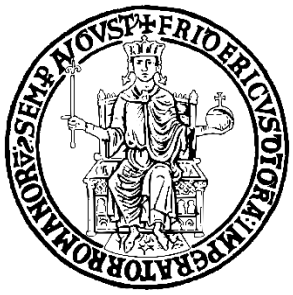
UNIVERSITA DEGLI STUDI DI NAPOLI FEDERICO II
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE



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

Bruno De Vivo
N86003676

Antonio Giordano
N86003842

Luglio 2022

Indice

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.

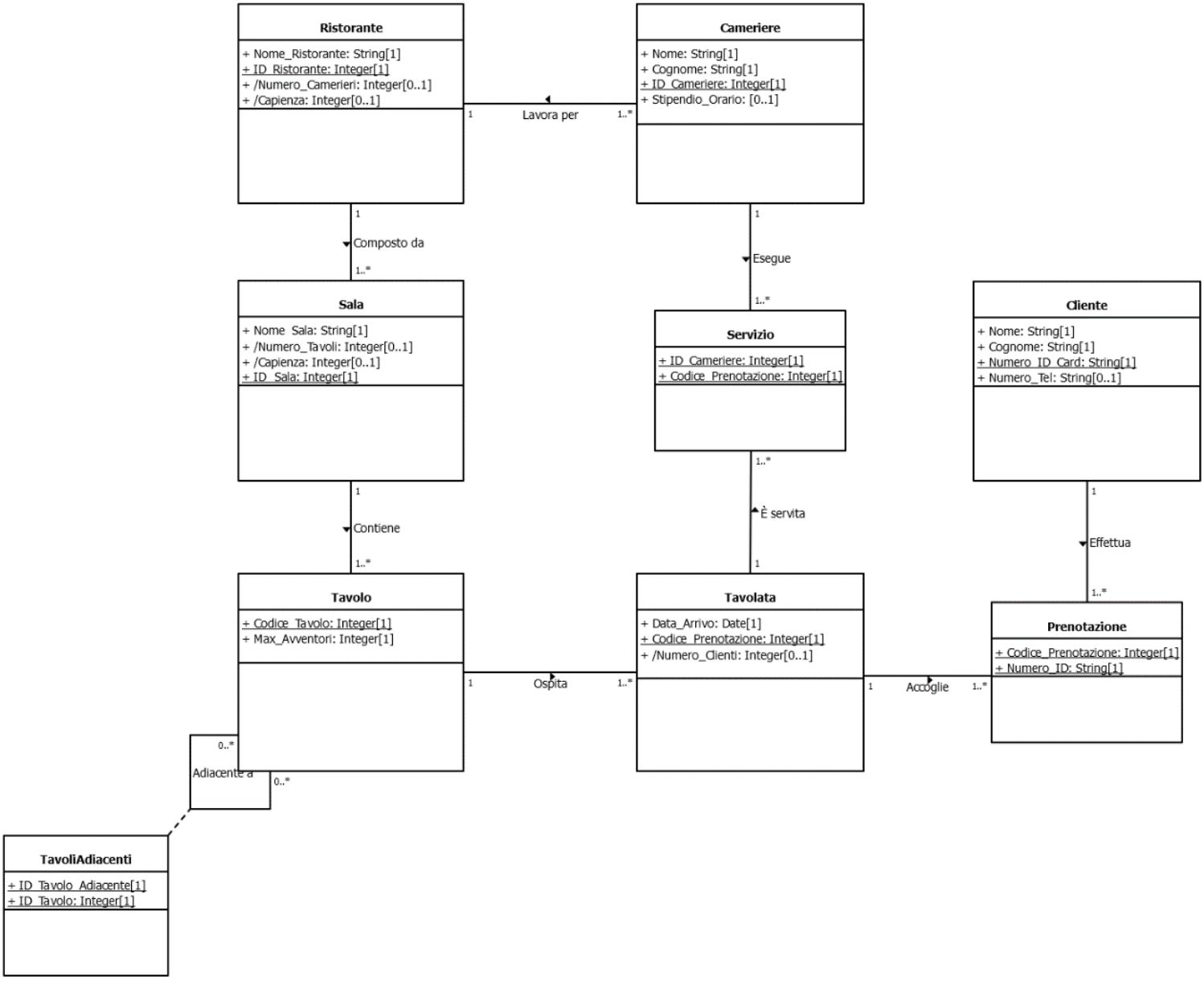
2.3 Ristrutturazione

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

Classe	*Descrizione*	*Attributi*
Ristorante	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 *Ristorante* univocamente.
Sala	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 contemporeaneamente. **ID_Sala** (Integer) [1]: Chiave tecnica che identifica ogni elemento della classe *Sala* univocamente.
Tavolo	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 *Tavolo* univocamente.
Tavolata	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 *Tavolata* univocamente.
Cameriere	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 *Cameriere* univocamente.
Cliente	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*
Composto da	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*
Contiene	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*
Adiacente a	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]
Ospita	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.	
Servizio	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]
Prenotazione	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]
Lavora per	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

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

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: (ID_Ristorante, Nome_Ristorante, Numero_Camerieri, Capienza)

Sala: (ID_Sala, Nome_Sala, Numero_Tavoli, Capienza, *ID_Ristorante*)
ID_Ristorante -> Ristorante.ID_Ristorante

Tavolo: (Codice_Tavolo, Max_Avventori, *ID_Sala*)
ID_Sala -> Sala.ID_Sala

