il relativo stato di salute. | **Persona** [0..1] ruolo (*può essere*): indica la persona che ha lo stato di salute.  **StatoSalute** [1] ruolo (*fa* *riferimento*): indica lo stato di salute di una persona. |

*Capitolo 2 Progettazione concettuale*

|  |  |  |
| --- | --- | --- |
| PersonaLavoratore | Descrive il tipo di persona | Persona [0..1] ruolo (*può essere*) : indica la possibilità che una persona sia un lavoratore.  Lavoratore [1] ruolo (*è*): indica la persona legata al lavoratore. |
| PersonaStudente | Descrive il tipo di persona | **Persona** [0..1] ruolo (*può essere*) : indica la possibilità che una persona sia uno studente.  **Studente** [1] ruolo (*è*): indica la persona legata al studente. |
| LuogoAmbienteLavoro | Descrive il tipo di luogo | **Luogo** [0..1] ruolo (*può essere*) : indica la possibilità che un luogo sia un ambiente di lavoro.  **AmbienteLavoro** [1] ruolo (*è*): indica il luogo legato all’ambiente di lavoro. |
| LuogoAmbienteOccasionale | Descrive il tipo di luogo | **Luogo** [0..1] ruolo (*può essere*) : indica la possibilità che un luogo sia un ambiente di un evento occasionale.  **AmbienteEventoOccasionale** [1] ruolo (*è*): indica il luogo legato all’ambiente di un evento occasionale. |
| LuogoAmbienteFamiliare | Descrive il tipo di luogo | **Luogo** [0..1] ruolo (*può essere*) : indica la possibilità che un luogo sia un ambiente familiare.  **AmbienteFamiliare** [1] ruolo (*è*): indica il luogo legato all’ambiente familiare. |
| LuogoAmbienteStudio | Descrive il tipo di luogo | **Luogo** [0..1] ruolo (*può essere*) : indica la possibilità che un luogo sia un ambiente di studio.  **AmbienteDiStudio** [1] ruolo (*è*): indica il luogo legato all’ambiente di studio. |

*Capitolo 2 Progettazione concettuale*

# **2.8 Dizionario vincoli**

Nel seguente dizionario di vincoli sono state adottate le seguenti convenzioni per facilitare la lettura dei vincoli.

* Vincoli di dominio: sono rappresentati con il prefisso VD seguiti dall’attributo a cui fa riferimento il vincolo e infine la classe che lo contiene.
* Vincolo di ennupla: sono rappresentati con il prefisso VE seguiti da una coppia di attributi a cui fa riferimento il vincolo e infine la classe che li contiene.
* Vincolo di unicità: sono rappresentati con il prefisso VU seguiti da un nome che identifica la funzione del vincolo.
* Vincolo interrelazioniale: sono rappresentati con il prefisso VI seguiti da un nome che identifica la funzione del vincolo.

*Capitolo 2 Progettazione concettuale*

|  |  |
| --- | --- |
| Nome vincolo | Descrizione |
| VD\_Data\_Somministrazione\* | Una somministrazione dev’essere effettuata nel passato, quindi il valore Date deve risultare minore di SYS.DATE (Data corrente). |
| VU\_PersonaUnVaccinoPerVolta | Una persona può effettuare un solo vaccino alla volta e quindi in una certa Data-Orario non potrà effettuare due vaccini contemporaneamente. |
| VE\_NSocietàTipo\_Lavoratore | Se un lavoratore è un libero professionista allora non avrà il nome della società, quindi se il valore Tipo = “Libero Professionista” allora NomeSocietà = NULL. |
| VE\_OAperturaOChiusura\_Alavoro | Un ambiente di lavoro aprirà prima di chiudere ovviamente, quindi l’attributo OrarioApertura deve risultare minore di OrarioChiusura. |
| VE\_LuogoApertoDimensione\_AEOccasionale | Se un evento occasionale si svolge in un luogo aperto allora non avrà delle dimensioni specifiche, quindi se l’attributo “LuogoAperto” = True allora “Dimensione” = NULL. |
| VE\_LavoratoreLavoratore\_Colleghi | Un lavoratore non può essere collega di sé stesso. |
| VE\_StudenteStudente\_Compagni | Uno studente non può essere compagno di sé stesso. |
| VE\_PersonaPersona\_Parentela | Una persona non può essere parente di se stessa |
| VU\_CodiceFiscale\_Studente | Due studenti non possono essere la stessa persona. |
| VU\_CodiceFiscale\_Lavoratore | Due lavoratori non possono essere la stessa persona |
| VD\_DataNascita\_Persona\* | Una persona non può essere nata nel futuro. |
| VD\_Mail\_Persona | La mail della persona deve avere una struttura standard di una comune mail. |
| VD\_Data\_ControlloMedico\* | La data di un controllo medico non può essere nel passato |

*Capitolo 2 Progettazione concettuale*

|  |  |
| --- | --- |
| VE\_TipoEsito\_ControlloMedico | Se il valore dell’attributo Tipo = “Altro” allora Esito deve essere NULL e vale anche il viceversa. |
| VD\_NomeMedico\_ControlloMedico | Il nome del medico deve essere formato dal nome, lo spazio e il cognome (Es. “Salvatore Esposito”). |
| VE\_OraUscitaOraIngresso\_Lavoro | Orario ingresso deve essere antecedente all’orario di uscita. |
| VD\_Data\_Lavoro\* | Data deve essere antecedente o uguale alla data attuale (SYSDATE). |
| VD\_Date\_StatoSalute\* | La data dell’inizio dei sintomi e del tampone eseguito deve essere antecedente alla data attuale (SYSDATE). |
| VE\_TestTamponeEseguitoDataTamponeEseguito\_StatoSalute | Se il test del tampone eseguito ha valore False allora la data del tampone deve essere uguale a NULL e vale il viceversa. |
| VU\_PersonaUnLuogoPerVolta | Una persona può visitare un solo luogo per volta, e quindi in una certa Data-Orario non potrà trovarsi in due luoghi contemporaneamente. |
| VU\_FrequentazioneUnica | Una persona può frequentare al massimo una volta uno specifico ambiente di studio. |
| VU\_ColleghiUnico | Un lavoratore può essere collega di un lavoratore al massimo una volta. |
| VU\_CompagniUnico | Uno studente può essere compagnio di uno studente al massimo una volta. |
| VU\_ParentelaUnica | Una persona può essere parente di un'altra persona al più una volta. |
| VU\_GestioneUnica | Una persona può gestire un luogo al più una volta. |
| VU\_CodiceFiscale\_StatoSalute | Ogni persona può avere al più uno stato di salute associato. |
| VU\_AmbienteDiStudio | Ogni luogo può essere al più un ambiente di studio. |
| VU\_Ambiente\_Lavoro | Ogni luogo può essere al più un ambiente di lavoro. |

*Capitolo 2 Progettazione concettuale*

|  |  |
| --- | --- |
| VU\_Ambiente\_Evento\_Occasionale | Ogni luogo può essere al più un ambiente di un evento occasionale. |
| VU\_Ambiente\_Familiare | Ogni luogo può essere al più un ambiente familiare. |
| VD\_Data\_Affluenza\* | Una persona non può andare in un luogo nel futuro |

*Capitolo 3*

Progettazione logica

# **3.1 Schema logico**

Nel seguente capitolo andremo a definire, a partire dal precedente schema concettuale, uno schema logico.

Sono state adottate le seguenti convenzioni al fine di facilitare la lettura degli schemi relazionali:

* Gli attributi sottolineati con una linea definiscono la chiave primaria dello schema relazionale.
* Gli attributi sottolineati con due linee definiscono le chiavi esterne dello schema relazionale.
* Nel riguardo al di sotto dello schema relazionale saranno definite le chiavi primaria a cui fanno riferimento le chiavi esterne.

|  |
| --- |
|  |
| Persona (Nome, Cognome, DataNascita, ComuneNascita, Sesso, CodiceFiscale, Residenza, Età, Telefono1, Telefono2, Mail1, Mail2) |
| *Nessuna…* |
| AmbienteLavoro (IDAmbienteLavoro, CapienzaDipendentiMax, OrarioApertura, OrarioChiusura, FK\_IDLuogo) |
| * FK\_IDLuogo *Luogo*.IDLuogo |
| AmbienteEventoOccasionale (IDEventoOccasionale, NumeroPartecipanti, Dimensione, LuogoAperto, FK\_IDLuogo) |
| * FK\_IDLuogo *Luogo*.IDLuogo |
| AmbienteFamiliare (IDAmbienteFamiliare, Dimensione, Tipo, FK\_IDLuogo) |
| * FK\_IDLuogo *Luogo*.IDLuogo |
| AmbienteDiStudio (IDAmbienteDiStudio, Tipo, FK\_IDLuogo) |
| * FK\_IDLuogo *Luogo*.IDLuogo |
| Aula (IDAula, NumeroMqaxPosti, FK\_IDAmbienteDiStudio) |
| * FK\_IDAmbienteDiStudio *AmbienteDiStudio*.IDAmbienteDiStudio |

