





**Corso di laurea in Informatica  
Insegnamento di Basi di Dati I  
Anno accademico 2021/2022**

**Progettazione e sviluppo di una base di dati relazionale per la gestione di un sistema di tracciamento contatti per ristoranti**

*Docenti:*

Prof. Adriano Peron

Prof. Sergio Di Martino

Prof. Luigi Libero Lucio Starace

*Autori:*

Angelo Di Maio

Matricola N86003699

*angelo.dimaio3@studenti.unina.it*

Santolo Barretta

Matricola N86003666

*san.barretta@studenti.unina.it*

23 dicembre 2021

*Questa pagina è stata lasciata intenzionalmente bianca.*

# **Indice**

[Indice 3](#_Toc91695485)

[Capitolo 1 4](#_Toc91695486)

[Descrizione del progetto 4](#_Toc91695487)

[1.1 Descrizione sintetica e analisi del problema 4](#_Toc91695488)

[1.2 L’organizzazione del sistema di tracciamento contatti 4](#_Toc91695489)

[Capitolo 2 6](#_Toc91695490)

[Progettazione concettuale 6](#_Toc91695491)

[Introduzione 6](#_Toc91695492)

[2.1 Alcune convenzioni per la lettura dei Class Diagram 6](#_Toc91695493)

[2.2 Class diagram 7](#_Toc91695494)

[2.3 Ristrutturazione del class diagram 8](#_Toc91695495)

[2.3.1 Analisi delle informazioni ridondanti 8](#_Toc91695496)

[2.3.2 Analisi degli identificativi 8](#_Toc91695497)

[2.3.3 Rimozione degli attributi strutturati 8](#_Toc91695498)

[2.3.4 Rimozione delle classi di associazione 9](#_Toc91695499)

[2.3.5 Rimozione delle gerarchie di specializzazione 9](#_Toc91695500)

[2.3.5 Class diagram ristrutturato 10](#_Toc91695501)

[2.4 Dizionario dei dati **Errore. Il segnalibro non è definito.**](#_Toc91695502)

[2.4.1 Dizionario delle classi **Errore. Il segnalibro non è definito.**](#_Toc91695503)

[2.4.2 Dizionario delle associazioni **Errore. Il segnalibro non è definito.**](#_Toc91695504)

[2.4.3 Dizionario dei vincoli **Errore. Il segnalibro non è definito.**](#_Toc91695505)

[2.4.4 Dizionario delle interrogazioni **Errore. Il segnalibro non è definito.**](#_Toc91695506)

# **Capitolo 1**

# **Descrizione del progetto**

## **1.1 Descrizione sintetica e analisi del problema**

Si provvederà alla progettazione e allo sviluppo di una base di dati relazionale per la gestione del tracciamento di contatti COVID-19 in ristoranti. Il sistema permetterà di tenere traccia dei contatti di un avventore in un ristorante, fornendo ulteriori informazioni per il tracciamento degli eventuali avventori risultati positivi al virus SARS-CoV-2. In particolare, il sistema deve permettere di gestire il tracciamento dei contatti di uno o più ristoranti. Ciascun ristorante è organizzato in una o più sale, identificate da un nome, le quali contengono uno o più tavoli. Ciascun tavolo presente all’interno di una sala sarà identificato da un codice univoco, avrà l’indicazione del numero massimo di avventori che possono sedersi a quel tavolo e dei tavoli adiacenti ad esso. Il sistema viene gestito da operatori che tengono traccia delle tavolate formate dagli avventori nei ristoranti. Un avventore sarà identificato da un nome, un cognome, un numero di carta d’identità e un numero di telefono. La tavolata viene identificata dalla data di arrivo degli avventori, i camerieri che l’hanno servita e il tavolo ad essa assegnata. I ristoranti effettueranno soltanto servizio serale nella fascia oraria 20 – 22, quindi in ciascuna data, al più una tavolata deve essere associata ad un dato tavolo, poiché il servizio è unico. Inoltre il numero di avventori che partecipano ad una tavolata non deve superare il numero massimo di avventori che possono sedersi al tavolo a cui essa è assegnata. Infine il sistema dovrà permettere la visualizzazione di alcune statistiche su base mensile e giornaliera, tra cui il numero totale di avventori accolti da ciascun ristorante.

## **1.2 L’organizzazione del sistema di tracciamento contatti**

Di seguito verrà descritta brevemente la **chiave di lettura** data al sistema da implementare, sulla quale si basa la progettazione della base di dati relazionale.

Si suppone che il sistema di tracciamento contatti COVID-19 venga utilizzato da uno ristorante o più ristoranti appartenenti ad una stessa catena, ubicati fisicamente nel territorio italiano e con gruppi di avventori accolti prevalentemente italiani. Il sistema verrà gestito dalla figura dell’Operatore il quale, una volta autenticato nella piattaforma sviluppata, con un Username e una Password, si occuperà di alcuni aspetti rilevanti, fra cui:

* **la gestione e l’inserimento nella piattaforma di tavolate di avventori**: si intende come Tavolata l’atto di associare i tavoli presenti in una sala del ristorante ad un singolo o ad un gruppo di avventori. La tavolata verrà identificata quindi da una data di arrivo degli avventori facenti parte della stessa e da un orario di arrivo e uno di uscita, che coincideranno sempre con quelli dell’unico servizio serale effettuato (nella fascia oraria 20-22). L’Operatore che gestisce la piattaforma si occuperà poi di registrare anche le informazioni sugli avventori facenti parte di una determinata tavolata, tra cui: le informazioni anagrafiche, la temperatura registrata dall’avventore al suo arrivo al ristorante (se oltre i 37.5° non sarà possibile registrare l’Avventore nel sistema) ed inoltre se possiede o meno green pass (se un Avventore non possiede green pass non sarà possibile associarlo ad un tavolo presente in una sala Interna ma solo ad un tavolo presente in una sala Esterna, se il ristorante ne è dotato). In questo modo gli operatori che gestiscono il sistema terranno traccia di tutte le tavolate formate in un ristorante;
* **la gestione e l’inserimento nella piattaforma di casi COVID-19 avvenuti nel ristorante**: nel caso in cui un avventore registrato ad una tavolata in un determinato ristorante risultasse positivo nei giorni seguenti al virus SARS-CoV-2, dopo sua preventiva segnalazione al ristorante (ad esempio tramite form online o telefonata), sarà compito degli operatori aprire un nuovo Caso nella piattaforma e gestirlo. La gestione di un Caso comprende l’inserimento in piattaforma dei dati dell’avventore risultato positivo, quindi la sua anagrafica, la data di registrazione del caso, il suo stato e delle eventuali note associate al caso. Lo stesso vale anche per eventuali camerieri risultati positivi. Si suppone che lo stato di un caso possa essere “NonRisolto” quando viene solamente inserito in piattaforma, “InRisoluzione” quando un operatore prenderà in carico il caso e avviserà tutti gli avventori che hanno avuto contatti con l’avventore risultato positivo, “Risolto” quando saranno stati avvisati tutti gli avventori venuti in contatto con l’avventore positivo. In questo modo il sistema permetterà di tenere traccia dei contatti di un avventore in un ristorante.

La figura dell’Operatore che gestisce l’intero sistema descritto può essere svolta da:

* uno o più Manager che gestiscono un determinato ristorante, i quali avranno dei privilegi limitati nel sistema relativi solo alla gestione di casi e all’inserimento di tavolate;
* direttamente dal Proprietario che amministra il ristorante o la catena formata da più ristoranti ed il quale avrà privilegi “*ad hoc*” nel sistema come la possibilità di aggiungere, modificare o rimuovere ristoranti, sale, tavoli presenti in piattaforma, oltre che la gestione del personale presente per quanto riguarda le figure del Manager e del Cameriere.

Per ulteriori chiarimenti generali si rimanda al successivo capitolo riguardante la **progettazione concettuale**, in cui è definito il **class diagram** della base di dati e viene riportato il **dizionario dei dati**.

# **Capitolo 2**

# **Progettazione concettuale**

## **Introduzione**

In questo capitolo viene affrontata la progettazione della base di dati al livello di astrazione più alto, ossia quello della progettazione concettuale. Dall’analisi iniziale del problema e dei requisiti che devono essere soddisfatti si arriverà ad uno schema concettuale indipendente dalla struttura dei dati e dall’implementazione fisica della base di dati. Lo schema concettuale verrà rappresentato usando un class diagram UML, in cui saranno presenti le entità rilevanti individuate ai fini della rappresentazione dei dati e le relazioni che intercorrono tra esse.

## **2.1 Alcune convenzioni per la lettura dei class diagram**

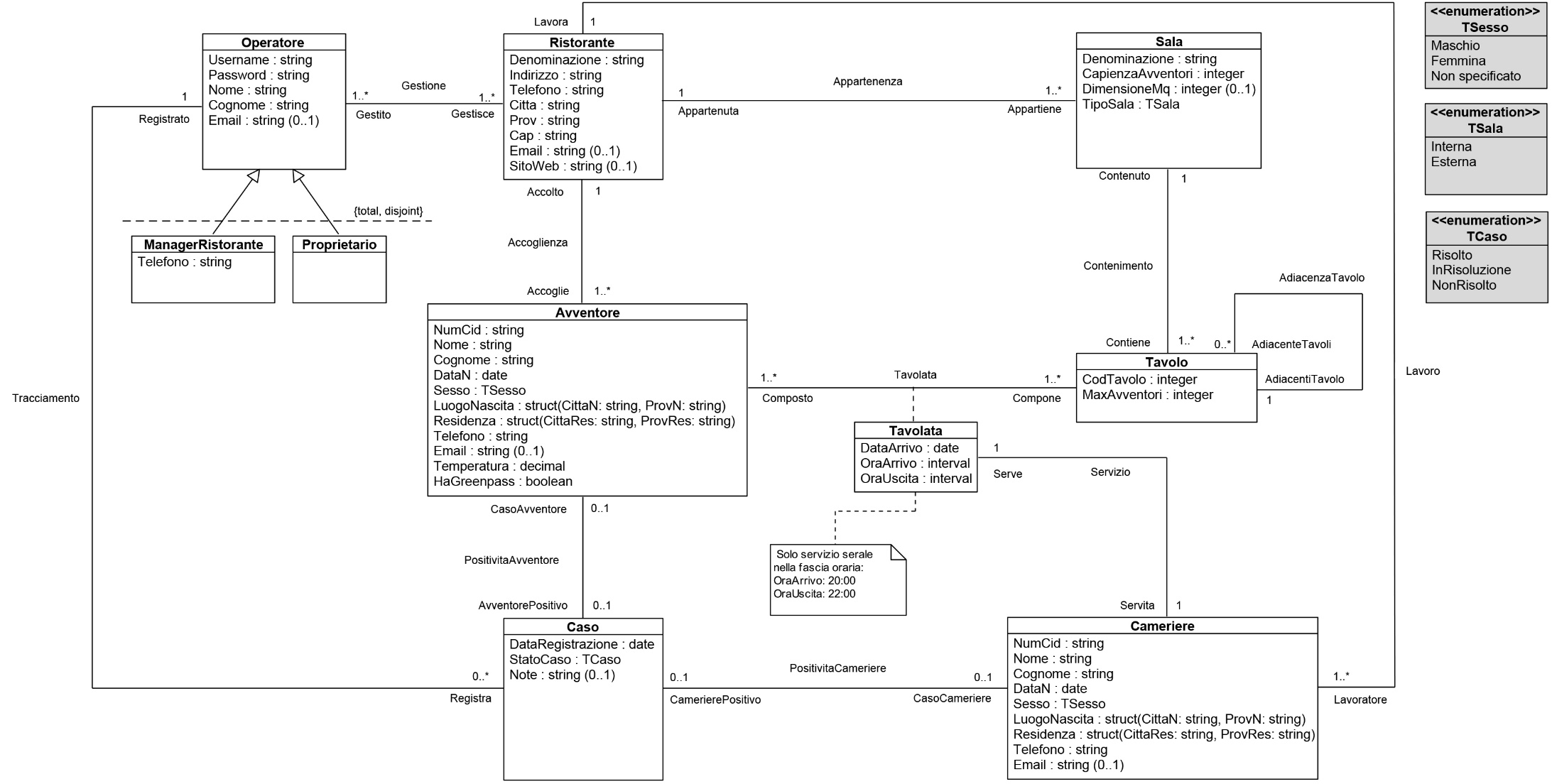
Al fine di semplificare la lettura dei class diagram UML che seguono si è scelto di adottare le seguenti convenzioni:

* tutti gli attributi, salvo quelli in cui è specificata, hanno cardinalità (1,1) o brevemente 1, ossia sono attributi a valore singolo totali;
* per le associazioni: la cardinalità di una partecipazione è situata a destra se la linea di collegamento è verticale, in alto se la linea di collegamento è orizzontale;
* le enumerazioni sono identificabili dal colore grigio.

**NOTA -** Nel caso in cui i class diagram non risultassero essere abbastanza leggibili si riportano di seguito i link alle rispettive versioni digitali esterne:

* **Class diagram:** <https://ibb.co/7WC7zWC>
* **Class diagram ristrutturato:** <https://ibb.co/VpHcWvx>

## **2.2 Class diagram**



**Immagine 2.1**: Class diagram della base di dati

## **2.3 Ristrutturazione del class diagram**

Al fine di rendere il class diagram conforme al modello relazionale che sarà adottato successivamente nella fase di progettazione logica e per migliorare l’efficienza dell’implementazione fisica si procede alla ristrutturazione dello stesso. Al termine del procedimento di ristrutturazione il class diagram non conterrà attributi strutturati e gerarchie di specializzazione.……………………………………

## **2.3.1 Analisi delle informazioni ridondanti**

Sono presenti due informazioni ridondanti nel class diagram di partenza. La prima riguarda gli attributi OraArrivo e OraUscita appartenenti alla classe di associazione Tavolata. Tali attributi avranno sempre lo stesso valore per ogni istanza di Tavolata, poiché dall’analisi dei requisiti sappiamo che ogni singolo ristorante effettuerà soltanto servizio serale nella fascia oraria OraArrivo: 20:00, OraUscita: 22:00. Per completezza si è deciso di non rimuovere i suddetti attributi nella ristrutturazione del class diagram. La seconda informazione ridondante riguarda l’attributo CapienzaAvventori appartenente all’entità Sala. E’ infatti possibile ottenere la CapienzaAvventori di una determinata Sala sommando l’attributo MaxAvventori dell’entità Tavolo, per tutti i tavoli appartenenti alla Sala presa in considerazione. Anche in questo caso, per completezza, si è deciso di non rimuovere l’attributo nella fase di ristrutturazione poiché si è pensato che in ottica pandemia e tracciamento contatti, la capienza del numero di avventori per una determinata sala rimanga fissa per un lungo periodo di tempo nello schema della base di dati e non variabile a seconda dei tavoli che vengono associati ad una sala. In conclusione per ogni Sala avremo una capienza fissa ed un massimo numero di tavoli che possono essere ad essa associati, la cui somma totale di MaxAvventori ovviamente dovrà essere minore o uguale alla CapienzaAvventori della Sala.

## **2.3.2 Analisi degli identificativi**

Per alcune entità del class diagram è stata decisa l’introduzione di chiavi primarie “surrogate” in modo da evitare l’impiego di chiavi candidate composte da più attributi. Tali chiavi primarie saranno degli identificativi numerici che permetteranno di discriminare con maggiore facilità le istanze.  
Le chiavi primarie surrogate introdotte sono le seguenti: CodProprietario, CodManager, CodRistorante, CodSala, CodTavolo, CodTavolata. Le entità Avventore e Cameriere saranno identificate dalla chiave primaria NumCid ossia una stringa che corrisponde al numero di carta d’identità, univoco per ogni persona.

## **2.3.3 Rimozione degli attributi strutturati**

È necessario gestire gli attributi strutturati LuogoNascita e Residenza presenti rispettivamente nelle entità Avventore e Cameriere. È stato deciso di estendere gli attributi che formano gli strutturati LuogoNascita e Residenza nell’entità cui appartengono, rimuovendo questi ultimi. Dunque alle entità Avventore e Cameriere verranno aggiunti i seguenti attributi: CittaN, ProvN (che costituiscono l’attributo strutturato LuogoNascita) e CittaRes, ProvRes (che costituiscono l’attributo strutturato Residenza).