TavoliAdiacenti: (*ID_Tavolo*, *ID_Tavolo_Adiacente*)
ID_Tavolo -> Tavolo.Codice_Tavolo
ID_Tavolo_Adiacente -> Tavolo.Codice_Tavolo

Tavolata: (Codice_Prenotazione, Data_Arrivo, Numero_Clienti, *Codice_Tavolo*)
Codice_Tavolo -> Tavolo.Codice_Tavolo

Cliente: (Nome, Cognome, Numero_ID_Card, Numero_Tel)

Cameriere: (Nome, Cognome, Stipendio_Orario, ID_Cameriere, *ID_Ristorante*)
ID_Ristorante -> Ristorante.ID_Ristorante

Servizio: (*ID_Cameriere*, *Codice_Prenotazione*)
ID_Cameriere -> Cameriere.ID_Cameriere
Codice_Prenotazione -> Tavolata.Codice_Prenotazione

Prenotazione: (*Codice_Prenotazione*, *Numero_ID*)
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

```
CREATE TABLE public."Ristorante" (
    "Nome_Ristorante" name NOT NULL,
    "ID_Ristorante" integer NOT NULL,
    "Numero_Camerieri" integer,
    "Capienza" integer
);


CREATE SEQUENCE public."Ristorante_ID_Ristorante_seq"
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE ONLY public."Ristorante" ALTER COLUMN "ID_Ristorante"
SET DEFAULT nextval('public."Ristorante_ID_Ristorante_seq"'::regclass);


ALTER TABLE ONLY public."Ristorante"
    ADD CONSTRAINT "Ristorante_pkey" PRIMARY KEY ("ID_Ristorante");


ALTER TABLE ONLY public."Ristorante"
    ADD CONSTRAINT unique_nome_ristorante UNIQUE ("Nome_Ristorante");

```

4.2.2 Sala

```
CREATE TABLE public."Sala" (
    "ID_Ristorante" integer NOT NULL,
    "Numero_Tavoli" integer,
    "Capienza" integer,
    "Nome_Sala" name NOT NULL,
    "ID_Sala" integer NOT NULL
);

CREATE SEQUENCE public."Sala_ID_Sala_seq"
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE ONLY public."Sala" ALTER COLUMN "ID_Sala"
SET DEFAULT nextval('public."Sala_ID_Sala_seq"'::regclass);


ALTER TABLE ONLY public."Sala"
    ADD CONSTRAINT "Sala_pkey" PRIMARY KEY ("ID_Sala");


ALTER TABLE ONLY public."Sala"
    ADD CONSTRAINT "ID_Ristorante" FOREIGN KEY ("ID_Ristorante")
    REFERENCES public."Ristorante"("ID_Ristorante");