*Capitolo 3 Progettazione logica*

|  |
| --- |
| ControlloMedico (IDControlloMedico, NomeMedico, NomeOspedale, Tipo, Esito, Data, FK\_IDLuogo, FK\_IDPersona) |
| * FK\_IDPersona *Persona*.CodiceFiscale * FK\_IDLuogo *Luogo*.IDLuogo |
| Lavoro (FK\_IDLavorate, FK\_IDAmbienteLavoro, Data, OrarioIngresso, OrarioUscita) |
| * FK\_IDLavorate *Lavoratore*.IDLavoratore * FK\_IDAmbienteLavoro *AmbienteLavoro*.IDAmbienteLavoro |
| Frequentazione (FK\_IDStudente, FK\_IDAmbienteDiStudio) |
| * FK\_IDStudente *Studente*.IDStudente * FK\_IDAmbienteDiStudio *AmbienteDiStudio*.IDAmbienteDiStudio |
| Affluenza (FK\_IDPersona, FK\_IDLuogo, Data, Orario) |
| * FK\_IDPersona *Persona*.IDPersona * FK\_IDLuogo *Luogo*.IDLuogo |
| Gestione (FK\_IDGestore, FK\_IDLuogoGestito) |
| * FK\_IDGestore *Persona*.IDPersona * FK\_IDLuogoGestito *Luogo*.IDLuogo |
| Lavoratore (IDLavoratore, Tipo, GiorniLavorativi, NomeSocietà, FK\_IDPersona) |
| * FK\_IDPersona *Persona*.IDPersona |
| Colleghi (FK\_IDLavoratore, FK\_IDLavoratoreCollega) |
| * FK\_IDLavoratore *Lavoratore*.IDLavoratore * FK\_IDCollega *Lavoratore*.IDLavoratore |
| Luogo (IDLuogo, Città, Provincia, Via, N\_Civico, CAP) |
| *Nessuna…* |

*Capitolo 3 Progettazione logica*

|  |
| --- |
| StatoSalute (IDStatoSalute, MalattieCroniche, Quarantena, Sintomatologia, DataInizioSintomi, TestTamponeEseguito, DataTamponeEseguito, FK\_IDPersona) |
| * FK\_IDPersona *Persona.*CodiceFiscale |
| Vaccino (IDVaccino, EnteDiRilascio, Controindicazioni, Tipo, PeriodoUtile) |
| *Nessuna…* |
| Somministrazione (Complicazioni, Data, FK\_IDPersona, FK\_IDVaccino) |
| * FK\_IDPersona *Persona.*CodiceFiscale * FK\_IDVaccino *Vaccino.*IDVaccino |
| Studente (IDStudente, FrequentaInPresenza, FK\_IDPersona) |
| * FK\_IDPersona *Persona.*CodiceFiscale |
| Compagni (FK\_IDStudente, FK\_IDCompagno) |
| * FK\_IDStudente *Studente.*IDStudente * FK\_IDCompagno *Studente.*IDStudente |
| Parentela (GradoDiParentela, FK\_IDPersona, FK\_IDParente) |
| * FK\_IDPersona *Persona*.CodiceFiscale * FK\_IDParente *Persona*.CodiceFiscale |

*Capitolo 4*

Progettazione fisica

# **4.1 Introduzione progettazione fisica**

In questo capitolo verrà mostrato il codice SQL per implementare la base di dati, è stato utilizzato il DBMS ‘Oracle Database 18c Express Edition’ per la gestione della stessa.

# **4.2 Convenzione adottate**

Per adattare gli schemi relazionale alla sintassi di Oracle sono adoperate le seguenti modifiche :

* In ogni classe in cui comparivano gli attributi Data e Orario sono stati sostituiti con TIMESTAMP che rappresenta entrambe le informazioni, questo cambiamento è dovuto al fatto che Oracle non ha l’implementazione del tipo TIME.
* Tutti i vincoli che riguardavano l’attributo Orario ora fanno riferimento all’ attributo Data dovuto alla convenzione precedente.
* I vincoli segnati con il simbolo (\*) sono stati implementati con dei trigger.
* Gli attributi con tipo Boolean sono stati implementati con il tipo Number con i seguenti valori:
  + 0 per rappresentare False
  + 1 per rappresentare True

# **4.3 Definizioni delle tabelle**

Nel seguente paragrafo sono riportate le implementazioni delle tabelle in SQL con relativi vincoli e trigger.

*Capitolo 4 Progettazione fisica*

# **4.3.1 Tabella: Persona**

**CREATE** **TABLE** Persona(

Nome VARCHAR2(32) **NOT NULL**,

Cognome VARCHAR2(32) **NOT NULL**,

DataNascita DATE **NOT NULL**,

ComuneNascita VARCHAR2(32) **NOT** **NULL**,

Sesso CHAR(1) **NOT** **NULL** CHECK(Sesso **IN** (‘M’, ‘F’)),

CodiceFiscale CHAR(16) **NOT NULL**,

Residenza VARCHAR2(32) **NOT NULL**,

Età INTEGER **NOT NULL**,

Telefono1 CHAR(13),

Telefono2 CHAR(13),

Mail1 VARCHAR2(64),

Mail2 VARCHAR2(64),

);

-- VINCOLI: Persona

**ALTER** **TABLE** Persona

ADD(

-- CHIAVE PRIMARIA

CONSTRAINT PK\_Persona PRIMARY KEY(CodiceFiscale),

);

-- IMPLEMENTAZIONEà VINCOLO VD\_Mail\_Persona

CONSTRAINT VD\_Mail\_Persona

CHECK ((Mail1 **LIKE** '\_%@\_%.\_\_%' OR Mail1 **IS NULL)**

AND (Mail2 **LIKE** '\_%@\_%.\_\_%' OR Mail2 **IS NULL)**

);

*Capitolo 4 Progettazione fisica*

# **4.3.2 Tabella: StatoSalute**

**CREATE** **TABLE** StatoSalute(

IDStatoSalute INTEGER,

MalattieCroniche VARCHAR2(256) DEFAULT **ON** **NULL** 'Nessuna',

Quarantena NUMBER(1,0) **NOT NULL** CHECK(Quarantena **IN** (0, 1)),

Sintomatologia VARCHAR2(30) DEFAULT **ON** **NULL** 'Asintomatico',

DataInizioSintomi DATE,

TestTampoEseguito NUMBER(1,0) DEFAULT **ON** **NULL** 0 CHECK(TestTampoEseguito **IN** (0, 1)),

DataTamponeEseguito DATE,

FK\_IDPersona CHAR(16) **NOT NULL**

);

-- GESTIONE VALORE NULL DEL PRIMARY KEY

CREATE **OR** REPLACE TRIGGER SET\_IDStatoSalute

BEFORE INSERT **ON** StatoSalute

FOR EACH ROW

DECLARE

tmp\_IDStatoSalute StatoSalute.IDStatoSalute%TYPE;

**BEGIN**

IF (:NEW.IDStatoSalute IS **NULL**) THEN

**SELECT** NVL(**MAX**(S.IDStatoSalute), 0) **+** 1 INTO tmp\_IDStatoSalute

**FROM** StatoSalute S;

:NEW.IDStatoSalute :**=** tmp\_IDStatoSalute;

END IF;

END;

**/**

*Capitolo 4 Progettazione fisica*

**ALTER** **TABLE** StatoSalute

ADD(

-- CHIAVE PRIMARIA

CONSTRAINT PK\_StatoSalute PRIMARY KEY(IDStatoSalute),

-- Implementazione VINCOLO VE\_CODICEFISCALE\_STATOSALUTE

CONSTRAINT "VE\_CODICEFISCALE\_STATOSALUTE" UNIQUE ("FK\_IDPERSONA")

-- CHIAVE ESTERNA

CONSTRAINT FK\_StatoSalute\_Persona FOREIGN KEY(FK\_IDPersona)

REFERENCES Persona(CodiceFiscale),

-- IMPLEMENTAZIONE VINCOLO VE\_TestTamponeEseguitoDataTamponeEseguito\_StatoSalute

CONSTRAINT VE\_TestTamponeEseguitoDataTamponeEseguito\_StatoSalute

CHECK ((TestTampoEseguito **=** 1 **OR** DataTamponeEseguito **IS** **NULL**)

**AND** (TestTampoEseguito **=** 0 **OR** DataTamponeEseguito **IS NOT** **NULL**)),

-- Enumerazione per Sintomatologia(TipoSintomi)

CONSTRAINT EnumTipoSintomiStatoSalute

CHECK( Sintomatologia **IN** ('Asintomatico', 'Pauci-Sintomatico', 'Lieve', 'Severe', 'Critico', 'Guarito'))

);

-- IMPLEMENTAZIONE VINCOLO VD\_Date\_StatoSalute

CREATE **OR** REPLACE TRIGGER check\_Date\_StatoSalute

BEFORE INSERT **OR** **UPDATE** **ON** StatoSalute

FOR EACH ROW

**BEGIN**

IF(:NEW.DataInizioSintomi **>** SYSDATE **OR** :NEW.DataTamponeEseguito**>** SYSDATE)

THEN

RAISE\_APPLICATION\_ERROR( **-**20111, 'Invalida data inserita');