È stata scelta questa gestione poiché tali attributi potrebbero essere utili per alcune interrogazioni riguardanti statistiche sul tracciamento dei contatti (ad esempio per stabilire quante persone di una determinata CittaRes siano risultate positive in un determinato ristorante, su base giornaliera o mensile).

## **2.3.4 Rimozione delle classi di associazione**

Si procede con l’eliminazione della classe di associazione Tavolata e la reintroduzione della stessa come entità, contenente gli stessi attributi della classe di associazione con l’aggiunta di un identificativo numerico, CodTavolata. Sono state introdotte dunque due nuove associazioni:

* Partecipazione fra le entità Avventore e Tavolata;
* Composizione fra le entità Tavolo e Tavolata.

Ciò faciliterà la creazione e la gestione delle tavolate di avventori.

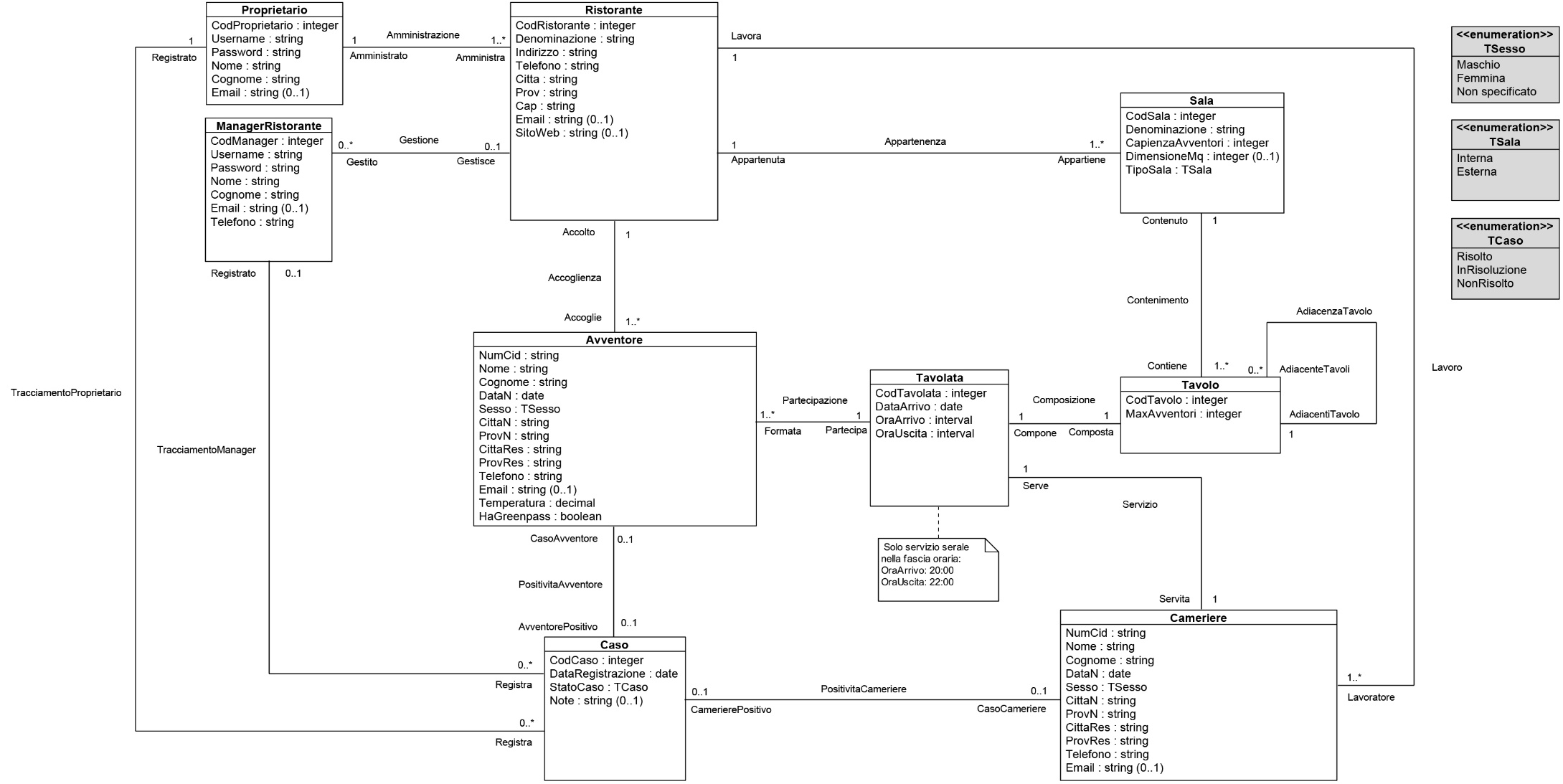
2.3.5 Rimozione delle gerarchie di specializzazione

Si procede con l’eliminazione della specializzazione riguardante l’entità Operatore che può essere specializzata in Proprietario oppure ManagerRistorante. Si tratta di una specializzazione totale e disgiunta; per tanto si procede alla sua eliminazione attraverso la “compressione” della superclasse nelle sottoclassi, le quali ereditano gli attributi e le associazioni della superclasse Operatore. Sono quindi state aggiunte le associazioni:

* Amministrazione e Gestione (in sostituzione della precedente, Gestione);
* TracciamentoProprietario e TracciamentoManager (in sostituzione della precedente, Tracciamento).