```

4.2.3 Tavolo

```
CREATE TABLE public."Tavolo" (
    "Codice_Tavolo" integer NOT NULL,
    "Max_Avventori" integer NOT NULL,
    "ID_Sala" integer NOT NULL,
    CONSTRAINT avventori_positivo CHECK (("Max_Avventori" > 0))
);

CREATE SEQUENCE public."Tavolo_Codice_Tavolo_seq"
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE ONLY public."Tavolo" ALTER COLUMN "Codice_Tavolo"
SET DEFAULT nextval('public."Tavolo_Codice_Tavolo_seq"'::regclass);


ALTER TABLE ONLY public."Tavolo"
    ADD CONSTRAINT "Tavolo_pkey" PRIMARY KEY ("Codice_Tavolo");

CREATE TRIGGER conta_posti_1sala_delete_tavolo AFTER DELETE ON public."Tavolo"
FOR EACH ROW EXECUTE FUNCTION public.conta_posti_1sala_delete_tavolo();

CREATE TRIGGER conta_posti_1sala_insert_tavolo AFTER INSERT ON public."Tavolo"
FOR EACH ROW EXECUTE FUNCTION public.conta_posti_1sala_insert_tavolo();

CREATE TRIGGER conta_posti_2ristorante_delete_tavolo AFTER DELETE ON public."Tavolo"
FOR EACH ROW EXECUTE FUNCTION public.conta_posti_2ristorante_delete_tavolo();

CREATE TRIGGER conta_posti_2ristorante_insert_tavolo AFTER INSERT ON public."Tavolo"
FOR EACH ROW EXECUTE FUNCTION public.conta_posti_2ristorante_insert_tavolo();

CREATE TRIGGER update_sala_delete_tavolo AFTER DELETE ON public."Tavolo"
FOR EACH ROW EXECUTE FUNCTION public.update_sala_delete_tavolo();

CREATE TRIGGER update_sala_insert_tavolo AFTER INSERT ON public."Tavolo"
FOR EACH ROW EXECUTE FUNCTION public.update_sala_insert_tavolo();

ALTER TABLE ONLY public."Tavolo"
    ADD CONSTRAINT "ID_Sala_FK" FOREIGN KEY ("ID_Sala")
    REFERENCES public."Sala"("ID_Sala") NOT VALID;

```

4.2.4 TavoliAdiacenti

```
CREATE TABLE public."TavoliAdiacenti" (
    "ID_Tavolo" integer NOT NULL,
    "ID_Tavolo_Adiacente" integer NOT NULL
);


ALTER TABLE ONLY public."TavoliAdiacenti"
    ADD CONSTRAINT "TavoliAdiacenti_pkey"
    PRIMARY KEY ("ID_Tavolo", "ID_Tavolo_Adiacente");

CREATE TRIGGER check_adiacenti BEFORE INSERT ON public."TavoliAdiacenti"
FOR EACH ROW EXECUTE FUNCTION public.check_tavoli_sale();

CREATE TRIGGER check_stesso_tavolo BEFORE INSERT ON public."TavoliAdiacenti"
FOR EACH ROW EXECUTE FUNCTION public.check_adiacenza_stesso_tavolo();

CREATE TRIGGER commutazione AFTER INSERT ON public."TavoliAdiacenti"
FOR EACH ROW EXECUTE FUNCTION public.commutazione_adiacenza();

ALTER TABLE ONLY public."TavoliAdiacenti"
    ADD CONSTRAINT "TavoloAD_FK" FOREIGN KEY ("ID_Tavolo_Adiacente")
    REFERENCES public."Tavolo"("Codice_Tavolo");

ALTER TABLE ONLY public."TavoliAdiacenti"
    ADD CONSTRAINT "Tavolo_FK" FOREIGN KEY ("ID_Tavolo")
    REFERENCES public."Tavolo"("Codice_Tavolo");
```

