

UNIVERSITA DEGLI STUDI DI NAPOLI FEDERICO II
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE



Progettazione e sviluppo di una Base di Dati Relazionale per la
gestione di un Sistema di Tracciamento Contatti per Ristoranti

Bruno De Vivo

Indice

Descrizione del Progetto	5
1.1 Analisi del Problema:.....	5
1.2 Realizzazione:	5
Progettazione Concettuale.....	6
2.1 Introduzione	6
2.2 Class Diagram	6
2.3 Ristrutturazione	6
2.4 Class Diagram Ristrutturato.....	7
2.5 Dizionario delle Classi	8
2.6 Dizionario delle Associazioni	9
2.7 Dizionario dei Vincoli.....	10
Progettazione Logica.....	11
3.1 Introduzione	11
3.2 Traduzione	11
Progettazione Fisica	12
4.1 Introduzione	12
4.2 Definizione delle Tabelle	13
4.2.1 Ristorante	13
4.2.2 Sala.....	14
4.2.3 Tavolo	15
4.2.4 TavoliAdiacenti.....	16
4.2.5 Tavolata.....	17
4.2.6 Cliente	18
4.2.7 Cameriere	19
4.2.8 Prenotazione.....	20
4.2.9 Servizio	20
4.3 Viste	21
4.3.1 Prenotazioni Ristorante	21
4.4 Vincoli – Triggers	21
4.4.1 Check_Adiacenza_Stesso_Tavolo	21
4.4.2 Check_Cameriere_Ristorante	21
4.4.3 Check_Date	22
4.4.4 Check_Max_Avventori.....	22
4.4.5 Check_Tavoli_Sale	22

4.5 Triggers Attributi Calcolati e Trigger semplici.....	23
4.5.1 Capienza (Sala) – Insert	23
4.5.2 Capienza (Sala) – Delete.....	23
4.5.3 Capienza (Ristorante) – Insert.....	24
4.5.4 Capienza (Ristorante) – Delete	24
4.5.5 Numero Camerieri – Insert	25
4.5.6 Numero Camerieri – Delete	25
4.5.7 Numero Tavoli – Insert	25
4.5.8 Numero Tavoli – Delete.....	26
4.5.9 Numero Clienti – Insert.....	26
4.5.10 Numero Clienti – Delete	26

Capitolo 1

Descrizione del Progetto

1.1 Analisi del Problema:

Ci si impone di costruire una Base di Dati utile per il tracciamento contatti per vari locali di ristorazione, nello specifico lo scopo è di poter rilevare: i clienti seduti ad un determinato tavolo; i clienti seduti ai tavoli adiacenti nella stessa data; i camerieri che hanno effettuato il servizio per quel tavolo. La necessità nasce dal poter agilmente tenere nota degli individui entrati in contatto per poter effettuare una segnalazione agli interessati e alle autorità sanitarie nel caso di una denuncia di caso di COVID-19.

1.2 Realizzazione:

Il sistema permette di contenere informazioni riguardo vari ristoranti, i quali sono caratterizzati dal nome del ristorante, un identificativo e attributi calcolati quali il numero di camerieri e la capienza; questi sono associati a tutte le rispettive sale che contengono, queste sono caratterizzate dal nome della sala, il codice del ristorante associato e attributi calcolati quali il numero di tavoli presenti all'interno della sala e la capienza di clienti; queste a loro volta sono associate a tutti i tavoli contenuti. Ogni tavolo è fornito di un codice identificativo, il nome della sala di cui fa parte e il numero massimo di avventori che possono sedersi al tavolo; il tavolo è associato ad una tavolata e a sé stesso attraverso un'associazione ricorsiva che contiene i codici tutti i tavoli che sono adiacenti ad altri tavoli e a quali. La tavolata è invece considerabile come la raccolta di tutti i clienti che siederanno a un determinato tavolo ad una determinata data, oltre a questi dati questa relazione contiene anche il numero di clienti seduti al tavolo come attributo calcolato; questa è associata alle relazioni di "Prenotazione" e di "Servizio" le quali sono classi di associazione che servono a rappresentare rispettivamente le prenotazioni che i clienti fanno per sedere a una tavolata e l'insieme di camerieri che serve la tavolata. Ogni cliente è identificato dal codice alfanumerico della carta d'identità, e ogni cameriere da un codice identificativo assegnatogli dal sistema. Quest'ultimo è poi associato al ristorante presso il quale presta servizio, in modo da impedire che nel sistema venga assegnato un cameriere a servire presso più tavolate nella stessa data e in ristoranti diversi.

Capitolo 2

Progettazione Concettuale

2.1 Introduzione

La progettazione concettuale si impone di produrre lo schema progettuale nel suo livello di astrazione maggiore, completamente slegata da una realizzazione fisica, e quindi dai mezzi attraverso la quale il progetto diventa concreto.

2.2 Class Diagram

Per questo progetto è stato scelto l'UML come linguaggio per rappresentare il progetto, in tutte le sue entità e gli attributi di ognuna. Attraverso questo prezioso strumento si sono potute individuare agilmente le finestre di miglioramento del progetto e la costruzione di vincoli per permettere il corretto funzionamento del sistema.