END IF;

END;

*Capitolo 4 Progettazione fisica*

# 4.3.4 Tabella: Parentela

**CREATE** **TABLE** Parentela(

GradoDiParentela VARCHAR2(30) **NOT NULL**,

FK\_IDPersona CHAR(16) **NOT NULL**,

FK\_IDParente CHAR(16) **NOT NULL**

);

-- VINCOLI: Parentela

**ALTER** **TABLE** Parentela

ADD(

-- CHIAVE ESTERNA

CONSTRAINT FK\_Parentela\_Persona1 FOREIGN KEY (FK\_IDPersona)

REFERENCES Persona(CodiceFiscale),

-- CHIAVE ESTERNA

CONSTRAINT FK\_Parentela\_Persona2 FOREIGN KEY (FK\_IDParente)

REFERENCES Persona(CodiceFiscale),

-- IMPLEMENTAZIONE VINCOLO VU\_ParentelaUnica

CONSTRAINT VU\_ParentelaUnica UNIQUE (FK\_IDPersona, FK\_IDParente),

-- IMPLEMENTAZIONE VINCOLO VE\_PersonaPersona\_Parentela

CONSTRAINT VE\_PersonaPersona\_Parentela

CHECK(FK\_IDPersona **<>** FK\_IDParente),

-- Enumerazione per GradoDiParentela(TipoParentela)

CONSTRAINT EnumTipoParentela

CHECK( GradoDiParentela **IN** ('Figlio', 'Figlia', 'Padre', 'Madre', 'Zia', 'Zio', 'Nonno', 'Nonna', 'Cugino', 'Cugina', 'Sorella', 'Fratello', ‘Moglie’, ‘Marito’))

);

*Capitolo 4 Progettazione fisica*

# 4.3.5 Tabella: Vaccino

**CREATE** **TABLE** Vaccino(

IDVaccino INTEGER,

EnteDiRilascio VARCHAR2(32) **NOT NULL**,

Controindicazioni VARCHAR2(32) DEFAULT **ON** **NULL** 'Nessuna',

Tipo VARCHAR2(32) **NOT NULL**,

PeriodoUtile INTEGER **NOT NULL**

);

-- GESTIONE VALORE NULL DEL PRIMARY KEY

CREATE **OR** REPLACE TRIGGER SET\_IDVaccino

BEFORE INSERT **ON** Vaccino

FOR EACH ROW

DECLARE

tmp\_IDVaccino Vaccino.IDVaccino%TYPE;

**BEGIN**

IF (:NEW.IDVaccino IS **NULL**) THEN

**SELECT** NVL(**MAX**(V.IDVaccino), 0) **+** 1 INTO tmp\_IDVaccino

**FROM** Vaccino V;

:NEW.IDVaccino :**=** tmp\_IDVaccino;

END IF;

END;

**/**

-- VINCOLI: Vaccino

**ALTER** **TABLE** Vaccino

ADD(

-- CHIAVE PRIMARIA

CONSTRAINT PK\_Vaccino PRIMARY KEY(IDVaccino),

-- Enumerazione per Tipo(TipoVaccino)

CONSTRAINT EnumTipoVaccino CHECK( Tipo **IN** ('SARS-CoV 2', 'Anti-Influenzale', 'Altro'))

);

*Capitolo 4 Progettazione fisica*

# 4.3.6 Tabella: Somministrazione

**CREATE** **TABLE** Somministrazione(

Complicazioni VARCHAR2(256) DEFAULT **ON** **NULL** 'Nessuna',

Data DATE **NOT NULL**,

FK\_IDPersona CHAR(16) **NOT NULL**,

FK\_IDVaccino INTEGER **NOT NULL**

);

-- VINCOLI: Somministrazione

**ALTER** **TABLE** Somministrazione

ADD(

-- CHIAVE ESTERNA

CONSTRAINT FK\_Somministrazione\_Persona FOREIGN KEY (FK\_IDPersona)

REFERENCES Persona(CodiceFiscale),

-- CHIAVE ESTERNA

CONSTRAINT FK\_Somministrazione\_Vaccino FOREIGN KEY (FK\_IDVaccino)

REFERENCES Vaccino(IDVaccino),

-- IMPLEMENTAZIONE VINCOLO VU\_PersonaUnVaccinoPerVolta

CONSTRAINT VU\_PersonaUnVaccinoPerVolta UNIQUE (FK\_IDPersona, FK\_IDVaccino, Data),

);

-- IMPLEMENTAZIONE VINCOLO VD\_Data\_Somministrazione

CREATE **OR** REPLACE TRIGGER check\_DataSomministrazione

BEFORE INSERT **OR** **UPDATE** **ON** Somministrazione

FOR EACH ROW

**BEGIN**

IF(:NEW.Data **>** SYSDATE)

THEN

RAISE\_APPLICATION\_ERROR( **-**20111, 'Invalida data inserita');

END IF;

END;

**/**

*Capitolo 4 Progettazione fisica*

# 4.3.7 Tabella: Affluenza

**CREATE** **TABLE** Affluenza(

Data TIMESTAMP **NOT NULL**,

FK\_IDPersona CHAR(16) **NOT NULL**,

FK\_IDLuogo INTEGER **NOT NULL**

);

-- VINCOLI: Affluenza

**ALTER** **TABLE** Affluenza

ADD(

-- CHIAVE ESTERNA

CONSTRAINT FK\_Affluenza\_Persona FOREIGN KEY (FK\_IDPersona)

REFERENCES Persona(CodiceFiscale),

-- CHIAVE ESTERNA

CONSTRAINT FK\_Affluenza\_Luogo FOREIGN KEY (FK\_IDLuogo)

REFERENCES Luogo(IDLuogo),

-- IMPLEMENTAZIONE VINCOLO VU\_PersonaUnLuogoPerVolta

CONSTRAINT VU\_PersonaUnLuogoPerVolta UNIQUE (FK\_IDPersona, Data)

);

-- IMPLEMENTAZIONE VINCOLO VD\_Data\_Affluenza

CREATE **OR** REPLACE TRIGGER CHECK\_DATA\_AFFLUENZA

BEFORE INSERT **OR** UPDATE **ON** AFFLUENZA

FOR EACH ROW

**BEGIN**

IF(:NEW.DATA **>=** SYSDATE )

THEN

RAISE\_APPLICATION\_ERROR( -20111, 'Invalida data inserita');

END IF;

END;

*Capitolo 4 Progettazione fisica*

# 4.3.8 Tabella: Compagni

**CREATE** **TABLE** Compagni(

FK\_IDStudente INTEGER **NOT NULL**,

FK\_IDCompagno INTEGER **NOT NULL**

);

-- VINCOLI: Compagni

**ALTER** **TABLE** Compagni

ADD(

-- CHIAVE ESTERNA

CONSTRAINT FK\_Compagni\_Studente FOREIGN KEY (FK\_IDStudente)

REFERENCES Studente(IDStudente) ON DELETE CASCADE,

-- CHIAVE ESTERNA

CONSTRAINT FK\_Compagni\_Compagno FOREIGN KEY (FK\_IDCompagno)

REFERENCES Studente(IDStudente) ON DELETE CASCADE,

-- IMPLEMENTAZIONE VINCOLO VU\_CompagniUnico

CONSTRAINT VU\_CompagniUnico UNIQUE (FK\_IDStudente, FK\_IDCompagno),

-- IMPLEMENTAZIONE VINCOLO VE\_StudenteStudente\_Compagni

CONSTRAINT VE\_StudenteStudente\_Compagni

CHECK(FK\_IDStudente **<>** FK\_IDCompagno)

);

*Capitolo 4 Progettazione fisica*

# 4.3.9 Tabella: Colleghi

**CREATE** **TABLE** Colleghi(

FK\_IDLavoratore INTEGER **NOT NULL**,

FK\_IDCollega INTEGER **NOT NULL**

);

-- VINCOLI: Colleghi

**ALTER** **TABLE** Colleghi

ADD(

-- CHIAVE ESTERNA

CONSTRAINT FK\_Colleghi\_Lavoratore FOREIGN KEY (FK\_IDLavoratore)

REFERENCES Lavoratore(IDLavoratore) ON DELETE CASCADE,

-- CHIAVE ESTERNA

CONSTRAINT FK\_Colleghi\_Collega FOREIGN KEY (FK\_IDCollega)

REFERENCES Lavoratore(IDLavoratore) ON DELETE CASCADE,

-- IMPLEMENTAZIONE VINCOLO VU\_ColleghiUnico

CONSTRAINT VU\_ColleghiUnico UNIQUE (FK\_IDLavoratore, FK\_IDCollega),

-- IMPLEMENTAZIONE VINCOLO VE\_LavoratoreLavoratore\_Colleghi

CONSTRAINT VE\_LavoratoreLavoratore\_Colleghi

CHECK(FK\_IDLavoratore **<>** FK\_IDCollega)

);

*Capitolo 4 Progettazione fisica*

# 4.3.10 Tabella: Studente

**CREATE** **TABLE** Studente(

IDStudente INTEGER ,

FrequentaInPresenza NUMBER(1,0) **NOT NULL** CHECK (FrequentaInPresenza **IN** (0,1)),