4.2.5 Tavolata

```
CREATE TABLE public."Tavolata" (
    "Data_Arrivo" date NOT NULL,
    "Codice_Prenotazione" integer NOT NULL,
    "Codice_Tavolo" integer NOT NULL,
    "Numero_Clienti" integer
);


CREATE SEQUENCE public."Tavolata_Codice_Prenotazione_seq"
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE ONLY public."Tavolata" ALTER COLUMN "Codice_Prenotazione"
SET DEFAULT nextval('public."Tavolata_Codice_Prenotazione_seq"'::regclass);

ALTER TABLE ONLY public."Tavolata"
    ADD CONSTRAINT "Tavolata_pkey" PRIMARY KEY ("Codice_Prenotazione");

CREATE TRIGGER check_tavolo_libero BEFORE INSERT ON public."Tavolata"
FOR EACH ROW EXECUTE FUNCTION public.check_date();

ALTER TABLE ONLY public."Tavolata"
    ADD CONSTRAINT "Codice_Tavolo" FOREIGN KEY ("Codice_Tavolo")
    REFERENCES public."Tavolo"("Codice_Tavolo");

```

4.2.6 Cliente

```
CREATE TABLE public."Cliente" (
    "Nome" name NOT NULL,
    "Cognome" name NOT NULL,
    "Numero_ID_Card" character(9) NOT NULL,
    "Numero_Tel" text,
    CONSTRAINT ck_only_numbers CHECK (("Numero_Tel" !~~ '%[^0-9]%'::text))
);

ALTER TABLE ONLY public."Cliente"
    ADD CONSTRAINT "Cliente_pkey"
    PRIMARY KEY ("Numero_ID_Card");

ALTER TABLE public."Cliente"
    ADD CONSTRAINT ck_lunghezza_id
    CHECK ((length("Numero_ID_Card") = 9)) NOT VALID;

ALTER TABLE ONLY public."Cliente"
    ADD CONSTRAINT unique_numero_tel
    UNIQUE ("Numero_Tel");

```

4.2.7 Cameriere

```
CREATE TABLE public."Cameriere" (
    "ID_Cameriere" integer NOT NULL,
    "Nome" name NOT NULL,
    "Cognome" name NOT NULL,
    "ID_Ristorante" integer NOT NULL,
    "Stipendio_Orario" integer
);


CREATE SEQUENCE public."Cameriere_ID_Cameriere_seq"
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE ONLY public."Cameriere" ALTER COLUMN "ID_Cameriere"
SET DEFAULT nextval('public."Cameriere_ID_Cameriere_seq"'::regclass);

ALTER TABLE ONLY public."Cameriere"
    ADD CONSTRAINT "Cameriere_pkey" PRIMARY KEY ("ID_Cameriere");

CREATE TRIGGER update_ristorante_delete_cameriere AFTER DELETE ON public."Cameriere"
FOR EACH ROW EXECUTE FUNCTION public.update_ristorante_delete_cameriere();

CREATE TRIGGER update_ristorante_insert_cameriere AFTER INSERT ON public."Cameriere"
FOR EACH ROW EXECUTE FUNCTION public.update_ristorante_insert_cameriere();

ALTER TABLE ONLY public."Cameriere"
    ADD CONSTRAINT "Cameriere_ID_Ristorante_FK" FOREIGN KEY ("ID_Ristorante")
    REFERENCES public."Ristorante"("ID_Ristorante") NOT VALID;