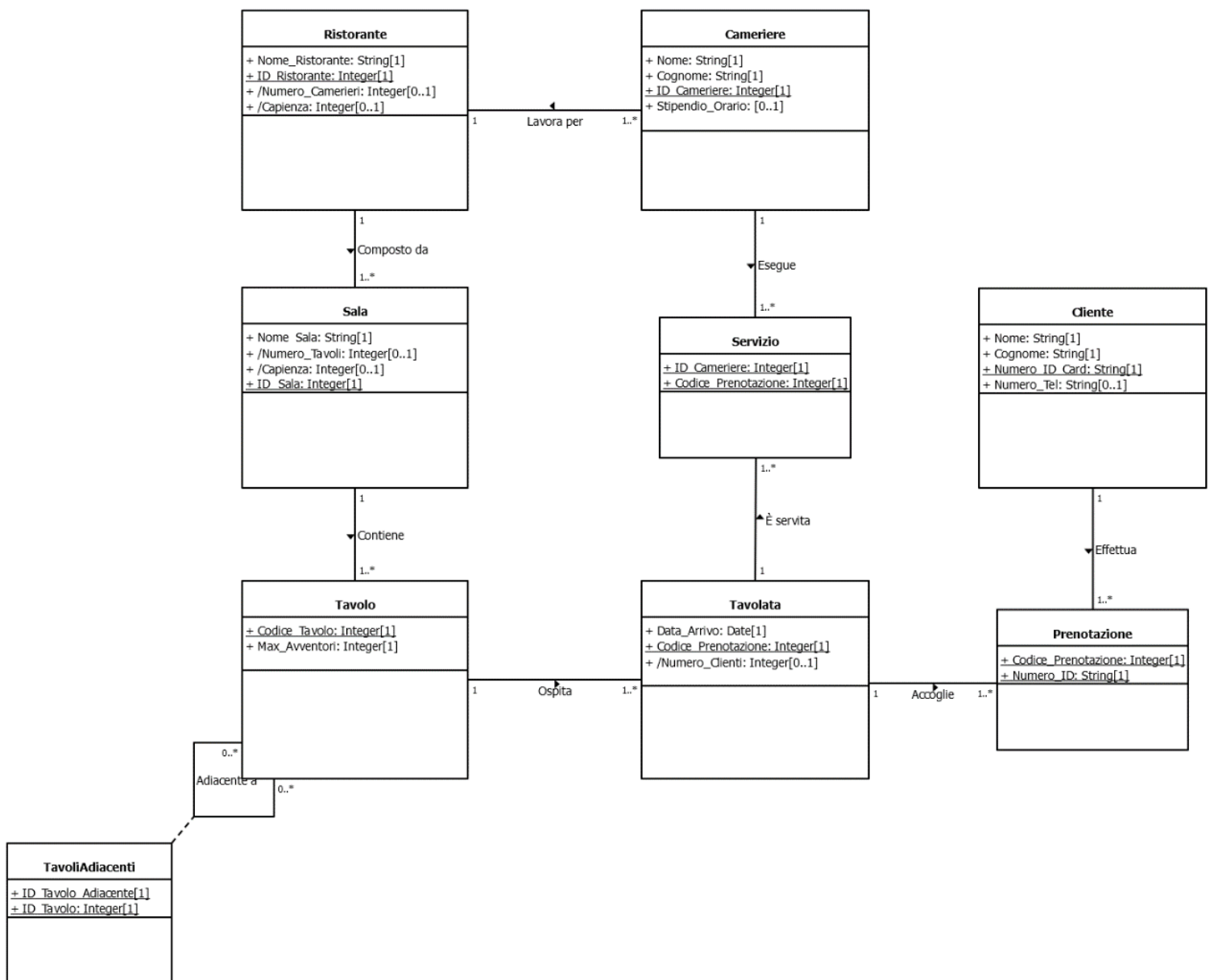
2.3 Ristrutturazione

La ristrutturazione del Class Diagram è un passaggio importante che va effettuato per alleggerire il carico del progetto il più possibile, tenendo comunque conto del target per il quale la base di dati sta venendo costruita.

Nel processo di ristrutturazione vengono rimosse le informazioni ridondanti, snelliti gli attributi multipli e le generalizzazioni. Solitamente i Class Diagram vengono utilizzati anche in contesti di Programmazione Orientata agli Oggetti, va quindi tenuto conto del fatto che benché sia possibile, in fase di ristrutturazione, il Class Diagram deve essere leggibile in un contesto relazionale.

Nel nostro caso la ristrutturazione del Class Diagram rispetto all'idea di partenza ha previsto di eliminare una generalizzazione delle entità "Cameriere" e "Cliente", le quali avevano una classe generalizzata "Persona", dalla quale ereditavano gli attributi di "Nome" e "Cognome", mantenuti poi nelle entità ristrutturate.