FK\_IDPersona CHAR(16) **NOT NULL**

);

-- VINCOLI: Studente

**ALTER** **TABLE** Studente

ADD(

-- CHIAVE PRIMARIA

CONSTRAINT PK\_Studente PRIMARY KEY (IDStudente) ,

-- CHIAVE ESTERNA

CONSTRAINT FK\_Studente\_Persona FOREIGN KEY(FK\_IDPersona)

REFERENCES Persona(CodiceFiscale) ON DELETE CASCADE ,

-- IMPLEMENTAZIONE VU\_CodiceFiscale\_Studente

CONSTRAINT VU\_CodiceFiscale\_Studente UNIQUE (FK\_IDPersona)

);

-- GESTIONE VALORE NULL DEL PRIMARY KEY

CREATE **OR** REPLACE TRIGGER SET\_IDStudente

BEFORE INSERT **ON** Studente

FOR EACH ROW

DECLARE

tmp\_IDStudente Studente.IDStudente%TYPE;

**BEGIN**

IF (:NEW.IDStudente IS **NULL**) THEN

**SELECT** NVL(**MAX**(S.IDStudente), 0) **+** 1 INTO tmp\_IDStudente

**FROM** Studente S;

:NEW.IDStudente :**=** tmp\_IDStudente;

END IF;

END;

*Capitolo 4 Progettazione fisica*

# 4.3.11 Tabella: Lavoratore

**CREATE** **TABLE** Lavoratore (

IDLavoratore INTEGER ,

Tipo VARCHAR2(32) **NOT NULL**,

GiorniLavorativi VARCHAR2(128) **NOT NULL**,

NomeSocietà VARCHAR2(32),

FK\_IDPersona CHAR(16) **NOT NULL**

);

-- VINCOLI: Lavoratore

**ALTER** **TABLE** Lavoratore

ADD(

-- CHIAVE PRIMARIA

CONSTRAINT PK\_Lavoratore PRIMARY KEY (IDLavoratore) ,

-- CHIAVE ESTERNA

CONSTRAINT FK\_Lavoratore\_Persona FOREIGN KEY(FK\_IDPersona)

REFERENCES Persona(CodiceFiscale) ON DELETE CASCADE ,

-- Enumerazione per Tipo(TipoContratto)

CONSTRAINT EnumTipoContratto

CHECK (Tipo **IN** ('Determinato','Indeterminato','A Chiamata', 'A Progetto', 'Libero Professionista'))

-- IMPLEMENTAZIONE VU\_CodiceFiscale\_Lavoratore

CONSTRAINT VU\_CodiceFiscale\_Lavoratore UNIQUE (FK\_IDPersona)

-- IMPLEMENTAZIONE VE\_NsocietàTipo\_Lavoratore

CONSTRAIN VE\_NSocietàTipo\_Lavoratore CHECK( tipo **<>** 'Libero Professionista' **OR**

nomeSocietà IS NULL)

);

*Capitolo 4 Progettazione fisica*

-- GESTIONE VALORE NULL DEL PRIMARY KEY

CREATE **OR** REPLACE TRIGGER SET\_IDLavoratore

BEFORE INSERT **ON** Lavoratore

FOR EACH ROW

DECLARE

tmp\_IDLavoratore Lavoratore.IDLavoratore%TYPE;

**BEGIN**

IF (:NEW.IDLavoratore IS **NULL**) THEN

**SELECT** NVL(**MAX**(L.IDLavoratore), 0) **+** 1 INTO tmp\_IDLavoratore

**FROM** Lavoratore L;

:NEW.IDLavoratore :**=** tmp\_IDLavoratore;

END IF;

END;

**/**

*Capitolo 4 Progettazione fisica*

# 4.3.12 Tabella: Lavoro

**CREATE** **TABLE** Lavoro(

FK\_IDLavoratore INTEGER ,

FK\_IDAmbienteDiLavoro INTEGER **NOT NULL**,

OrarioIngresso TIMESTAMP **NOT NULL**,

OrarioUscita TIMESTAMP **NOT NULL**

);

-- VINCOLI: Lavoro

**ALTER** **TABLE** Lavoro

ADD(

-- CHIAVE ESTERNA

CONSTRAINT FK\_Lavoro\_Lavoratore FOREIGN KEY(FK\_IDLavoratore)

REFERENCES Lavoratore(IDLavoratore) ON DELETE CASCADE ,

-- CHIAVE ESTERNA

CONSTRAINT FK\_Lavoro\_AmbienteDiLavoro FOREIGN KEY(FK\_IDAmbienteDiLavoro)

REFERENCES AmbienteLavoro(IDAmbienteLavoro) ON DELETE CASCADE ,

-- IMPLEMENTAZIONE VINCOLO VE\_OraUscitaOraIngresso\_Lavoro

CONSTRAINT VE\_OraUscitaOraIngresso\_Lavoro

CHECK (OrarioIngresso**<**OrarioUscita) ,

);

*Capitolo 4 Progettazione fisica*

**-** IMPLEMENTAZIONE VINCOLO VD\_Data\_Lavoro

CREATE **OR** REPLACE TRIGGER check\_Data\_Lavoro

BEFORE INSERT **OR** **UPDATE** **ON** Lavoro

FOR EACH ROW

**BEGIN**

IF(:NEW.OrarioIngresso **>=** SYSDATE **OR** :NEW.OrarioUscita **>=** SYSDATE)

THEN

RAISE\_APPLICATION\_ERROR( **-**20111, 'Invalida data inserita');

END IF;

END;

**/**

*Capitolo 4 Progettazione fisica*

# 4.3.13 Tabella: Luogo

**CREATE** **TABLE** Luogo (

IDLuogo INTEGER **NOT NULL**,

Città VARCHAR2(32) **NOT NULL**,

Provincia VARCHAR2(32) **NOT NULL**,

Via VARCHAR2(32) **NOT NULL**,

N\_Civico VARCHAR2(8) **NOT NULL**,

CAP VARCHAR2(5) **NOT NULL**

);

-- VINCOLI: Luogo

**ALTER** **TABLE** Luogo

ADD(

-- CHIAVE PRIMARIA

CONSTRAINT PK\_Luogo PRIMARY KEY (IDLuogo)

);

-- GESTIONE VALORE NULL DEL PRIMARY KEY

CREATE **OR** REPLACE TRIGGER SET\_IDLuogo

BEFORE INSERT **ON** Luogo

FOR EACH ROW

DECLARE

tmp\_IDLuogo Luogo.IDLuogo%TYPE;

**BEGIN**

IF (:NEW.IDLuogo IS **NULL**) THEN

**SELECT** NVL(**MAX**(L.IDLuogo), 0) **+** 1 INTO tmp\_IDLuogo

**FROM** Luogo L;

:NEW.IDLuogo :**=** tmp\_IDLuogo;

END IF;

END;

**/**

*Capitolo 4 Progettazione fisica*

# 4.3.14 Tabella: AmbienteFamiliare

**CREATE** **TABLE** AmbienteFamiliare (

IDAmbienteFamiliare INTEGER **NOT NULL**,

Dimensione INTEGER **NOT NULL**,

Tipo VARCHAR2(32) **NOT NULL**,

FK\_IDLuogo INTEGER **NOT NULL**

);

-- VINCOLI: AmbienteFamiliare

**ALTER** **TABLE** AmbienteFamiliare

ADD(

-- CHIAVE PRIMARIA

CONSTRAINT PK\_AmbienteFamiliare PRIMARY KEY (IDAmbienteFamiliare) ,

-- CHIAVE ESTERNA

CONSTRAINT FK\_AmbienteFamiliare\_Luogo FOREIGN KEY(FK\_IDLuogo)

REFERENCES Luogo(IDLuogo) ON DELETE CASCADE,

-- Enumerazione per Tipo(TipoAmbienteFamiliare)

CONSTRAINT EnumTipoAmbienteFamiliare

CHECK (Tipo **IN** ('Appartamento', 'Casa')),

-- IMPLEMENTAZIONE VINCOLO VU\_AMBIENTE\_FAMILIARE

CONSTRAINT VU\_AMBIENTE\_FAMILIARE UNIQUE (FK\_IDLUOGO)

);

*Capitolo 4 Progettazione fisica*

-- GESTIONE VALORE NULL DEL PRIMARY KEY

CREATE **OR** REPLACE TRIGGER SET\_IDAmbienteFamiliare

BEFORE INSERT **ON** AmbienteFamiliare

FOR EACH ROW

DECLARE

tmp\_IDAmbienteFamiliare AmbienteFamiliare.IDAmbienteFamiliare%TYPE;

**BEGIN**

IF (:NEW.IDAmbienteFamiliare IS **NULL**) THEN

**SELECT** NVL(**MAX**(A.IDAmbienteFamiliare), 0) **+** 1 INTO tmp\_IDAmbienteFamiliare

**FROM** AmbienteFamiliare A;

:NEW.IDAmbienteFamiliare :**=** tmp\_IDAmbienteFamiliare;

END IF;

END;