```

4.2.8 Prenotazione

```
CREATE TABLE public."Prenotazione" (
    "Codice_Prenotazione" integer NOT NULL,
    "Numero_ID" character(9) NOT NULL
);

CREATE TRIGGER check_avventori BEFORE INSERT ON public."Prenotazione"
FOR EACH ROW EXECUTE FUNCTION public.check_max_avventori();

CREATE TRIGGER update_tavolata_delete_clienteprenotazione
AFTER DELETE ON public."Prenotazione"
FOR EACH ROW EXECUTE FUNCTION public.update_tavolata_delete_clienteprenotazione();

CREATE TRIGGER update_tavolata_insert_clienteprenotazione AFTER INSERT ON public."Prenotazione"
FOR EACH ROW EXECUTE FUNCTION public.update_tavolata_insert_clienteprenotazione();

ALTER TABLE ONLY public."Prenotazione"
    ADD CONSTRAINT "Codice_Prenotazione_FK" FOREIGN KEY ("Codice_Prenotazione")
    REFERENCES public."Tavolata"("Codice_Prenotazione");

ALTER TABLE ONLY public."Prenotazione"
    ADD CONSTRAINT "Numero_ID_FK" FOREIGN KEY ("Numero_ID")
    REFERENCES public."Cliente"("Numero_ID_Card");

```

4.2.9 Servizio

```
CREATE TABLE public."Servizio" (
    "ID_Cameriere" integer NOT NULL,
    "Codice_Prenotazione" integer NOT NULL
);


ALTER TABLE ONLY public."Servizio"
    ADD CONSTRAINT "Servizio_pkey" PRIMARY KEY ("ID_Cameriere", "Codice_Prenotazione");

CREATE TRIGGER check_servizio BEFORE INSERT ON public."Servizio"
FOR EACH ROW EXECUTE FUNCTION public.check_cameriere_ristorante();

ALTER TABLE ONLY public."Servizio"
    ADD CONSTRAINT "Cameriere_FK" FOREIGN KEY ("ID_Cameriere")
    REFERENCES public."Cameriere"("ID_Cameriere") NOT VALID;

ALTER TABLE ONLY public."Servizio"
    ADD CONSTRAINT "Prenotazione_FK" FOREIGN KEY ("Codice_Prenotazione")
    REFERENCES public."Tavolata"("Codice_Prenotazione") NOT VALID;

```

4.3 Viste

4.3.1 Prenotazioni Ristorante

```
SELECT "Ristorante"."ID_Ristorante",
   "Tavolata"."Codice_Prenotazione"
  FROM "Ristorante"
    JOIN "Sala" ON "Ristorante"."ID_Ristorante" = "Sala"."ID_Ristorante"
    JOIN "Tavolo" ON "Sala"."ID_Sala" = "Tavolo"."ID_Sala"
    JOIN "Tavolata" ON "Tavolo"."Codice_Tavolo" = "Tavolata"."Codice_Tavolo";
```

4.4 Vincoli – Triggers

4.4.1 Check_Adiacenza_Stesso_Tavolo

```
BEGIN
IF NEW."ID_Tavolo" = NEW."ID_Tavolo_Adiacente" THEN
RAISE EXCEPTION 'Un tavolo non può essere adiacente a sè stesso.';
END IF;

RETURN NEW;
END
```

4.4.2 Check_Cameriere_Ristorante

```
BEGIN

IF
(SELECT "Cameriere"."ID_Ristorante" FROM "Cameriere"
WHERE "Cameriere"."ID_Cameriere" = NEW."ID_Cameriere") <> (SELECT "Ristorante"."ID_Ristorante" FROM "Ristorante"
INNER JOIN "Sala" ON "Ristorante"."ID_Ristorante" = "Sala"."ID_Ristorante"
INNER JOIN "Tavolo" ON "Sala"."ID_Sala" = "Tavolo"."ID_Sala"
INNER JOIN "Tavolata" ON "Tavolo"."Codice_Tavolo" = "Tavolata"."Codice_Tavolo"
WHERE "Tavolata"."Codice_Prenotazione" = NEW."Codice_Prenotazione")
THEN
RAISE EXCEPTION 'Il Cameriere non può gestire la tavolata perché registrato a un altro Ristorante.';
END IF;
RETURN NEW;
END
```