In questo modo la figura dell’operatore generico che gestirà la piattaforma per il tracciamento verrà dettagliata maggiormente, così come descritto al punto [**1.2**](#_1.2_L’organizzazione_del).

## **2.3.5 Class diagram ristrutturato**

****

**Immagine 2.2**: Class diagram ristrutturato della base di dati

2.4 Dizionario dei dati per il class diagram ristrutturato

## **2.4.1 Dizionario delle classi**

|  |  |  |
| --- | --- | --- |
| *Classe* | *Descrizione* | *Attributi* |
| Proprietario | Descrittore di un generico proprietario di uno o più ristoranti | **CodProprietario** (*integer*):chiave surrogata che serve ad identificare un proprietario.  **Username** (*string*): username associato ad un proprietario.  **Password** (*string*): password associata ad un proprietario.  **Nome** (*string*): nome di un proprietario.  **Cognome** (*string*): cognome di un proprietario.  **Email** (*string, opzionale*): email associata ad un proprietario. |
| ManagerRistorante | Descrittore del manager di un ristorante. | **CodManager** (*integer*): chiave surrogata che serve ad identificare univocamente un manager di un ristorante.  **Username** (*string*): username associato al manager.  **Password** (*string*): password associata al manager.  **Nome** (*string*): nome di un manager.  **Cognome** (*string*): cognome associato di un manager.  **Email** (*string, opzionale*): email associata al manager.  **Telefono** (*string*): numero di telefono associato al manager. |
| Ristorante | Descrittore di un ristorante. | **CodRistorante** (*integer*): chiave surrogata che serve ad identificare univocamente un ristorante.  **Denominazione** (*string*): nome di un ristorante.  **Indirizzo** (*string*): indirizzo di un ristorante.  **Telefono** (*string*): numero di telefono di un ristorante.  **Citta** (*string*): città di ubicazione di un ristorante.  **Prov** (*string*): provincia di ubicazione di un ristorante.  **Cap** (*string*): cap della provincia di ubicazione di un ristorante.  **Email** (*string, opzionale*): email associata ad un ristorante.  **SitoWeb** (*string, opzionale*): sito web associato ad un ristorante. |
| Sala | Descrittore della sala di un ristorante. | **CodSala** (*integer*): chiave surrogata che serve ad identificare univocamente una sala.  **Denominazione** (*string*): nome di una sala.  **CapienzaAvventori** (*integer*): capienza massima di avventori per una sala.  **DimensioneMq** (*string, opzionale*): dimensione in metri quadri di una sala.  **TipoSala** (*TSala*): attributo che indica se la sala è interna oppure esterna. |
| Tavolo | Descrittore di un tavolo del ristorante. | **CodTavolo** (*integer*): chiave surrogata che serve ad identificare univocamente un tavolo.  **MaxAvventori** (*integer*): numero massimo di avventori per ogni tavolo. |
| Tavolata | Descrittore della tavolata di avventori. | **CodTavolata** (*integer*): chiave surrogata che serve ad identificare univocamente ciascuna tavolata.  **DataArrivo** (*date*): data in cui si svolge la tavolata.  **OraArrivo** (*interval*): orario di arrivo della tavolata (vale sempre 20:00).  **OraUscita** (*interval*): orario di uscita della tavolata (vale sempre 22:00). |
| Avventore | Descrittore di un generico avventore. | **NumCid** (*string*): numero della carta di identità che serve anche da chiave primaria per identificare univocamente ciascun avventore.  **Nome** (*string*): nome di un avventore.  **Cognome** (*string*): cognome di un avventore.  **DataN** (*date*): data di nascita di un avventore.  **Sesso** (*TSesso*): sesso di un avventore.  **CittaN** (*string*): città di nascita di un avventore.  **ProvN** (*string*): provincia di nascita di un avventore.  **CittaRes** (*string*): città di residenza di un avventore.  **ProvRes** (*string*): provincia di residenza di un avventore.  **Telefono** (*string*): numero di telefono di un avventore.  **Email** (*string, opzionale*): email associata ad un avventore.  **Temperatura** (*decimal*): temperatura misurata all’ingresso di un ristorante.  **HaGreenpass** (*boolean*): indica il possesso o meno del green pass da parte di un avventore. |
| Cameriere | Descrittore di un cameriere del ristorante. | **NumCid** (*string*): numero della carta di identità che serve anche da chiave primaria per identificare univocamente ciascun cameriere.  **Nome** (*string*): nome di un cameriere.  **Cognome** (*string*): cognome di un cameriere.  **DataN** (*date*): data di nascita di un cameriere.  **Sesso** (*TSesso*): sesso di un cameriere.  **CittaN** (*string*): città di nascita di un cameriere.  **ProvN** (*string*): provincia di nascita di un cameriere.  **CittaRes** (*string*): città di residenza di un cameriere.  **ProvRes** (*string*): provincia di residenza di un cameriere.  **Telefono** (*string*): numero di telefono di un cameriere.  **Email** (*string, opzionale*): email associata ad un cameriere. |
| Caso | Descrittore di un caso covid. | **CodCaso** (*integer*): chiave surrogata che serve per identificare univocamente ciascun caso.  **DataRegistrazione** (*date*): data di registrazione di un caso.  **StatoCaso** (*TStato*): stato di un caso (Risolto, InRisoluzione, NonRisolto).  **Note** (*string, opzionale*): note relative al caso. |