**/**

*Capitolo 4 Progettazione fisica*

# 4.3.15 Tabella: AmbienteEventoOccasionale

**CREATE** **TABLE** AmbienteEventoOccasionale (

IDEventoOccasionale INTEGER ,

NumeroPartecipanti INTEGER ,

Dimensione INTEGER ,

LuogoAperto NUMBER(1,0) DEFAULT **ON** **NULL** 0 CHECK (LuogoAperto **IN** (0,1)),

FK\_IDLuogo INTEGER **NOT NULL**

);

-- VINCOLI: AmbienteEventoOccasionale

**ALTER** **TABLE** AmbienteEventoOccasionale

ADD(

--CHIAVE PRIMARIA

CONSTRAINT PK\_AmbienteEventoOccasionale PRIMARY KEY (IDEventoOccasionale) ,

--CHIAVE ESTERNA

CONSTRAINT FK\_AmbienteEventoOccasionale\_Luogo FOREIGN KEY(FK\_IDLuogo)

REFERENCES Luogo(IDLuogo) ON DELETE CASCADE ,

-- IMPLEMENTAZIONE VINCOLO VU\_AMBIENTE\_EVENTO\_OCCASIONALE

CONSTRAINT VU\_AMBIENTE\_EVENTO\_OCCASIONALE UNIQUE (FK\_IDLUOGO),

--IMPLEMENTAZIONE VINCOLO VE\_LuogoApertoDimensione\_AEOccasionale

CONSTRAINT VE\_LuogoApertoDimensione\_AEOccasionale

CHECK (LuogoAperto **=** 0 **OR** Dimensione **=** **NULL**)

);

*Capitolo 4 Progettazione fisica*

-- GESTIONE VALORE NULL DEL PRIMARY KEY

CREATE **OR** REPLACE TRIGGER SET\_IDEventoOccasionale

BEFORE INSERT **ON** AmbienteEventoOccasionale

FOR EACH ROW

DECLARE

tmp\_IDEventoOccasionale AmbienteEventoOccasionale.IDEventoOccasionale%TYPE;

**BEGIN**

IF (:NEW.IDEventoOccasionale IS **NULL**) THEN

**SELECT** NVL(**MAX**(A.IDEventoOccasionale), 0) **+** 1 INTO tmp\_IDEventoOccasionale

**FROM** AmbienteEventoOccasionale A;

:NEW.IDEventoOccasionale :**=** tmp\_IDEventoOccasionale;

END IF;

END;

**/**

*Capitolo 4 Progettazione fisica*

# 4.3.16 Tabella: AmbienteLavoro

**CREATE** **TABLE** AmbienteLavoro(

IDAmbienteLavoro INTEGER,

CapienzaDipendentiMax INTEGER **NOT NULL**,

OrarioApertura TIMESTAMP **NOT NULL**,

OrarioChiusura TIMESTAMP **NOT NULL**,

FK\_IDLuogo INTEGER **NOT NULL**

);

-- VINCOLI: AmbienteLavoro

**ALTER** **TABLE** AmbienteLavoro

ADD(

-- CHIAVE PRIMARIA

CONSTRAINT PK\_AmbienteLavoro PRIMARY KEY (IDAmbienteLavoro) ,

-- CHIAVE ESTERNA

CONSTRAINT FK\_AmbienteLavoro\_Luogo FOREIGN KEY(FK\_IDLuogo)

REFERENCES Luogo(IDLuogo) ON DELETE CASCADE,

-- IMPLEMENTAZIONE VINCOLO VE\_OAperturaOChiusura\_ALavoro

CONSTRAINT VE\_OAperturaOChiusura\_ALavoro CHECK (OrarioApertura **<** OrarioChiusura)

-- IMPLEMENTAZIONE VINCOLO VU\_AMBIENTE\_LAVORO

CONSTRAINT VU\_AMBIENTE\_LAVORO UNIQUE (FK\_IDLUOGO),

);

*Capitolo 4 Progettazione fisica*

-- GESTIONE VALORE NULL DEL PRIMARY KEY

CREATE **OR** REPLACE TRIGGER SET\_IDAmbienteLavoro

BEFORE INSERT **ON** AmbienteLavoro

FOR EACH ROW

DECLARE

tmp\_IDAmbienteLavoro AmbienteLavoro.IDAmbienteLavoro%TYPE;

**BEGIN**

IF (:NEW.IDAmbienteLavoro IS **NULL**) THEN

**SELECT** NVL(**MAX**(A.IDAmbienteLavoro), 0) **+** 1 INTO tmp\_IDAmbienteLavoro

**FROM** AmbienteLavoro A;

:NEW.IDAmbienteLavoro :**=** tmp\_IDAmbienteLavoro;

END IF;

END;

**/**

*Capitolo 4 Progettazione fisica*

# 4.3.17 Tabella: AmbienteDiStudio

**CREATE** **TABLE** AmbienteDiStudio (

IDAmbienteDiStudio INTEGER ,

Tipo VARCHAR2(32) **NOT NULL**,

FK\_IDLuogo INTEGER **NOT NULL**

);

-- VINCOLI: AmbienteDiStudio

**ALTER** **TABLE** AmbienteDiStudio

ADD(

-- CHIAVE PRIMARIA

CONSTRAINT PK\_AmbienteDiStudio PRIMARY KEY (IDAmbienteDiStudio) ,

-- CHIAVE ESTERNA

CONSTRAINT FK\_AmbienteDiStudio\_Luogo FOREIGN KEY(FK\_IDLuogo)

REFERENCES Luogo(IDLuogo) ON DELETE CASCADE ,

-- Enumerazione per Tipo(TipoAmbienteIstituzionale)

CONSTRAINT EnumTipoAmbienteIstituzionale

CHECK (Tipo **IN** ('Università', 'ScuolaInfanzia', 'ScuolaMedia', 'ScuolaSuperiore', 'ScuolaElementare'))

-- IMPLEMENTAZIONE VINCOLO VU\_AMBIENTEDISTUDIO

CONSTRAINT VU\_AMBIENTEDISTUDIO UNIQUE (FK\_IDLUOGO),

);

*Capitolo 4 Progettazione fisica*

-- GESTIONE VALORE NULL DEL PRIMARY KEY

CREATE **OR** REPLACE TRIGGER SET\_IDAmbienteDiStudio

BEFORE INSERT **ON** AmbienteDiStudio

FOR EACH ROW

DECLARE

tmp\_IDAmbienteDiStudio AmbienteDiStudio.IDAmbienteDiStudio%TYPE;

**BEGIN**

IF (:NEW.IDAmbienteDiStudio IS **NULL**) THEN

**SELECT** NVL(**MAX**(A.IDAmbienteDiStudio), 0) **+** 1 INTO tmp\_IDAmbienteDiStudio

**FROM** AmbienteDiStudio A;

:NEW.IDAmbienteDiStudio :**=** tmp\_IDAmbienteDiStudio;

END IF;

END;

**/**

*Capitolo 4 Progettazione fisica*

# 4.3.18 Tabella: Aula

**CREATE** **TABLE** Aula (

IDAula INTEGER ,

NumeroMaxPosti INTEGER **NOT NULL**,

FK\_IDAmbienteDiStudio INTEGER **NOT NULL**

);

-- VINCOLI: Aula

**ALTER** **TABLE** Aula

ADD(

-- CHIAVE PRIMARIA

CONSTRAINT PK\_Aula PRIMARY KEY (IDAula) ,

-- CHIAVE ESTERNA

CONSTRAINT FK\_Aula\_AmbienteDiStudio FOREIGN KEY(FK\_IDAmbienteDiStudio)

REFERENCES AmbienteDiStudio(IDAmbienteDiStudio) ON DELETE **SET** **NULL**

);

-- GESTIONE VALORE NULL DEL PRIMARY KEY

CREATE **OR** REPLACE TRIGGER SET\_IDAula

BEFORE INSERT **ON** Aula

FOR EACH ROW

DECLARE

tmp\_IDAula Aula.IDAula%TYPE;

**BEGIN**

IF (:NEW.IDAula IS **NULL**) THEN

**SELECT** NVL(**MAX**(A.IDAula), 0) **+** 1 INTO tmp\_IDAula

**FROM** Aula A;

:NEW.IDAula :**=** tmp\_IDAula;

END IF;

END;

**/**

*Capitolo 4 Progettazione fisica*

# 4.3.19 Tabella: ControlloMedico

**CREATE** **TABLE** ControlloMedico (

IDControlloMedico INTEGER ,

NomeMedico VARCHAR2(64) DEFAULT **ON** **NULL** 'Non specificato',

NomeStruttura VARCHAR2(64) DEFAULT **ON** **NULL** 'Non specificato',

Tipo VARCHAR2(32) **NOT NULL**,

Esito VARCHAR2(32),

Data DATE **NOT NULL**,

FK\_IDLuogo INTEGER **NOT NULL**,

FK\_IDPersona CHAR(16) **NOT NULL**

);

-- GESTIONE VALORE NULL DEL PRIMARY KEY

CREATE **OR** REPLACE TRIGGER SET\_IDControlloMedico

BEFORE INSERT **ON** ControlloMedico

FOR EACH ROW

DECLARE

tmp\_IDControlloMedico ControlloMedico.IDControlloMedico%TYPE;