4.4.3 Check_Date

```
DECLARE tavolo_trovato integer;
BEGIN
SELECT "Tavolata"."Codice_Tavolo" INTO tavolo_trovato FROM "Tavolata"
WHERE NEW."Data_Arrivo" = "Data_Arrivo"
AND
NEW."Codice_Tavolo" = "Codice_Tavolo";

IF FOUND THEN
RAISE EXCEPTION 'Tavolo occupato per la data inserita.';
END IF;
RETURN NEW;
END
```

4.4.4 Check_Max_Avventori

```
DECLARE max_avventori integer;
BEGIN
SELECT "Tavolo"."Max_Avventori" INTO max_avventori FROM "Tavolo"
INNER JOIN "Tavolata" ON "Tavolo"."Codice_Tavolo" = "Tavolata"."Codice_Tavolo"
INNER JOIN "Prenotazione" ON "Tavolata"."Codice_Prenotazione" = "Prenotazione"."Codice_Prenotazione";

IF
(SELECT COUNT(*) FROM "Prenotazione"
INNER JOIN "Tavolata" ON "Prenotazione"."Codice_Prenotazione" = "Tavolata"."Codice_Prenotazione"
INNER JOIN "Tavolo" ON "Tavolata"."Codice_Tavolo" = "Tavolo"."Codice_Tavolo"
WHERE "Prenotazione"."Codice_Prenotazione" = NEW."Codice_Prenotazione") = max_avventori
THEN
RAISE EXCEPTION 'Numero massimo di avventori raggiunto.';
END IF;
RETURN NEW;
END
```

4.4.5 Check_Tavoli_Sale

```
DECLARE sala1 text;
DECLARE sala2 text;
BEGIN

SELECT "Tavolo"."ID_Sala" INTO sala1 FROM "Tavolo"
WHERE "Tavolo"."Codice_Tavolo" = NEW."ID_Tavolo";

SELECT "Tavolo"."ID_Sala" INTO sala2 FROM "Tavolo"
WHERE "Tavolo"."Codice_Tavolo" = NEW."ID_Tavolo_Adiacente";

IF sala1<>sala2 THEN
RAISE EXCEPTION 'I tavoli non possono essere adiacenti: non sono nella stessa sala';
END IF;
RETURN NEW;
END
```

4.5 Triggers Attributi Calcolati e Trigger semplici

4.5.1 Capienza (Sala) – Insert

```
DECLARE x integer;
BEGIN

SELECT SUM("Max_Avventori") INTO x FROM "Tavolo"
WHERE "Tavolo"."ID_Sala" = NEW."ID_Sala";

UPDATE "Sala" SET "Capienza" = x
WHERE "Sala"."ID_Sala" = NEW."ID_Sala";
RETURN NEW;
END
```

4.5.2 Capienza (Sala) – Delete

```
DECLARE x integer;
BEGIN

SELECT SUM("Max_Avventori") INTO x FROM "Tavolo"
WHERE "Tavolo"."ID_Sala" = OLD."ID_Sala";

UPDATE "Sala" SET "Capienza" = x
WHERE "Sala"."ID_Sala" = OLD."ID_Sala";
RETURN NEW;
END
```

4.5.3 Capienza (Ristorante) – Insert

```
DECLARE x integer;
DECLARE idRist integer;
BEGIN

SELECT "Ristorante"."ID_Ristorante" INTO idRist FROM "Ristorante"
INNER JOIN "Sala" ON "Ristorante"."ID_Ristorante" = "Sala"."ID_Ristorante"
INNER JOIN "Tavolo" ON "Sala"."ID_Sala" = "Tavolo"."ID_Sala"
WHERE "Tavolo"."Codice_Tavolo" = NEW."Codice_Tavolo";

SELECT SUM("Capienza") INTO x FROM "Sala"
WHERE "Sala"."ID_Ristorante" = idRist;

UPDATE "Ristorante" SET "Capienza" = x
WHERE "Ristorante"."ID_Ristorante" = idRist;

RETURN NEW;
END
```