**Tabella 2.1**: Dizionario delle classi - fine.

## **2.4.2 Dizionario delle associazioni**

|  |  |  |
| --- | --- | --- |
| *Associazione* | *Descrizione* | *Classi coinvolte* |
| Amministrazione | Esprime l’amministrazione da parte di un proprietario di uno o più ristoranti | **Proprietario [1]** *ruolo* (**Amministrato**): indica il proprietario che amministra uno o più ristoranti.  **Ristorante [1..\*]** *ruolo* (**Amministra**): indica il ristorante che viene amministrato da un proprietario. |
| Gestione | Esprime la possibile gestione da parte di uno o più manager di un ristorante. | **ManagerRistorante [0..\*]** *ruolo* (**Gestito**):  indica il manager che può gestire un ristorante.  **Ristorante [0..1]** *ruolo* (**Gestisce**): indica il ristorante che può essere gestito da uno o più manager. |
| Appartenenza | Esprime l’appartenenza di una o più sale ad un ristorante. | **Ristorante [1]** *ruolo* (**Appartenuta**): indica il ristorante a cui appartengono una o più sale.  **Sala [1..\*]** *ruolo* (**Appartiene**): indica una sala che appartiene ad un ristorante. |
| Contenimento | Esprime il contenimento di tavoli da parte di una sala. | **Sala [1]** *ruolo* (**Contenuto**): indica una sala contenente uno o più tavoli.  **Tavolo [1..\*]** *ruolo* (**Contiene**): indica un tavolo contenuto in una sala. |
| Servizio | Esprime il servizio di un cameriere ad una tavolata. | **Cameriere [1]** *ruolo* (**Servita**): indica il cameriere che serve una tavolata  **Tavolata [1]** *ruolo* (**Serve**): indica la tavolata servita da un cameriere. |
| Partecipazione | Esprime uno o più avventori che partecipano ad una tavolata. | **Tavolata [1..\*]** *ruolo* (**Partecipa**): indica la tavolata a cui partecipa uno o più avventori.  **Avventore [1]** *ruolo* (**Formata**): indica uno o più avventori che partecipano ad una tavolata. |
| Composizione | Esprime la composizione di una tavolata con un tavolo. | **Tavolata [1]** *ruolo* (**Compone**): indica la tavolata composta da un tavolo.  **Tavolo [1]** *ruolo* (**Composta**): indica il tavolo che compone una tavolata. |
| AdiacenzaTavolo | Esprime la possibile adiacenza di altri tavoli ad un tavolo. | **Tavolo [1]** *ruolo* (**AdiacentiTavolo**): indica il tavolo a cui possono essere adiacenti altri tavoli.  **Tavolo [0..\*]** *ruolo* (**AdiacenteTavoli**): indica la possibilità di adiacenza di altri tavoli ad un tavolo. |
| Lavoro | Esprime il rapporto lavorativo tra uno o più camerieri ed un ristorante. | **Cameriere [1..\*]** *ruolo* (**Lavoratore**): indica il cameriere che lavora per un ristorante.  **Ristorante [1]** *ruolo* (**Lavora**): indica il ristorante per cui lavorano uno o più camerieri. |
| Accoglienza | Esprime l’accoglienza da parte di un ristorante di uno o più avventori. | **Ristorante [1]** *ruolo* (**Accolto**): indica il ristorante che accoglie uno o più avventori  **Avventore [1..\*]** *ruolo* (**Accoglie**): indica uno o più avventori accolti da un ristorante. |
| PositivitaAvventore | Esprime la possibilità di esistenza di un caso di positività per un avventore. | **Avventore [0..1]** *ruolo* (**CasoAvventore**): indica l’avventore che può risultare positivo.  **Caso [0..1]** *ruolo* (**AvventorePositivo**): indica il possibile caso di un avventore positivo. |
| PositivitaCameriere | Esprime la possibilità di esistenza di un caso di positività per un cameriere. | **Caso [0..1]** *ruolo* (**CamerierePositivo**): indica il cameriere che può risultare positivo.  **Cameriere [0..1]** *ruolo* (**CasoCameriere**): indica il possibile caso di un cameriere positivo. |
| TracciamentoProprietario | Esprime il tracciamento da parte di un proprietario di uno o più possibili casi. | **Proprietario [1]** *ruolo* (**Registrato**): indica il proprietario che registra uno o più possibili casi.  **Caso [0..\*]** *ruolo* (**Registra**): indica uno o più possibili casi registrati da un proprietario. |
| TracciamentoManager | Esprime il possibile tracciamento da parte di un manager di uno o più possibili casi. | **Manager [0..1]** *ruolo* (**Registrato**): indica il manager che può registrare uno o più possibili casi.  **Caso [0..\*]** *ruolo* (**Registra**): indica uno o più possibili casi registrati da un manager. |