**BEGIN**

IF (:NEW.IDControlloMedico IS **NULL**) THEN

**SELECT** NVL(**MAX**(C.IDControlloMedico), 0) **+** 1 INTO tmp\_IDControlloMedico

**FROM** ControlloMedico C;

:NEW.IDControlloMedico :**=** tmp\_IDControlloMedico;

END IF;

END;

-- IMPLEMENTAZIONE VINCOLO VD\_DATA\_CONTROLLOMEDICO

CREATE **OR** REPLACE TRIGGER CHECK\_DATA\_CONTROLLOMEDICO

BEFORE INSERT **OR** UPDATE **ON** ControlloMedico

FOR EACH ROW

**BEGIN**

IF(:NEW.DATA **>=** SYSDATE )

THEN

RAISE\_APPLICATION\_ERROR( -20111, 'Invalida data inserita');

END IF;

END;

*Capitolo 4 Progettazione fisica*

-- VINCOLI: ControlloMedico

**ALTER** **TABLE** ControlloMedico

ADD(

-- CHIAVE PRIMARIA

CONSTRAINT PK\_ControlloMedico PRIMARY KEY (IDControlloMedico) ,

-- CHIAVE ESTERNA

CONSTRAINT FK\_ControlloMedico\_Luogo FOREIGN KEY(FK\_IDLuogo)

REFERENCES Luogo(IDLuogo) ON DELETE **SET** **NULL** ,

-- CHIAVE ESTERNA

CONSTRAINT FK\_ControlloMedico\_Persona FOREIGN KEY(FK\_IDPersona)

REFERENCES Persona(CodiceFiscale) ,

-- Enumerazione per Tipo(TipoControllo)

CONSTRAINT EnumTipoControllo

CHECK (Tipo **IN** ('Tampone', 'TestSierologico', 'Altro')) ,

-- Enumerazione per Esito(TipoEsito)

CONSTRAINT EnumTipoEsito

CHECK (Esito **IN** ('Positivo', 'Negativo', 'In attesa')) ,

-- IMPLEMENTAZIONE VINCOLO VE\_TipoEsito\_ControlloMedico

CONSTRAINT VE\_TipoEsito\_ControlloMedico

CHECK ((Tipo **<>** 'Altro' **OR** Esito **IS** **NULL**) **AND** (Esito **IS NOT** **NULL** **OR** Tipo **=** 'Altro')),

-- IMPLEMENTAZIONEà VINCOLO VD\_NomeMedico\_ControlloMedico

CONSTRAINT VD\_NomeMedico\_ControlloMedico

CHECK (NomeMedico **LIKE** '\_% \_%')

);

*Capitolo 4 Progettazione fisica*

# 4.3.20 Tabella: Frequentazione

**CREATE** **TABLE** Frequentazione (

FK\_IDStudente INTEGER **NOT NULL**,

FK\_IDAmbienteDiStudio INTEGER **NOT NULL**

);

-- VINCOLI: Frequentazione

**ALTER** **TABLE** Frequentazione

ADD(

-- CHIAVE ESTERNA

CONSTRAINT FK\_Frequentazione\_Studente FOREIGN KEY(FK\_IDStudente)

REFERENCES Studente(IDStudente) ON DELETE CASCADE ,

-- CHIAVE ESTERNA

CONSTRAINT FK\_Frequentazione\_AmbienteDiStudio FOREIGN KEY(FK\_IDAmbienteDiStudio)

REFERENCES AmbienteDiStudio(IDAmbienteDiStudio) ON DELETE CASCADE ,

-- IMPLEMENTAZIONE VINCOLO VU\_FrequentazioneUnica

CONSTRAINT FrequentazioneUnica UNIQUE (FK\_IDStudente,FK\_IDAmbienteDiStudio)

);

*Capitolo 4 Progettazione fisica*

# 4.3.21 Tabella: Gestione

**CREATE** **TABLE** Gestione(

FK\_IDGestore CHAR(16) **NOT NULL**,

FK\_IDLuogoGestito INTEGER **NOT NULL**

);

-- VINCOLI: Gestione

**ALTER** **TABLE** Gestione

ADD(

-- CHIAVE ESTERNA

CONSTRAINT FK\_Gestione\_Gestore FOREIGN KEY(FK\_IDGestore)

REFERENCES Persona(CodiceFiscale) ON DELETE CASCADE ,

-- CHIAVE ESTERNA

CONSTRAINT FK\_Gestione\_LuogoGestito FOREIGN KEY(FK\_IDLuogoGestito)

REFERENCES Luogo(IDLuogo) ON DELETE CASCADE ,

-- IMPLEMENTAZIONE VINCOLO VU\_GestioneUnica

CONSTRAINT VU\_GestioneUnica UNIQUE (FK\_IDGestore,FK\_IDLuogoGestito)

);

*Capitolo 5*

Manuale d’uso

# **5.1 Introduzione al manuale d’uso**

Nel seguente capitolo verranno mostrate e spiegate le modalità per interagire con la base di dati tramite l’apposita interfaccia.

Le possibili operazioni che si possono effettuare tramite l’interfaccia sono :

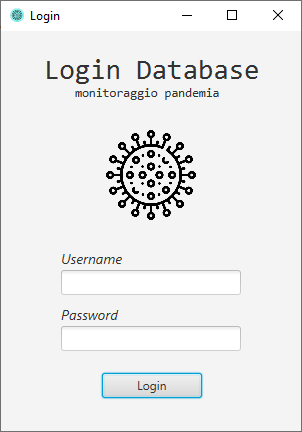
* Creare un record
* Modificare un record esistente
* Ricercare dei record

Durante le comuni operazioni possono chiaramente essere riportati errori dovuti ai vari vincoli di integrità, quindi per chiarezza di seguito saranno riportati i messaggi di errori con la relativa spiegazione del perché è comparso.

|  |  |
| --- | --- |
| Messaggio di errore | Spiegazione |
| ORA-01400: impossibile inserire NULL in ("..."."..."."NomeCampo") | Il seguente errore viene visualizzato quando il campo "NomeCampo" non è stato compilato nonostante quest'ultimo fosse obbligatorio per portare a termine correttamente l'operazione.  Per risolverlo bisogna dunque ripetere l'operazione compilando correttamente il campo. |
| Attenzione ambiguità: Ci sono più ... con i dati inseriti, ripetere l'operazione specificando anche il ... | Il seguente errore viene visualizzato quando è stata riscontrata un’ambiguità.  L'ambiguità è una situazione in cui ci sono piu entità che corrispondono ai campi utilizzati fino a quel momento.  Es. Vogliamo modificare uno studente. Specificando solo il nome dello studente potremmo andare in contro ad un'ambiguità poichè ci sono possono essere piu persone con lo stesso nome.  Per risolvere il problema sarà necessario ripetere l'operazione specificando altri campi, oppure indicando il codice identificativo dell'entità. |
| ORA-00001: violata restrizione di unicità (...) | Il seguente errore viene visualizzato quando si sta provando ad inserire un’entità già esistente e per la quale è stata definita un vincolo di unicità. |
| ORA-02290: restrizione di controllo violata | Il seguente errore viene visualizzato quando si prova ad inserire i valori in un campo senza rispettare il formato corretto. |

*Capitolo 5 Manuale d’uso*

# **5.2 Introduzione all’interfaccia: Login**

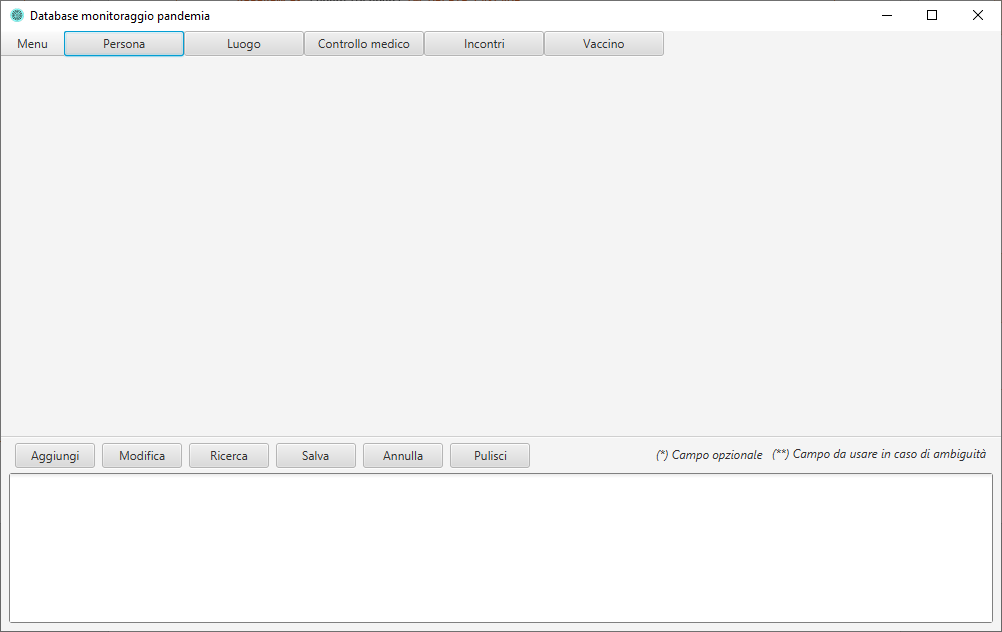


*Fig.* 1

Schermata di login, sono richiesti i campi *Username* e *Password* per accedere alla schermata primaria.