2.4 Class Diagram Ristrutturato



2.5 Dizionario delle Classi

<i>Classe</i>	<i>Descrizione</i>	<i>Attributi</i>
<i>Ristorante</i>	Descrittore di un ristorante	Nome_Ristorante (string) [1]: Nome del ristorante. Numero_Camerieri (integer) [0..1]: Attributo calcolato che esprime il numero di camerieri che prestano servizio al ristorante. Capienza (integer) [0..1]: Attributo calcolato che esprime il numero massimo di clienti che il ristorante può ospitare contemporaneamente. ID_Ristorante (integer) [1]: Chiave tecnica che identifica ogni elemento della classe <i>Ristorante</i> univocamente.
<i>Sala</i>	Descrittore di una sala di un ristorante	Nome_Sala (String) [1]: Nome della sala. Numero_Tavoli (Integer) [0..1]: Attributo calcolato che esprime il numero di tavoli presenti nella sala. Capienza (Integer) [0..1]: Attributo calcolato che esprime il numero massimo di clienti che la sala può ospitare contemporaneamente. ID_Sala (Integer) [1]: Chiave tecnica che identifica ogni elemento della classe <i>Sala</i> univocamente.
<i>Tavolo</i>	Descrittore di un tavolo in una sala	Max_Avventori (Integer) [1]: Attributo che esprime il numero massimo di clienti che possono sedersi ad un tavolo contemporaneamente. Codice_Tavolo (Integer) [1]: Chiave tecnica che identifica ogni elemento della classe <i>Tavolo</i> univocamente.
<i>Tavolata</i>	Descrittore di una tavolata che siede ad un tavolo	Data_Arrivo (Date) [1]: Attributo che esprime la data nella quale la tavolata ha preso posto nel ristorante. Numero_Clienti (Integer) [0..1]: Attributo calcolato che esprime il numero di clienti presenti alla tavolata. Codice_Prenotazione (Integer) [1]: Chiave tecnica che identifica ogni elemento della classe <i>Tavolata</i> univocamente.
<i>Cameriere</i>	Descrittore di un cameriere	Nome (String) [1]: Nome del cameriere. Cognome (String) [1]: Cognome del cameriere. ID_Cameriere (Integer) [1]: Chiave tecnica che identifica ogni elemento della classe <i>Cameriere</i> univocamente.
<i>Cliente</i>	Descrittore di un cliente	Nome (String) [1]: Nome del cliente. Cognome (String) [1]: Cognome del cliente. Numero_Tel (String) [0..1]: Numero di telefono del cliente. Numero_ID_Card (String) [1]: Codice identificativo della carta di identità del cliente.

2.6 Dizionario delle Associazioni

Associazione	Descrizione	Classi Coinvolte	Classi di Associazione
<i>Composto da</i>	Esprime il concetto secondo il quale una o più sale compongono un ristorante	Ristorante [1..*] ruolo (Composto da): Indica l'elemento di cui è composto il ristorante. Sala [1] ruolo (Compone): Indica l'elemento di cui Sala è componente.	Nessuna classe di associazione
<i>Contiene</i>	Esprime il concetto secondo il quale una sala contiene dei tavoli al suo interno	Sala [1..*] ruolo (Contiene): Indica l'elemento presente all'interno della sala. Tavolo [1] ruolo (Contenuto): Indica l'elemento in cui Tavolo è contenuto.	Nessuna classe di associazione
<i>Adiacente a</i>	Esprime il concetto di adiacenza tra tavoli	Tavolo [0..*] ruolo (Adiacente a): Indica l'elemento a cui Tavolo è adiacente. Tavolo [0..*] ruolo (Adiacente a): Indica l'elemento a cui Tavolo è adiacente.	TavoliAdiacenti: ID_Tavolo: Integer [1] ID_Tavolo_Adiacente: Integer [1]
<i>Ospita</i>	Esprime il concetto per cui un tavolo ospita una tavolata	Tavolo [0..*] ruolo (Ospita): Indica l'elemento che Tavolo ospita. Tavolata [1] ruolo (Ospitato da): Indica l'elemento che ospita la Tavolata.	
<i>Servizio</i>	Esprime il concetto per cui un cameriere serve una tavolata	Tavolata [1..*] ruolo (È servita da): Indica l'elemento da cui viene servita la tavolata. Cameriere [1..*] ruolo (Serve): Indica l'elemento che il cameriere serve.	Servizio: ID_Cameriere: Integer [1] Codice_Prenotazione: Integer [1]
<i>Prenotazione</i>	Esprime il concetto per cui un cliente si prenota per una tavolata	Tavolata [1..*] ruolo (Accoglie): Indica l'elemento che la tavolata accoglie. Cliente [1..*] ruolo (Si prenota per): Indica l'elemento per cui si prenota il cliente.	Prenotazione: Codice_Prenotazione: Integer [1] Numero_ID: String [1]
<i>Lavora per</i>	Esprime il concetto per cui un cameriere lavora in un ristorante	Cameriere [1] ruolo (Lavora per): Indica l'elemento per cui il cameriere lavora. Ristorante [1..*] ruolo (Fa lavorare): Indica l'elemento che il ristorante fa lavorare.	

2.7 Dizionario dei Vincoli

<i>Vincolo</i>	<i>Tipo</i>	<i>Descrizione</i>
<i>unique_nome_ristorante</i>	Intrarelazionale	Garantisce l'unicità del <i>nome</i> del <i>ristorante</i> .
<i>avventori_positivo</i>	Dominio	Impedisce l'inserimento di un numero minore di 1 come massimo di <i>avventori</i> a un tavolo.
<i>ck_only_numbers</i>	Dominio	Impedisce l'inserimento di valori che non siano numeri per l'attributo <i>Numero_Tel</i> di <i>Cliente</i> .
<i>ck_lunghezza_id</i>	Dominio	La lunghezza del codice della carta d'identità non può essere diversa dal valore stabilito.
<i>unique_numero_tel</i>	Intrarelazionale	Garantisce l'unicità del <i>numero di telefono</i> di un <i>cliente</i> .
<i>check_adiacenza_stesso_tavolo</i>	N-upla	Impedisce di inserire un <i>tavolo</i> come adiacente a sé stesso.
<i>check_cameriere_ristorante</i>	Interrelazionale	Impedisce di inserire un <i>cameriere</i> a servire presso un <i>ristorante</i> per il quale non lavora e a più tavoli in ristoranti diversi contemporaneamente.
<i>check_date</i>	Interrelazionale	Impedisce l'inserimento di una <i>tavolata</i> per una data in cui il <i>tavolo</i> scelto è già occupato.
<i>check_max_avventori</i>	Interrelazionale	Impedisce l'inserimento di un numero di <i>clienti</i> maggiore rispetto a quanti il <i>tavolo</i> scelto per la <i>tavolata</i> ne possa ospitare.
<i>check_tavoli_sale</i>	Interrelazionale	Impedisce che un tavolo venga inserito come adiacente a un tavolo appartenente ad una diversa sala
<i>/Numero_Camerieri</i>	Attributo Calcolato	Si ottiene contando il numero di tutti i <i>camerieri</i> che lavorano per un dato <i>ristorante</i> .
<i>/Capienza (Sala)</i>	Attributo Calcolato	Si ottiene sommando gli attributi <i>Max_Avventori</i> di tutti i <i>tavoli</i> presenti nella <i>sala</i> .
<i>/Capienza (Ristorante)</i>	Attributo Calcolato	Si ottiene sommando gli attributi <i>Capienza</i> di tutte le <i>sale</i> presenti nel <i>ristorante</i> .
<i>/Numero_Tavoli</i>	Attributo Calcolato	Si ottiene contando il numero di tutti i <i>tavoli</i> presenti nella <i>sala</i> .
<i>/Numero_Clienti</i>	Attributo Calcolato	Si ottiene contando il numero di tutti i <i>clienti</i> prenotati per la <i>tavolata</i>