**Tabella 2.2**: Dizionario delle associazioni - fine.

## **2.4.3 Dizionario dei vincoli**

|  |  |  |
| --- | --- | --- |
| *Vincolo* | *Tipo* | *Descrizione* |
| Has greenpass | Intrarelazionale | Se l’attributo HaGreenPass è FALSE allora un Avventore può essere associato solo ad una Tavolata che abbia un Tavolo la cui ubicazione è in una Sala di tipo “Esterna”. |
| Numero avventori a tavolata legale | Interrelazionale | Non deve esistere una tavolata il cui numero di avventori sia superiore all’attributo MaxAvventori del Tavolo cui è assegnata. |
| Email legale | Dominio | Per tutte le entità che possiedono l’attributo di tipo stringa “Email”. L’email deve rispettare la sua forma standard ovvero contenere almeno un carattere prima della @, almeno un carattere tra essa e il punto e almeno due caratteri nella parte finale. |
| Sito web legale | Dominio | Per l’attributo di tipo stringa “SitoWeb” dell’entità ristorante, deve rispettare la seguente forma standard ovvero contenere tre caratteri iniziali corrispondenti a “www” seguiti da un punto, almeno due caratteri, ed infine un punto seguito da almeno due caratteri nella parte finalela stringa non deve contenere più di 75 caratteri |
| Numero di telefono legale | Dominio | L’attributo Telefono presente nelle varie entità di tipo stringa deve contenere solo numeri e rispettare il formato standard italiano. |
| Data nascita legale | Interrelazionale | La DataN di un determinato avventore deve essere maggiore della DataArrivo della tavolata in cui esso è associato. |
| Data arrivo legale | Dominio | La DataArrivo di una Tavolata deve essere maggiore o uguale alla Data corrente. |
| CAP legale | Dominio | L'attributo “CAP” di tipo stringa deve contenere tutti numeri (e deve essere di 5 cifre) |
| CapienzaAvventori legale | Dominio | L’attributo CapienzaAvventori di Sala deve essere maggiore di 0. |
| DimensioneMq legale | Dominio | La dimensione in mq di una Sala deve essere maggiore di 0. |
| MaxAvventori legale | Dominio | L’attributo MaxAvventori di Tavolo deve essere > 0 e ≤ all’attributo CapienzaAvventori di Sala. |
| Somma MaxAvventori legale |  | La somma totale dell’attributo MaxAvventori per ogni istanza di Tavolo deve essere ≤ all’attributo CapienzaAvventori della Sala che contiene le istanze di Tavolo. |
| Temperatura avventore legale | Dominio | La temperatura registrata per un avventore deve essere maggiore o uguale a 35° e minore uguale a 37,5°. |
| Data di arrivo avventori a tavolata legale | Dominio | La DataArrivo di una Tavolata deve essere maggiore o uguale alla Data corrente. |
| Cameriere associato a tavolata legale | Dominio | Non deve esistere una tavolata in cui il numcid del cameriere che serve al tavolo che compone la tavolata sia diverso dal numcid associato al tavolo cui il cameriere è assegnato di default. |
| Password legale | Dominio | Una password deve contenere minimo 8 caratteri ed inoltre deve rispettare il seguente formato:  - deve esserci almeno una lettera;  - deve esserci almeno un numero. |
| Età cameriere legale | Dominio | L’età di un cameriere deve essere > di 18 anni |
| Unica tavolata associata a tavolo | Interelazionale | Per ogni DataArrivo di Tavolata, deve esistere un solo CodTavolata associato ad un dato CodTavolo. |
| Unico username proprietario | Interelazionale | L’username del proprietario deve essere diverso per ogni istanza. |
| Unico username manager | Interelazionale | L’username del manager deve essere diverso per ogni istanza. |
| TSesso | Dominio | Il campo Sesso può assumere solo i valori: Maschio, Femmina, Non specificato. |
| TSala | Dominio | Il campo Sala può assumere solo i valori: Interno, Esterno. |
| TCaso | Dominio | Il campo Caso può assumere solo i valori: Risolto, InRisoluzione, NonRisolto. |

**Tabella 2.3**: Dizionario dei vincoli.

**2.4.4 Dizionario delle interrogazioni**

|  |  |  |  |
| --- | --- | --- | --- |
| *Interrogazione* | *Descrizione* | *Frequenza* | *Classi coinvolte* |
| Numero giornaliero totale di avventori per ristorante |  | Giornaliera |  |
| Numero mensile totale di avventori per ristorante |  | Mensile? |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Tabella 2.4**: Dizionario delle interrogazioni.

# **Capitolo 3**

**Progettazione logica**

## **Introduzione**

In questo capitolo viene affrontata la fase successiva alla progettazione concettuale della base di dati, scendendo ad un livello di astrazione più basso e vicino all’implementazione vera e propria della base di dati. Si tradurrà lo schema concettuale ristrutturato in uno **schema logico**, dipendente dal **modello dei dati** scelto ossia quello **relazionale**. Negli schemi relazionali che seguiranno le **chiavi primarie** sono indicate con una *singola sottolineatura* mentre le **chiavi esterne** con una *doppia sottolineatura*.

3.1 Schema logico

## **3.1.1 Traduzione in schemi relazionali**

**Proprietario**(CodProprietario, Username, Password, Nome, Cognome, Email)

**Chiavi esterne**: nessuna.

**Ristorante**(CodRistorante, Denominazione, Indirizzo, Telefono, Citta, Prov, Cap, Email, SitoWeb, Proprietario)

**Chiavi esterne**: Proprietario → Proprietario.CodProprietario

**ManagerRistorante**(CodManager, Username, Password, Nome, Cognome, Email, Telefono, RistoranteGestito)

**Chiavi esterne**: RistoranteGestito → Ristorante.CodRistorante

**Sala**(CodSala, Denominazione, CapienzaAvventori, DimensioneMq, TipoSala, Ristorante)

**Chiavi esterne**: Ristorante → Ristorante.CodRistorante

**Cameriere**(NumCid, Nome, Cognome, DataN, Sesso, CittaN, ProvN, CittaRes, ProvRes, Telefono, Email, Ristorante)

**Chiavi esterne**: Ristorante → Ristorante.CodRistorante

**Tavolo**(CodTavolo, MaxAvventori, Sala, CameriereIncaricato, TavoloAdiacente)

**Chiavi esterne**: Sala → Sala.CodSala  
**………………:** CameriereIncaricato → Cameriere.NumCid  
**………………:** TavoloAdiacente → Tavolo.CodTavolo

