Sistema di tracciamento contatti per ristoranti

Documentazione del Database dell'applicativo

Traccia 1 – Gruppo OOBD2122_22 – Borrelli Jonathan

1 INDICE

2 DESCRIZIONE DEL PROGETTO

L'applicativo ha lo scopo di tenere traccia di tutti gli avventori di uno o più ristoranti, salvando nel database il tavolo da loro occupato, la data in cui sono arrivati, e i camerieri che li hanno serviti. Esso permette di ottenere anche statistiche sulla quantità totale di avventori di ogni ristorante su base giornaliera, mensile e annuale.

Un gruppo di avventori che occupa un tavolo in un determinato giorno è detto Tavolata.

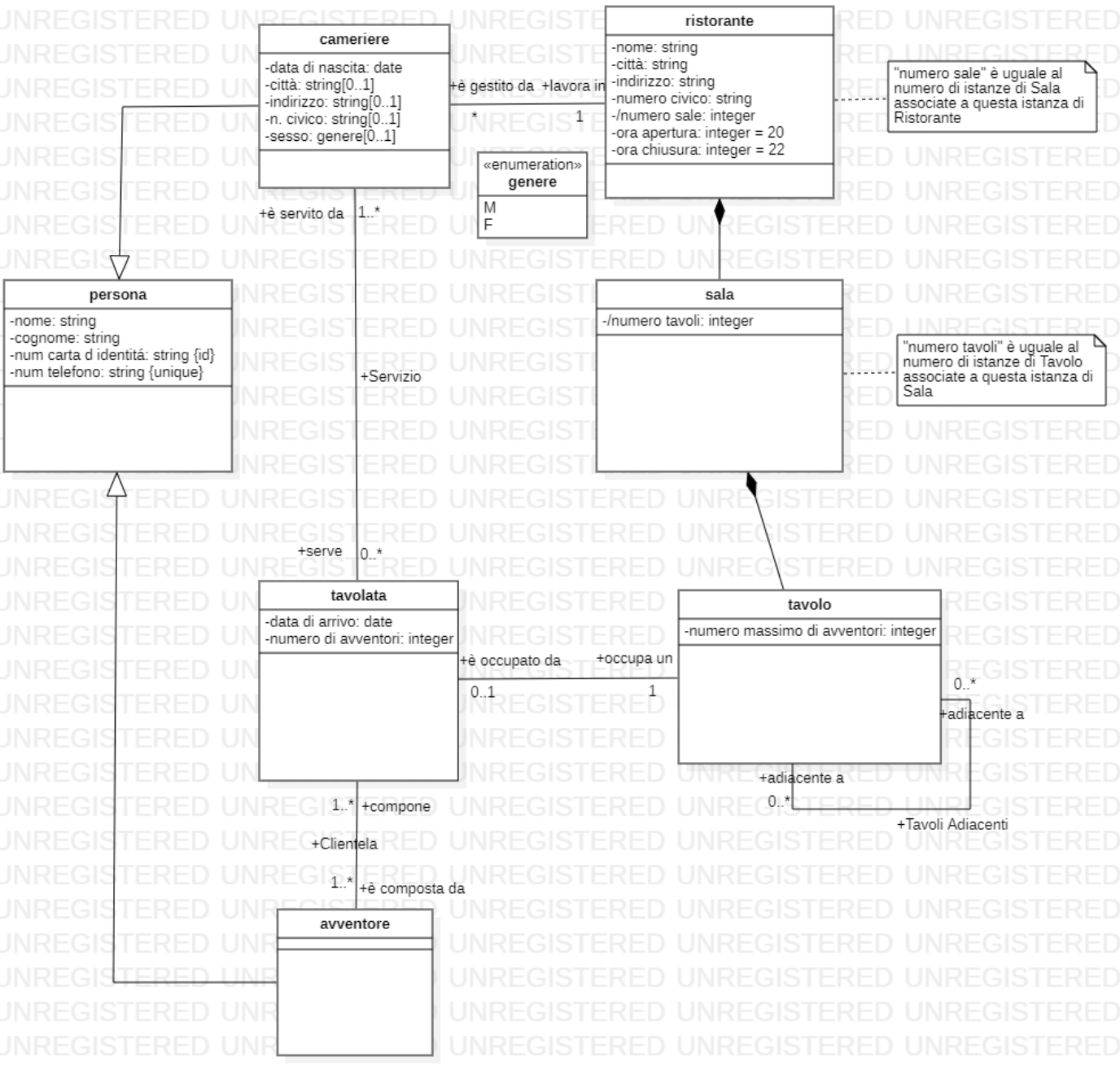
Per convenzione tavoli messi vicino, cioè attaccati, sono considerati come un unico tavolo. Per esempio, due tavoli da 4 posti attaccati dovranno essere inseriti nel database come un unico tavolo da 6 posti.