Capitolo 3

Progettazione Logica

3.1 Introduzione

Dopo aver completato la progettazione concettuale si passa ad un livello di astrazione intermedio. Da qui si sceglie il modello di dati da adottare, che sarà quello relazionale, in virtù della conoscenza che si ha di quest'ultimo verranno quindi tradotte le idee ottenute in fase di progettazione concettuale in una forma più concreta e vicina a quella che sarà poi la progettazione fisica che tratteremo più avanti.

Verranno quindi espresse le entità secondo uno schema logico attinente al modello relazionale puro, e implementati gli elementi importanti per far sì che il modello relazionale possa avere senso.

Per esattezza, le chiavi primarie saranno sottolineate e le chiavi secondarie saranno in corsivo.

3.2 Traduzione

Ristorante:	(<u>ID_Ristorante</u> , Nome_Ristorante, Numero_Camerieri, Capienza)
Sala:	(<u>ID_Sala</u> , Nome_Sala, Numero_Tavoli, Capienza, <i>ID_Ristorante</i>) ID_Ristorante -> Ristorante.ID_Ristorante
Tavolo:	(<u>Codice_Tavolo</u> , Max_Avventori, <i>ID_Sala</i>) ID_Sala -> Sala.ID_Sala
TavoliAdiacenti:	(<i>ID_Tavolo</i> , <i>ID_Tavolo_Adiacente</i>) ID_Tavolo -> Tavolo.Codice_Tavolo ID_Tavolo_Adiacente -> Tavolo.Codice_Tavolo
Tavolata:	(<u>Codice_Prenotazione</u> , Data_Arrivo, Numero_Clienti, <i>Codice_Tavolo</i>) Codice_Tavolo -> Tavolo.Codice_Tavolo
Cliente:	(Nome, Cognome, <u>Numero_ID_Card</u> , Numero_Tel)
Cameriere:	(Nome, Cognome, Stipendio_Orario, <u>ID_Cameriere</u> , <i>ID_Ristorante</i>) ID_Ristorante -> Ristorante.ID_Ristorante
Servizio:	(<i>ID_Cameriere</i> , <i>Codice_Prenotazione</i>) ID_Cameriere -> Cameriere.ID_Cameriere Codice_Prenotazione -> Tavolata.Codice_Prenotazione
Prenotazione:	(<i>Codice_Prenotazione</i> , <u>Numero_ID</u>) Codice_Prenotazione -> Tavolata.Codice_Prenotazione Numero_ID -> Cliente.Numero_ID_Card

Capitolo 4

Progettazione Fisica

4.1 Introduzione

Per la realizzazione fisica del database è stato scelto PostgreSQL come DBMS. Le variazioni rispetto alle differenti progettazioni in quanto a nomenclature e simili sono assenti, in fase di revisione si è proceduto a uniformare tutti i nomi di attributi ed entità in modo da avere una presentazione coerente del progetto dall'inizio alla fine.

4.2 Definizione delle Tabelle

4.2.1 Ristorante

```
1 CREATE TABLE public."Ristorante" (  
2     "Nome_Ristorante" name NOT NULL,  
3     "ID_Ristorante" integer NOT NULL,  
4     "Numero_Camerieri" integer,  
5     "Capienza" integer  
6 );  
7  
8  
9 CREATE SEQUENCE public."Ristorante_ID_Ristorante_seq"  
10     AS integer  
11     START WITH 1  
12     INCREMENT BY 1  
13     NO MINVALUE  
14     NO MAXVALUE  
15     CACHE 1;  
16  
17  
18 ALTER TABLE ONLY public."Ristorante" ALTER COLUMN "ID_Ristorante"  
19 SET DEFAULT nextval('public."Ristorante_ID_Ristorante_seq" '::regclass);  
20  
21  
22 ALTER TABLE ONLY public."Ristorante"  
23     ADD CONSTRAINT "Ristorante_pkey" PRIMARY KEY ("ID_Ristorante");  
24  
25  
26 ALTER TABLE ONLY public."Ristorante"  
27     ADD CONSTRAINT unique_nome_ristorante UNIQUE ("Nome_Ristorante");  
28
```