**Tavolata**(CodTavolata, DataArrivo, OraArrivo, OraUscita, Tavolo, Cameriere)

**Chiavi esterne**: Tavolo → Tavolo.CodTavolo  
**………………:** Cameriere → Cameriere.NumCid

**Avventore**(NumCid, Nome, Cognome, DataN, Sesso, CittaN, ProvN, CittaRes, ProvRes, Telefono, Email, Temperatura, HaGreenpass, Ristorante, Tavolata)

**Chiavi esterne**: Ristorante → Ristorante.CodRistorante  
**………………:** Tavolata → Tavolata.CodTavolata

**Caso**(CodCaso, DataRegistrazione, StatoCaso, Note, AvventorePositivo, CamerierePositivo, RegistraProprietario, RegistraManager)

**Chiavi esterne**: AvventorePositivo → Avventore.NumCid  
**………………:** CamerierePositivo → Cameriere.NumCid  
**………………:** RegistraProprietario → Proprietario.CodProprietario  
**………………:** RegistraManager → ManagerRistorante.CodManager

## **3.1.2 Traduzione delle associazioni**

Nella seguente tabella vengono riportate le traduzioni delle associazioni individuate nello schema logico.

|  |  |
| --- | --- |
| *Associazione* | *Implementazione* |
| Amministrazione | Chiave esterna in **Ristorante** → **Proprietario** |
| Gestione | Chiave esterna in **ManagerRistorante** → **Ristorante** |
| Appartenenza | Chiave esterna in **Sala** → **Ristorante** |
| Contenimento | Chiave esterna in **Tavolo** → **Sala** |
| Assegnazione | Chiave esterna in **Tavolo** → **Cameriere** |
| Servizio | Chiave esterna in **Tavolata** → **Cameriere** |
| Partecipazione | Chiave esterna in **Avventore** → **Tavolata** |
| Composizione | Chiave esterna in **Tavolata** → **Tavolo** |
| AdiacenzaTavolo | Chiave esterna in **Tavolo** → **Tavolo** |
| Lavoro | Chiave esterna in **Cameriere** → **Ristorante** |
| Accoglienza | Chiave esterna in **Avventore** → **Ristorante** |
| PositivitaAvventore | Chiave esterna in **Caso** → **Avventore** |
| PositivitaCameriere | Chiave esterna in **Caso** → **Cameriere** |
| TracciamentoProprietario | Chiave esterna in **Caso** → **Proprietario** |
| TracciamentoManager | Chiave esterna in **Caso** → **ManagerRistorante** |

**Tabella 3.1**: Traduzione delle associazioni

## **3.1.3 Schema logico generale**

Alla fine del procedimento di traduzione, in conclusione, perveniamo al seguente schema logico:

|  |  |
| --- | --- |
| Proprietario | (CodProprietario, Username, Password, Nome, Cognome, Email) |
| Ristorante | (CodRistorante, Denominazione, Indirizzo, Telefono, Citta, Prov, Cap, Email, SitoWeb, Proprietario) |
| ManagerRistorante | (CodManager, Username, Password, Nome, Cognome, Email, Telefono, RistoranteGestito) |
| Sala | (CodSala, Denominazione, CapienzaAvventori, DimensioneMq, TipoSala, Ristorante) |
| Cameriere | (NumCid, Nome, Cognome, DataN, Sesso, CittaN, ProvN, CittaRes, ProvRes, Telefono, Email, Ristorante) |
| Tavolo | (CodTavolo, MaxAvventori, Sala, CameriereIncaricato, TavoloAdiacente) |
| Tavolata | (CodTavolata, DataArrivo, OraArrivo, OraUscita, Tavolo, Cameriere) |
| Avventore | (NumCid, Nome, Cognome, DataN, Sesso, CittaN, ProvN, CittaRes, ProvRes, Telefono, Email, Temperatura, HaGreenpass, Ristorante, Tavolata) |
| Caso | (CodCaso, DataRegistrazione, StatoCaso, Note, AvventorePositivo, CamerierePositivo, RegistraProprietario, RegistraManager) |

**Tabella 3.2**: Schema logico

# **Capitolo 4**

**Progettazione fisica**

## **Introduzione**

In questo capitolo viene trattata l’ultima fase relativa alla progettazione della base di dati, ossia la progettazione fisica. Dopo aver scelto il **DBMS** su cui implementare la base di dati, si passerà alla traduzione dello schema logico in uno **schema fisico** dei dati, attraverso la definizione delle tabelle, l’implementazione dei relativi vincoli di integrità, dei trigger e delle procedure individuate in **SQL**.

4.1 Note sull’implementazione

Il DBMS scelto per l’implementazione della base di dati è **Oracle Database** nella versione **19C**. Siccome nel DBMS Oracle non è implementato il tipo di dato **boolean**, esso è stato simulato con un carattere (CHAR) che può assumere solo i valori in {‘T’,‘F’} che rappresentano rispettivamente TRUE e FALSE.

4.2 Definizione delle tabelle

## **4.2.1 Definizione della tabella PROPRIETARIO**

**4.2.2 Definizione della tabella RISTORANTE**