# **5.3 Introduzione all’interfaccia: Schermata primaria**

2



*Fig.* 2

1

4

3

**1** Il menù dov’è possibile effettuare il ***logout*** o accedere alla schermata***help.***

**2** Raggruppamento in macrocategorie per accedere alla gestione del database.

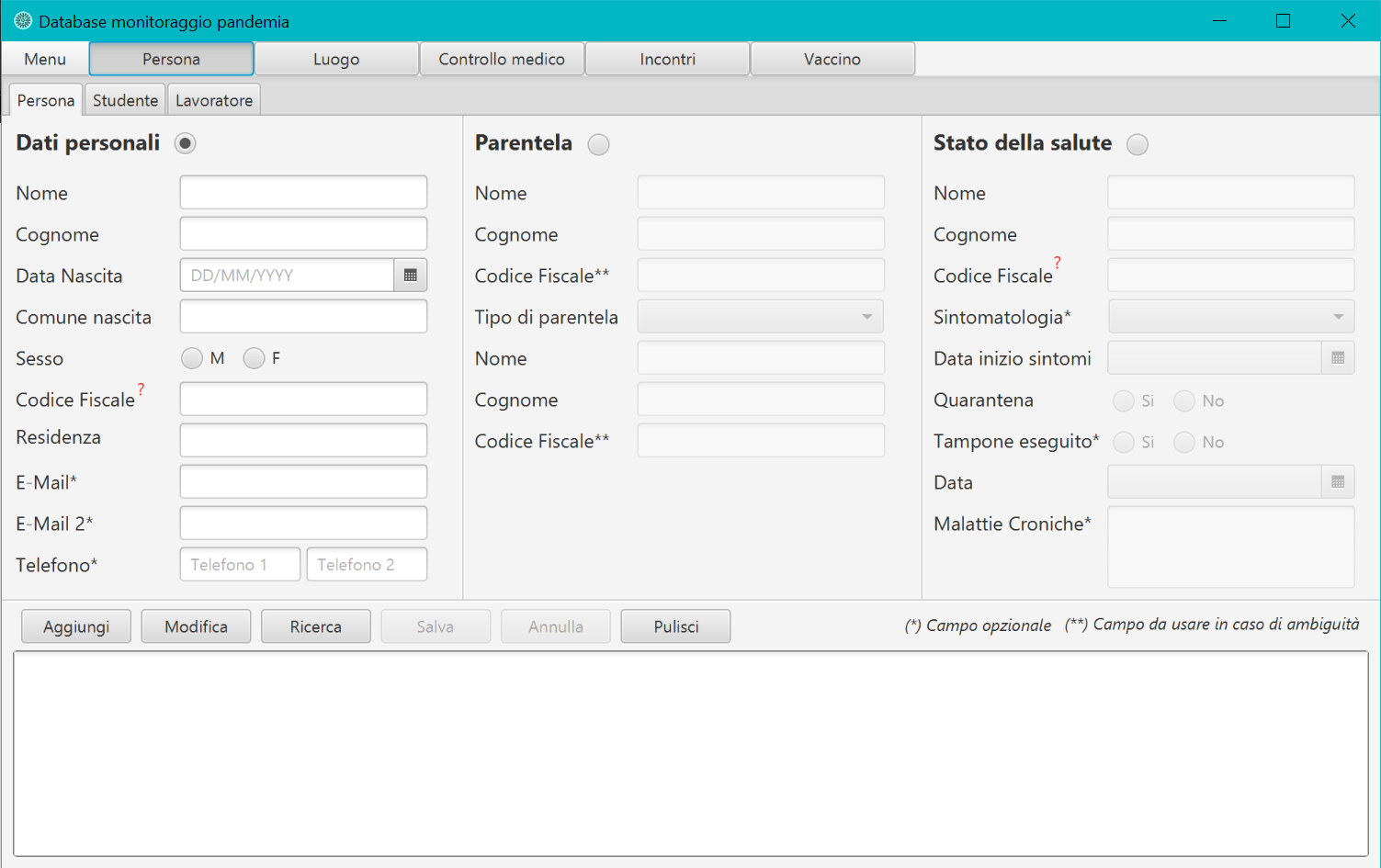
**3** I pulsanti per effettuare le principali operazioni.

**4** La console dove vengono mostrati i risultati delle operazioni effettuate.

*Capitolo 5 Manuale d’uso*

# **5.4 schermata di selezione**

In questo paragrafo verrano descritte le procedure da eseguire per aggiungere un record all’interno del database.



3**.**

2**.**

*Fig.* 3

1**.**

*Solo a titolo di esempio è stato fatto riferimento alla gestione di una persona, ma gli stessi passi possono essere seguiti per una qualsiasi entità*

*Capitolo 5 Manuale d’uso*

# **5.5 Aggiunta di un record**

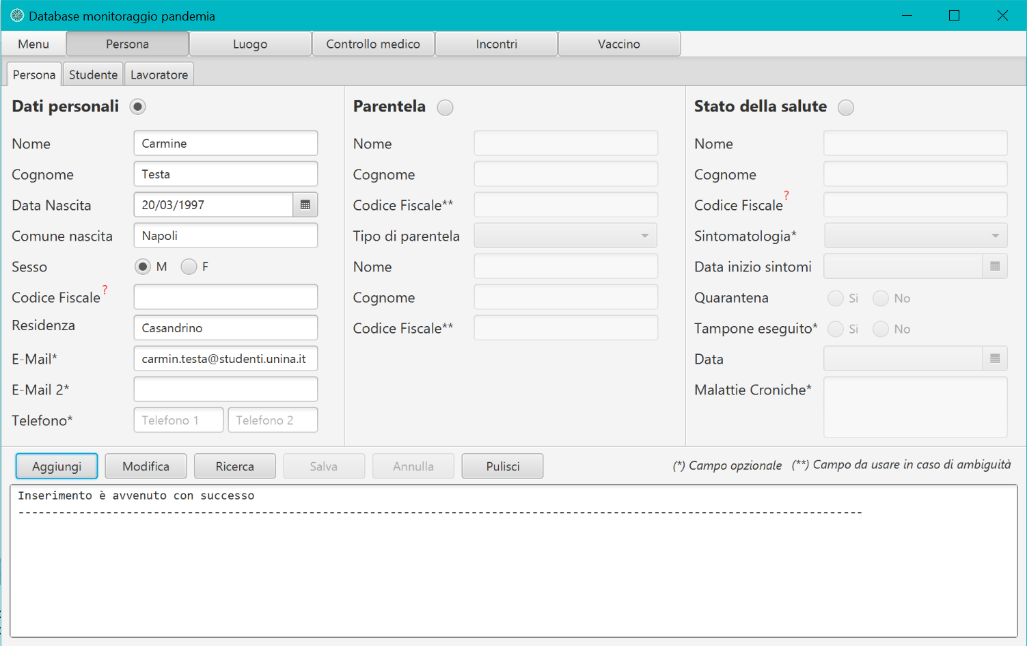
1. **SELEZIONARE LA MACROCATEGORIA** : la prima cosa da fare è selezionare la macrocategoria a cui appartiene l’entità che vogliamo aggiungere al database. Nell’esempio corrente verrà selezionata “Persona”(Fig. 3).
2. **SELEZIONARE UNA CLASSE** : dopo aver selezionato la macrocategoria apparirà una schermata contenente tutte le classi che ne fanno riferimento. Bisogna dunque selezionare la classe che rappresenta l’entità da aggiungere. In questo caso tra Lavoratore, Studente e Persona verrà scelta “Persona”(Fig. 3).
3. **SELEZIONARE GLI ATTRIBUTI** : una volta selezionata una classe, quest’ultima potrà avere piu attributi legati ad essa. In particolare, per una persona oltre ad aggiungerla, è possibile aggiungere una parentela tra due familiari oppure aggiungere la condizione clinica di una persona selezionando “Stato della salute”. Nel caso specifico in quanto si vuole aggiungere una persona generica bisogna selezionare “Dati personali”(Fig. 3).

(La selezione degli attributi è possibile tramite il pulsante di selezione accanto al titolo)

1. **COMPILARE I CAMPI** : selezionati gli attributi bisogna dunque compilarli (almeno quelli non opzionali)
2. **AGGIUNGERE** : arrivati a questo punto basterà premere “Aggiungi” è nel caso di successo sarà riportato un messaggio nella console con il successo dell’operazione, in caso contrario verranno riportati i vari messaggi di errori che possono essere uno di quelli descritti in precedenza.

**Note**

Per comodità è possibile anche non inserire il codice fiscale. Se così fosse sarà automaticamente generato in seguito.