4.2.2 Sala

```
1 CREATE TABLE public."Sala" (  
2     "ID_Ristorante" integer NOT NULL,  
3     "Numero_Tavoli" integer,  
4     "Capienza" integer,  
5     "Nome_Sala" name NOT NULL,  
6     "ID_Sala" integer NOT NULL  
7 );  
8  
9 CREATE SEQUENCE public."Sala_ID_Sala_seq"  
10     AS integer  
11     START WITH 1  
12     INCREMENT BY 1  
13     NO MINVALUE  
14     NO MAXVALUE  
15     CACHE 1;  
16  
17  
18 ALTER TABLE ONLY public."Sala" ALTER COLUMN "ID_Sala"  
19 SET DEFAULT nextval('public."Sala_ID_Sala_seq" '::regclass);  
20  
21  
22 ALTER TABLE ONLY public."Sala"  
23     ADD CONSTRAINT "Sala_pkey" PRIMARY KEY ("ID_Sala");  
24  
25  
26 ALTER TABLE ONLY public."Sala"  
27     ADD CONSTRAINT "ID_Ristorante" FOREIGN KEY ("ID_Ristorante")  
28     REFERENCES public."Ristorante"("ID_Ristorante");  
29
```

4.2.3 Tavolo

```
1 CREATE TABLE public."Tavolo" (  
2     "Codice_Tavolo" integer NOT NULL,  
3     "Max_Avventori" integer NOT NULL,  
4     "ID_Sala" integer NOT NULL,  
5     CONSTRAINT avventori_positivo CHECK (("Max_Avventori" > 0))  
6 );  
7  
8 CREATE SEQUENCE public."Tavolo_Codice_Tavolo_seq"  
9     AS integer  
10    START WITH 1  
11    INCREMENT BY 1  
12    NO MINVALUE  
13    NO MAXVALUE  
14    CACHE 1;  
15  
16 ALTER TABLE ONLY public."Tavolo" ALTER COLUMN "Codice_Tavolo"  
17 SET DEFAULT nextval('public."Tavolo_Codice_Tavolo_seq"':regclass);  
18  
19  
20 ALTER TABLE ONLY public."Tavolo"  
21     ADD CONSTRAINT "Tavolo_pkey" PRIMARY KEY ("Codice_Tavolo");  
22  
23 CREATE TRIGGER conta_posti_1sala_delete_tavolo AFTER DELETE ON public."Tavolo"  
24 FOR EACH ROW EXECUTE FUNCTION public.conta_posti_1sala_delete_tavolo();  
25  
26 CREATE TRIGGER conta_posti_1sala_insert_tavolo AFTER INSERT ON public."Tavolo"  
27 FOR EACH ROW EXECUTE FUNCTION public.conta_posti_1sala_insert_tavolo();  
28  
29 CREATE TRIGGER conta_posti_2ristorante_delete_tavolo AFTER DELETE ON public."Tavolo"  
30 FOR EACH ROW EXECUTE FUNCTION public.conta_posti_2ristorante_delete_tavolo();  
31  
32 CREATE TRIGGER conta_posti_2ristorante_insert_tavolo AFTER INSERT ON public."Tavolo"  
33 FOR EACH ROW EXECUTE FUNCTION public.conta_posti_2ristorante_insert_tavolo();  
34  
35 CREATE TRIGGER update_sala_delete_tavolo AFTER DELETE ON public."Tavolo"  
36 FOR EACH ROW EXECUTE FUNCTION public.update_sala_delete_tavolo();  
37  
38 CREATE TRIGGER update_sala_insert_tavolo AFTER INSERT ON public."Tavolo"  
39 FOR EACH ROW EXECUTE FUNCTION public.update_sala_insert_tavolo();  
40  
41 ALTER TABLE ONLY public."Tavolo"  
42     ADD CONSTRAINT "ID_Sala_FK" FOREIGN KEY ("ID_Sala")  
43     REFERENCES public."Sala"("ID_Sala") NOT VALID;  
44
```


4.2.4 TavoliAdiacenti

```
1 CREATE TABLE public."TavoliAdiacenti" (  
2     "ID_Tavolo" integer NOT NULL,  
3     "ID_Tavolo_Adiacente" integer NOT NULL  
4 );  
5  
6  
7 ALTER TABLE ONLY public."TavoliAdiacenti"  
8     ADD CONSTRAINT "TavoliAdiacenti_pkey"  
9     PRIMARY KEY ("ID_Tavolo", "ID_Tavolo_Adiacente");  
10  
11 CREATE TRIGGER check_adiacenti BEFORE INSERT ON public."TavoliAdiacenti"  
12 FOR EACH ROW EXECUTE FUNCTION public.check_tavoli_sale();  
13  
14 CREATE TRIGGER check_stesso_tavolo BEFORE INSERT ON public."TavoliAdiacenti"  
15 FOR EACH ROW EXECUTE FUNCTION public.check_adiacenza_stesso_tavolo();  
16  
17 CREATE TRIGGER commutazione AFTER INSERT ON public."TavoliAdiacenti"  
18 FOR EACH ROW EXECUTE FUNCTION public.commutazione_adiacenza();  
19  
20 ALTER TABLE ONLY public."TavoliAdiacenti"  
21     ADD CONSTRAINT "TavoloAD_FK" FOREIGN KEY ("ID_Tavolo_Adiacente")  
22     REFERENCES public."Tavolo"("Codice_Tavolo");  
23  
24 ALTER TABLE ONLY public."TavoliAdiacenti"  
25     ADD CONSTRAINT "Tavolo_FK" FOREIGN KEY ("ID_Tavolo")  
26     REFERENCES public."Tavolo"("Codice_Tavolo");
```