Il database non prevede che uno stesso tavolo possa essere usato più volte nella stessa giornata da più di tavolate. Quindi, fissata la data, ad un tavolo può essere associata al più una Tavolata.

3 CLASS DIAGRAM

3.1 SCHEMA CONCETTUALE

Al fine di semplificare la lettura dei class diagram UML che seguono si è scelto di adottare la seguente convenzione: Tutti gli attributi, ad eccezioni di quelli in cui specificata, hanno molteplicità pari ad [1].

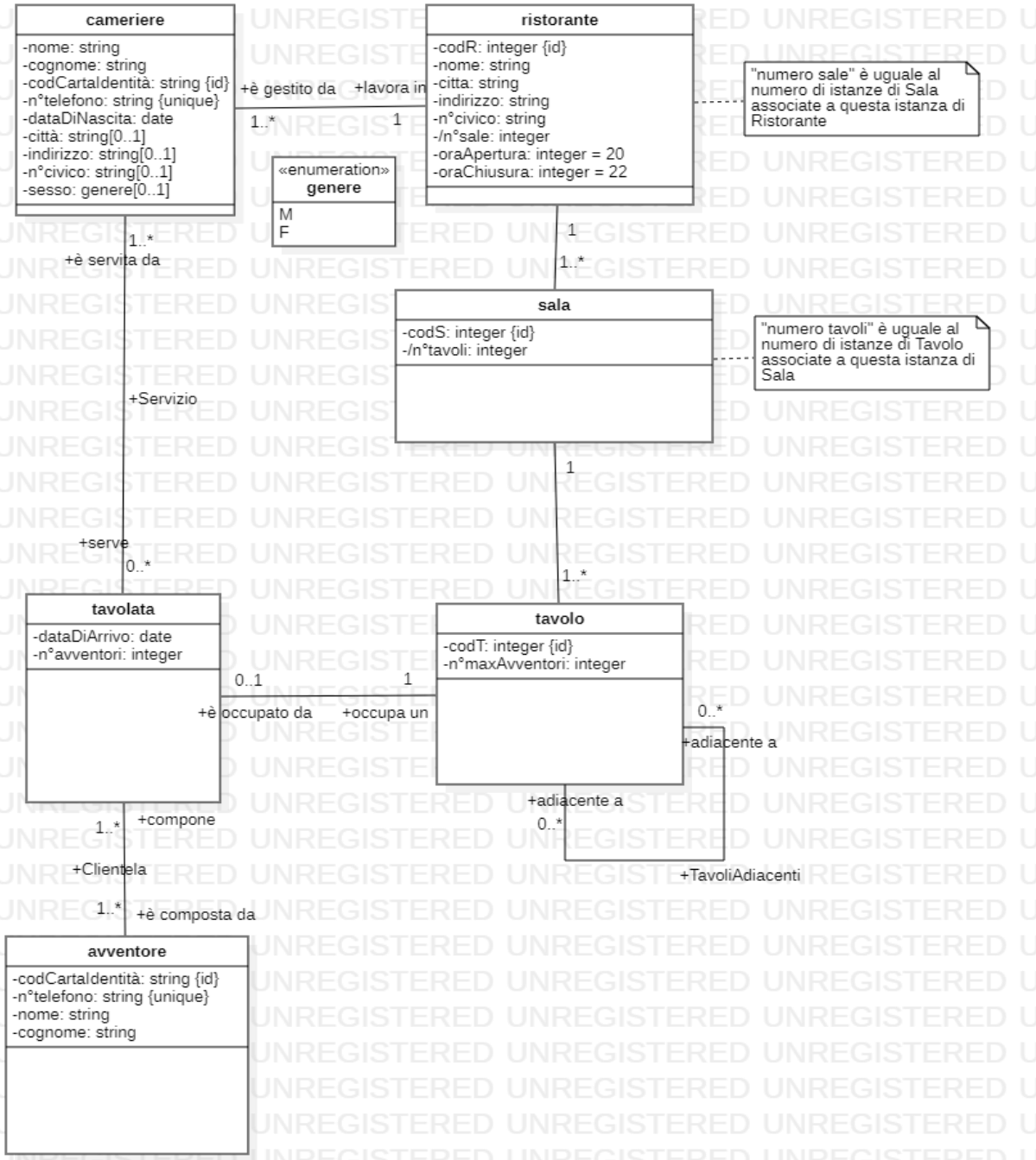


Tab 3.1: Schema concettuale

3.2 SCHEMA RISTRUTTURATO

Cambiamenti:

- Gli attributi scritti con un nome in forma estesa sono stati modificati con un nome più compatto e accettabile dal Database (ES: numero civico -> n°civico)
- Per le classi Ristorante, Sala e Tavolo sono state aggiunte le chiavi surrogate codR, codS e codT come codici identificativi al fine di evitare l'impiego di chiavi primarie composte
- Per quanto concerne la Generalizzazione è stato scelto di eliminare la classe Persona e di inserire tutti i suoi attributi dentro le sue specializzazioni Avventore e Cameriere
- Le composizioni sono state banalmente riportate a delle associazioni 1 a molti, in quanto equivalenti



Tab 3.2: Schema ristrutturato

4 DIZIONARI

4.1 DIZIONARIO DELLE CLASSI

CLASSE	DESCRIZIONE	ATTRIBUTI