4.5.4 Capienza (Ristorante) – Delete

```
DECLARE x integer;
DECLARE idRist integer;
BEGIN

SELECT "Ristorante"."ID_Ristorante" INTO idRist FROM "Ristorante"
INNER JOIN "Sala" ON "Ristorante"."ID_Ristorante" = "Sala"."ID_Ristorante"
INNER JOIN "Tavolo" ON "Sala"."ID_Sala" = "Tavolo"."ID_Sala"
WHERE "Tavolo"."Codice_Tavolo" = OLD."Codice_Tavolo";

SELECT SUM("Capienza") INTO x FROM "Sala"
WHERE "Sala"."ID_Ristorante" = idRist;

UPDATE "Ristorante" SET "Capienza" = x
WHERE "Ristorante"."ID_Ristorante" = idRist;

RETURN NEW;
END
```

4.5.5 Numero Camerieri – Insert

```
DECLARE x integer;
BEGIN

SELECT COUNT(*) INTO x FROM "Cameriere"
WHERE "Cameriere"."ID_Ristorante" = NEW."ID_Ristorante";

UPDATE "Ristorante" SET "Numero_Camerieri" = x
WHERE "Ristorante"."ID_Ristorante" = NEW."ID_Ristorante";
RETURN NEW;
END
```

4.5.6 Numero Camerieri – Delete

```
DECLARE x integer;
BEGIN

SELECT COUNT(*) INTO x FROM "Cameriere"
WHERE "Cameriere"."ID_Ristorante" = OLD."ID_Ristorante";

UPDATE "Ristorante" SET "Numero_Camerieri" = x
WHERE "Ristorante"."ID_Ristorante" = OLD."ID_Ristorante";
RETURN NEW;
END
```

4.5.7 Numero Tavoli – Insert

```
DECLARE x integer;
BEGIN
SELECT COUNT(*) INTO x FROM "Tavolo"
WHERE "Tavolo"."ID_Sala" = NEW."ID_Sala";

UPDATE "Sala" SET "Numero_Tavoli" = x
WHERE "Sala"."ID_Sala" = NEW."ID_Sala";
RETURN NEW;
END
```

4.5.8 Numero Tavoli – Delete

```
DECLARE x integer;
BEGIN
SELECT COUNT(*) INTO x FROM "Tavolo"
WHERE "Tavolo"."ID_Sala" = OLD."ID_Sala";

UPDATE "Sala" SET "Numero_Tavoli" = x
WHERE "Sala"."ID_Sala" = OLD."ID_Sala";
RETURN NEW;
END
```

4.5.9 Numero Clienti – Insert

```
DECLARE x integer;
BEGIN

SELECT COUNT(*) INTO x FROM "Prenotazione"
WHERE "Prenotazione"."Codice_Prenotazione" = NEW."Codice_Prenotazione";

UPDATE "Tavolata" SET "Numero_Clienti" = x
WHERE "Tavolata"."Codice_Prenotazione" = NEW."Codice_Prenotazione";
RETURN NEW;
END
```

4.5.10 Numero Clienti – Delete

```
DECLARE x integer;
BEGIN

SELECT COUNT(*) INTO x FROM "Prenotazione"
WHERE "Prenotazione"."Codice_Prenotazione" = OLD."Codice_Prenotazione";

UPDATE "Tavolata" SET "Numero_Clienti" = x
WHERE "Tavolata"."Codice_Prenotazione" = OLD."Codice_Prenotazione";
RETURN NEW;
END
```