4.2.5 Tavolata

```
1 CREATE TABLE public."Tavolata" (  
2     "Data_Arrivo" date NOT NULL,  
3     "Codice_Prenotazione" integer NOT NULL,  
4     "Codice_Tavolo" integer NOT NULL,  
5     "Numero_Clienti" integer  
6 );  
7  
8  
9 CREATE SEQUENCE public."Tavolata_Codice_Prenotazione_seq"  
10     AS integer  
11     START WITH 1  
12     INCREMENT BY 1  
13     NO MINVALUE  
14     NO MAXVALUE  
15     CACHE 1;  
16  
17 ALTER TABLE ONLY public."Tavolata" ALTER COLUMN "Codice_Prenotazione"  
18 SET DEFAULT nextval('public."Tavolata_Codice_Prenotazione_seq" '::regclass);  
19  
20 ALTER TABLE ONLY public."Tavolata"  
21     ADD CONSTRAINT "Tavolata_pkey" PRIMARY KEY ("Codice_Prenotazione");  
22  
23 CREATE TRIGGER check_tavolo_libero BEFORE INSERT ON public."Tavolata"  
24 FOR EACH ROW EXECUTE FUNCTION public.check_date();  
25  
26 ALTER TABLE ONLY public."Tavolata"  
27     ADD CONSTRAINT "Codice_Tavolo" FOREIGN KEY ("Codice_Tavolo")  
28     REFERENCES public."Tavolo"("Codice_Tavolo");  
29
```

4.2.6 Cliente

```
1 CREATE TABLE public."Cliente" (  
2     "Nome" name NOT NULL,  
3     "Cognome" name NOT NULL,  
4     "Numero_ID_Card" character(9) NOT NULL,  
5     "Numero_Tel" text,  
6     CONSTRAINT ck_only_numbers CHECK (("Numero_Tel" !~~ '%[^0-9]%'::text))  
7 );  
8  
9 ALTER TABLE ONLY public."Cliente"  
10     ADD CONSTRAINT "Cliente_pkey"  
11     PRIMARY KEY ("Numero_ID_Card");  
12  
13 ALTER TABLE public."Cliente"  
14     ADD CONSTRAINT ck_lunghezza_id  
15     CHECK ((length("Numero_ID_Card") = 9)) NOT VALID;  
16  
17 ALTER TABLE ONLY public."Cliente"  
18     ADD CONSTRAINT unique_numero_tel  
19     UNIQUE ("Numero_Tel");  
20
```

4.2.7 Cameriere

```
1 CREATE TABLE public."Cameriere" (  
2     "ID_Cameriere" integer NOT NULL,  
3     "Nome" name NOT NULL,  
4     "Cognome" name NOT NULL,  
5     "ID_Ristorante" integer NOT NULL,  
6     "Stipendio_Orario" integer  
7 );  
8  
9  
10 CREATE SEQUENCE public."Cameriere_ID_Cameriere_seq"  
11     AS integer  
12     START WITH 1  
13     INCREMENT BY 1  
14     NO MINVALUE  
15     NO MAXVALUE  
16     CACHE 1;  
17  
18 ALTER TABLE ONLY public."Cameriere" ALTER COLUMN "ID_Cameriere"  
19 SET DEFAULT nextval('public."Cameriere_ID_Cameriere_seq" '::regclass);  
20  
21 ALTER TABLE ONLY public."Cameriere"  
22     ADD CONSTRAINT "Cameriere_pkey" PRIMARY KEY ("ID_Cameriere");  
23  
24 CREATE TRIGGER update_ristorante_delete_cameriere AFTER DELETE ON public."Cameriere"  
25 FOR EACH ROW EXECUTE FUNCTION public.update_ristorante_delete_cameriere();  
26  
27 CREATE TRIGGER update_ristorante_insert_cameriere AFTER INSERT ON public."Cameriere"  
28 FOR EACH ROW EXECUTE FUNCTION public.update_ristorante_insert_cameriere();  
29  
30 ALTER TABLE ONLY public."Cameriere"  
31     ADD CONSTRAINT "Cameriere_ID_Ristorante_FK" FOREIGN KEY ("ID_Ristorante")  
32     REFERENCES public."Ristorante"("ID_Ristorante") NOT VALID;  
33
```

4.2.8 Prenotazione

```
1 CREATE TABLE public."Prenotazione" (  
2     "Codice_Prenotazione" integer NOT NULL,  
3     "Numero_ID" character(9) NOT NULL  
4 );  
5  
6 CREATE TRIGGER check_avventori BEFORE INSERT ON public."Prenotazione"  
7 FOR EACH ROW EXECUTE FUNCTION public.check_max_avventori();  
8  
9 CREATE TRIGGER update_tavolata_delete_clienteprenotazione  
10 AFTER DELETE ON public."Prenotazione"  
11 FOR EACH ROW EXECUTE FUNCTION public.update_tavolata_delete_clienteprenotazione();  
12  
13 CREATE TRIGGER update_tavolata_insert_clienteprenotazione AFTER INSERT ON public."Prenotazione"  
14 FOR EACH ROW EXECUTE FUNCTION public.update_tavolata_insert_clienteprenotazione();  
15  
16 ALTER TABLE ONLY public."Prenotazione"  
17     ADD CONSTRAINT "Codice_Prenotazione_FK" FOREIGN KEY ("Codice_Prenotazione")  
18     REFERENCES public."Tavolata"("Codice_Prenotazione");  
19  
20 ALTER TABLE ONLY public."Prenotazione"  
21     ADD CONSTRAINT "Numero_ID_FK" FOREIGN KEY ("Numero_ID")  
22     REFERENCES public."Cliente"("Numero_ID_Card");  
23
```