*Fig.* 4

*Capitolo 5 Manuale d’uso*

# **5.6 Modifica di un record**

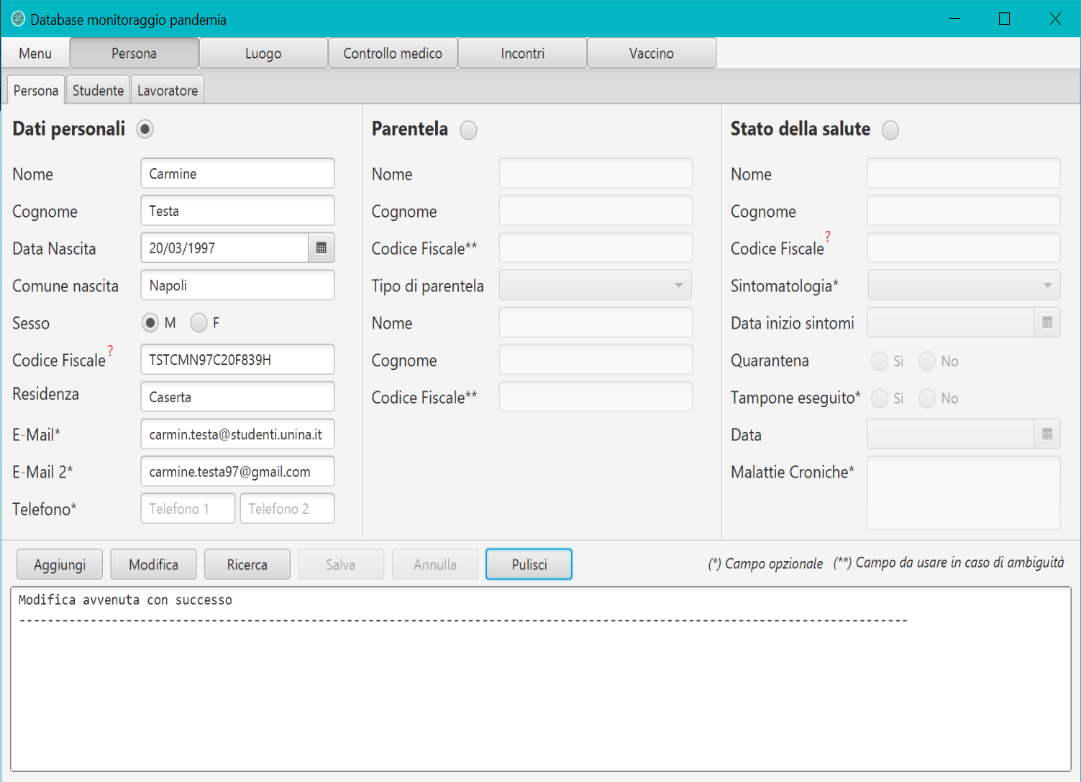
1. **SELEZIONARE LA MACROCATEGORIA** : la prima cosa da fare è selezionare la macrocategoria a cui appartiene l’entità che vogliamo aggiungere al database. Nell’esempio corrente verrà selezionata “Persona”(Fig. 3).
2. **SELEZIONARE UNA CLASSE** : dopo aver selezionato la macrocategoria apparirà una schermata contenente tutte le classi che ne fanno riferimento. Bisogna dunque selezionare la classe che rappresenta l’entità da modificare. In questo caso tra Lavoratore, Studente e Persona verrà scelta “Persona”(Fig. 3).
3. **SELEZIONARE GLI ATTRIBUTI** : una volta selezionata una classe, quest’ultima potrà avere piu attributi legati ad essa. Nel caso specifico in quanto si vuole modifica una persona generica bisogna selezionare “Dati personali”(Fig. 3).

(La selezione degli attributi è possibile tramite il pulsante di selezione accanto al titolo)

1. **SELEZIONARE UN RECORD** : ora bisogna specificare un record da modificare compilando il campo che lo identifica. Nel caso specifico dato che si vuole modificare una persona sarà necessario compilare il campo “Codice fiscale” della persona che si vuole modificare.

(Generalmente il campo da compilare sarà identificato con un “?” accanto al campo)

1. **MODIFICA** : compilato correttamente il campo che identifica il record bisogna premere modifica. Una volta premuto modifica, nel caso esista un record corrispondente alla chiave inserita, verranno automaticamente riempiti i campi e si potranno modificarli. In questo esempio è stata cambiata residenza da Casandrino in Caserta ed è stata aggiunta una seconda mail.
2. **SALVA** : soddisfatti dei cambiamenti apportati sarà necessario salvare le modifiche tramite il tasto “Salva” e se il cambiamento è andato a buon fine sarà mostrato un messaggio nella console con l’imminente successo, in caso contrario verranno riportati i vari messaggi di errori che possono essere uno di quelli descritti in precedenza. Esiste anche la possibilità di annullare il cambiamento tramite il pulsante “Annulla”.



*Fig.* 5

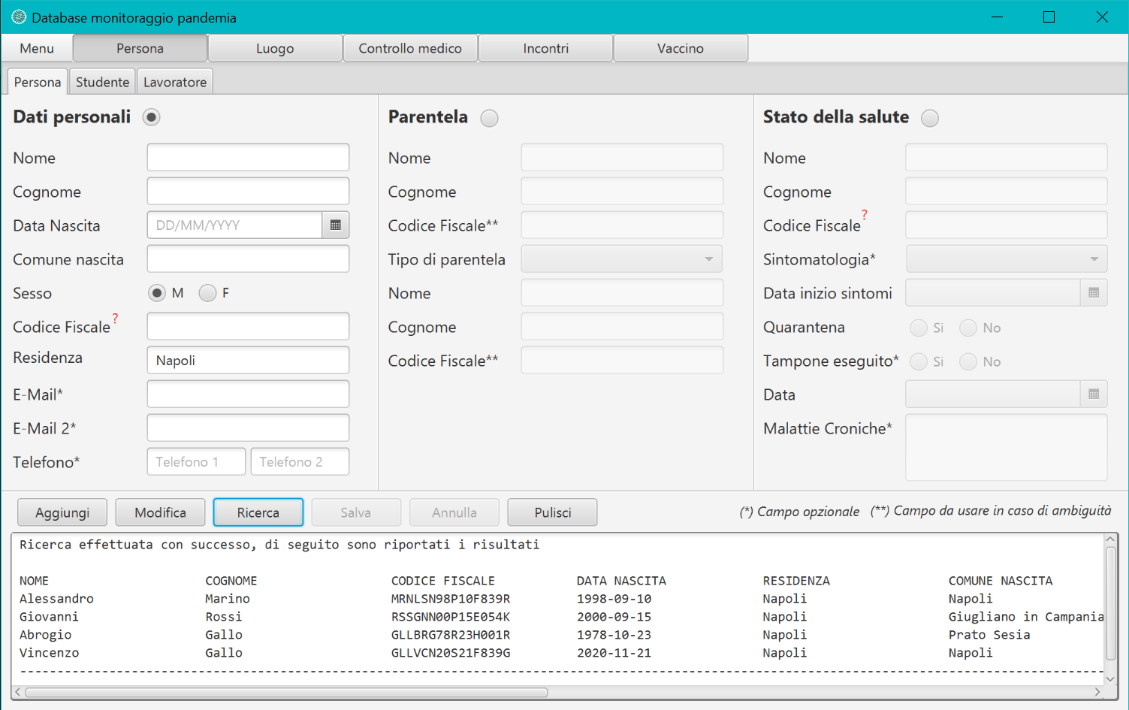
*Capitolo 5 Manuale d’uso*

# **5.7 Ricerca**

1. **SELEZIONARE LA MACROCATEGORIA** : la prima cosa da fare è selezionare la macrocategoria a cui appartiene l’entità che vogliamo aggiungere al database. Nell’esempio corrente verrà selezionata “Persona”(Fig. 3).
2. **SELEZIONARE UNA CLASSE** : dopo aver selezionato la macrocategoria apparirà una schermata contenente tutte le classi che ne fanno riferimento. Bisogna dunque selezionare la classe che rappresenta l’entità da ricercare. In questo caso tra Lavoratore, Studente e Persona verrà scelta “Persona”(Fig. 3).
3. **SELEZIONARE GLI ATTRIBUTI** : una volta selezionata una classe, quest’ultima potrà avere piu attributi legati ad essa. Nel caso specifico in quanto si vuole ricercare delle persone generiche bisogna selezionare “Dati personali”(Fig. 3).

(La selezione degli attributi è possibile tramite il pulsante di selezione accanto al titolo)

1. **COMPILARE I CAMPI** : selezionati gli attributi, bisogna compilare quelli su cui si vuole fare la ricerca. Inserendo alcuni campi il sistema applicherà un filtro con i parametri inseriti, restituendo i record che soddisfano la richiesta. Come esempio suppongo di voler effettuare una ricerca tra le persone e come parametro inserisco “M” e residenza “Napoli” in questo modo come risultato avrò tutte le persone di sesso maschile con residenza a Napoli.
2. **RICERCA** : per ottenere i risultati dalla ricerca sarà necessario premere “Ricerca”.



*Fig.* 6