4.2.9 Servizio

```
1 CREATE TABLE public."Servizio" (  
2     "ID_Cameriere" integer NOT NULL,  
3     "Codice_Prenotazione" integer NOT NULL  
4 );  
5  
6  
7 ALTER TABLE ONLY public."Servizio"  
8     ADD CONSTRAINT "Servizio_pkey" PRIMARY KEY ("ID_Cameriere", "Codice_Prenotazione");  
9  
10 CREATE TRIGGER check_servizio BEFORE INSERT ON public."Servizio"  
11 FOR EACH ROW EXECUTE FUNCTION public.check_cameriere_ristorante();  
12  
13 ALTER TABLE ONLY public."Servizio"  
14     ADD CONSTRAINT "Cameriere_FK" FOREIGN KEY ("ID_Cameriere")  
15     REFERENCES public."Cameriere"("ID_Cameriere") NOT VALID;  
16  
17 ALTER TABLE ONLY public."Servizio"  
18     ADD CONSTRAINT "Prenotazione_FK" FOREIGN KEY ("Codice_Prenotazione")  
19     REFERENCES public."Tavolata"("Codice_Prenotazione") NOT VALID;  
20
```


4.3 Viste

4.3.1 Prenotazioni Ristorante

```
1 SELECT "Ristorante"."ID_Ristorante",
2     "Tavolata"."Codice_Prenotazione"
3 FROM "Ristorante"
4     JOIN "Sala" ON "Ristorante"."ID_Ristorante" = "Sala"."ID_Ristorante"
5     JOIN "Tavolo" ON "Sala"."ID_Sala" = "Tavolo"."ID_Sala"
6     JOIN "Tavolata" ON "Tavolo"."Codice_Tavolo" = "Tavolata"."Codice_Tavolo";
```

4.4 Vincoli – Triggers

4.4.1 Check_Adiacenza_Stesso_Tavolo

```
1 BEGIN
2 IF NEW."ID_Tavolo" = NEW."ID_Tavolo_Adiacente" THEN
3 RAISE EXCEPTION 'Un tavolo non può essere adiacente a sè stesso.';
4 END IF;
5
6 RETURN NEW;
7 END
```

4.4.2 Check_Cameriere_Ristorante

```
1 BEGIN
2
3 IF
4 (SELECT "Cameriere"."ID_Ristorante" FROM "Cameriere"
5 WHERE "Cameriere"."ID_Cameriere" = NEW."ID_Cameriere") <> (SELECT "Ristorante"."ID_Ristorante" FROM "Ristorante"
6 INNER JOIN "Sala" ON "Ristorante"."ID_Ristorante" = "Sala"."ID_Ristorante"
7 INNER JOIN "Tavolo" ON "Sala"."ID_Sala" = "Tavolo"."ID_Sala"
8 INNER JOIN "Tavolata" ON "Tavolo"."Codice_Tavolo" = "Tavolata"."Codice_Tavolo"
9 WHERE "Tavolata"."Codice_Prenotazione" = NEW."Codice_Prenotazione")
10 THEN
11 RAISE EXCEPTION 'Il Cameriere non può gestire la tavolata perché registrato a un altro Ristorante.';
12 END IF;
13 RETURN NEW;
14 END
```

4.4.3 Check_Date

```
1 DECLARE tavolo_trovato integer;
2 BEGIN
3 SELECT "Tavolata"."Codice_Tavolo" INTO tavolo_trovato FROM "Tavolata"
4 WHERE NEW."Data_Arrivo" = "Data_Arrivo"
5 AND
6 NEW."Codice_Tavolo" = "Codice_Tavolo";
7
8 IF FOUND THEN
9 RAISE EXCEPTION 'Tavolo occupato per la data inserita.';
10 END IF;
11 RETURN NEW;
12 END
```

4.4.4 Check_Max_Avventori

```
1 DECLARE max_avventori integer;
2 BEGIN
3 SELECT "Tavolo"."Max_Avventori" INTO max_avventori FROM "Tavolo"
4 INNER JOIN "Tavolata" ON "Tavolo"."Codice_Tavolo" = "Tavolata"."Codice_Tavolo"
5 INNER JOIN "Prenotazione" ON "Tavolata"."Codice_Prenotazione" = "Prenotazione"."Codice_Prenotazione";
6
7 IF
8 (SELECT COUNT(*) FROM "Prenotazione"
9 INNER JOIN "Tavolata" ON "Prenotazione"."Codice_Prenotazione" = "Tavolata"."Codice_Prenotazione"
10 INNER JOIN "Tavolo" ON "Tavolata"."Codice_Tavolo" = "Tavolo"."Codice_Tavolo"
11 WHERE "Prenotazione"."Codice_Prenotazione" = NEW."Codice_Prenotazione") = max_avventori
12 THEN
13 RAISE EXCEPTION 'Numero massimo di avventori raggiunto.';
14 END IF;
15 RETURN NEW;
16 END
```

4.4.5 Check_Tavoli_Sale

```
1 DECLARE sala1 text;
2 DECLARE sala2 text;
3 BEGIN
4
5 SELECT "Tavolo"."ID_Sala" INTO sala1 FROM "Tavolo"
6 WHERE "Tavolo"."Codice_Tavolo" = NEW."ID_Tavolo";
7
8 SELECT "Tavolo"."ID_Sala" INTO sala2 FROM "Tavolo"
9 WHERE "Tavolo"."Codice_Tavolo" = NEW."ID_Tavolo_Adiacente";
10
11 IF sala1 <> sala2 THEN
12 RAISE EXCEPTION 'I tavoli non possono essere adiacenti: non sono nella stessa sala';
13 END IF;
14 RETURN NEW;
15 END
```

4.5 Triggers Attributi Calcolati e Trigger semplici

4.5.1 Capienza (Sala) – Insert

```
1 DECLARE x integer;
2 BEGIN
3
4 SELECT SUM("Max_Avventori") INTO x FROM "Tavolo"
5 WHERE "Tavolo"."ID_Sala" = NEW."ID_Sala";
6
7 UPDATE "Sala" SET "Capienza" = x
8 WHERE "Sala"."ID_Sala" = NEW."ID_Sala";
9 RETURN NEW;
10 END
```

4.5.2 Capienza (Sala) – Delete

```
1 DECLARE x integer;
2 BEGIN
3
4 SELECT SUM("Max_Avventori") INTO x FROM "Tavolo"
5 WHERE "Tavolo"."ID_Sala" = OLD."ID_Sala";
6
7 UPDATE "Sala" SET "Capienza" = x
8 WHERE "Sala"."ID_Sala" = OLD."ID_Sala";
9 RETURN NEW;
10 END
```

4.5.3 Capienza (Ristorante) – Insert

```
1 DECLARE x integer;
2 DECLARE idRist integer;
3 BEGIN
4
5 SELECT "Ristorante"."ID_Ristorante" INTO idRist FROM "Ristorante"
6 INNER JOIN "Sala" ON "Ristorante"."ID_Ristorante" = "Sala"."ID_Ristorante"
7 INNER JOIN "Tavolo" ON "Sala"."ID_Sala" = "Tavolo"."ID_Sala"
8 WHERE "Tavolo"."Codice_Tavolo" = NEW."Codice_Tavolo";
9
10 SELECT SUM("Capienza") INTO x FROM "Sala"
11 WHERE "Sala"."ID_Ristorante" = idRist;
12
13 UPDATE "Ristorante" SET "Capienza" = x
14 WHERE "Ristorante"."ID_Ristorante" = idRist;
15
16 RETURN NEW;
17 END
```

4.5.4 Capienza (Ristorante) – Delete

```
1 DECLARE x integer;
2 DECLARE idRist integer;
3 BEGIN
4
5 SELECT "Ristorante"."ID_Ristorante" INTO idRist FROM "Ristorante"
6 INNER JOIN "Sala" ON "Ristorante"."ID_Ristorante" = "Sala"."ID_Ristorante"
7 INNER JOIN "Tavolo" ON "Sala"."ID_Sala" = "Tavolo"."ID_Sala"
8 WHERE "Tavolo"."Codice_Tavolo" = OLD."Codice_Tavolo";
9
10 SELECT SUM("Capienza") INTO x FROM "Sala"
11 WHERE "Sala"."ID_Ristorante" = idRist;
12
13 UPDATE "Ristorante" SET "Capienza" = x
14 WHERE "Ristorante"."ID_Ristorante" = idRist;
15
16 RETURN NEW;
17 END
```


4.5.5 Numero Camerieri – Insert

```
1 DECLARE x integer;
2 BEGIN
3
4 SELECT COUNT(*) INTO x FROM "Cameriere"
5 WHERE "Cameriere"."ID_Ristorante" = NEW."ID_Ristorante";
6
7 UPDATE "Ristorante" SET "Numero_Camerieri" = x
8 WHERE "Ristorante"."ID_Ristorante" = NEW."ID_Ristorante";
9 RETURN NEW;
10 END
```

4.5.6 Numero Camerieri – Delete

```
1 DECLARE x integer;
2 BEGIN
3
4 SELECT COUNT(*) INTO x FROM "Cameriere"
5 WHERE "Cameriere"."ID_Ristorante" = OLD."ID_Ristorante";
6
7 UPDATE "Ristorante" SET "Numero_Camerieri" = x
8 WHERE "Ristorante"."ID_Ristorante" = OLD."ID_Ristorante";
9 RETURN NEW;
10 END
```

4.5.7 Numero Tavoli – Insert

```
1 DECLARE x integer;
2 BEGIN
3 SELECT COUNT(*) INTO x FROM "Tavolo"
4 WHERE "Tavolo"."ID_Sala" = NEW."ID_Sala";
5
6 UPDATE "Sala" SET "Numero_Tavoli" = x
7 WHERE "Sala"."ID_Sala" = NEW."ID_Sala";
8 RETURN NEW;
9 END
```

4.5.8 Numero Tavoli – Delete

```
1 DECLARE x integer;
2 BEGIN
3 SELECT COUNT(*) INTO x FROM "Tavolo"
4 WHERE "Tavolo"."ID_Sala" = OLD."ID_Sala";
5
6 UPDATE "Sala" SET "Numero_Tavoli" = x
7 WHERE "Sala"."ID_Sala" = OLD."ID_Sala";
8 RETURN NEW;
9 END
```

4.5.9 Numero Clienti – Insert

```
1 DECLARE x integer;
2 BEGIN
3
4 SELECT COUNT(*) INTO x FROM "Prenotazione"
5 WHERE "Prenotazione"."Codice_Prenotazione" = NEW."Codice_Prenotazione";
6
7 UPDATE "Tavolata" SET "Numero_Clienti" = x
8 WHERE "Tavolata"."Codice_Prenotazione" = NEW."Codice_Prenotazione";
9 RETURN NEW;
10 END
```

4.5.10 Numero Clienti – Delete

```
1 DECLARE x integer;
2 BEGIN
3
4 SELECT COUNT(*) INTO x FROM "Prenotazione"
5 WHERE "Prenotazione"."Codice_Prenotazione" = OLD."Codice_Prenotazione";
6
7 UPDATE "Tavolata" SET "Numero_Clienti" = x
8 WHERE "Tavolata"."Codice_Prenotazione" = OLD."Codice_Prenotazione";
9 RETURN NEW;
10 END
```