RISTORANTE	Descrittore di un ristorante	**codR** (*integer*): chiave tecnica. Identifica univocamente ciascuna istanza di Ristorante. **Nome** (*string*): nome del Ristorante. **Città** (*string*): città di appartenenza del Ristorante. **Indirizzo** (*string*): via o piazza in cui si trova il Ristorante. **N°civico** (*string*): numero civico del Ristorante. Insieme all'indirizzo e alla città identifica la località precisa del Ristorante. **/n°sale** (*integer*): intero positivo corrispondente alla quantità di sale presenti nel Ristorante. **oraApertura** (*integer*): intero da 0 a 23 corrispondente all'ora in cui apre il Ristorante. **oraChiusura** (*integer*): intero da 0 a 23 corrispondente all'ora in cui chiude il Ristorante.
SALA	Descrittore di una sala di un Ristorante	**codS** (*integer*): chiave tecnica. Identifica univocamente ciascuna istanza di Sala. **/n°tavoli** (*integer*): intero positivo corrispondente alla quantità di sale presenti nel Ristorante.
TAVOLO	Descrittore di un tavolo di una Sala	**codT** (*integer*): chiave tecnica. Identifica univocamente ciascuna istanza di Tavolo. **N°maxAvventori** (*integer*): intero positivo corrispondente al numero di posti disponibili occupabili dal Tavolo
TAVOLIADIACENTI	Elenca a coppie di due i tavoli adiacenti. Due tavoli sono considerati adiacenti se sono sufficientemente vicini per un possibile contagio covid.	
TAVOLATA	Descrittore di un tavolo occupato da degli avventori in una certa data	**dataDiArrivo** (*date*): data in cui gli avventori hanno occupato il tavolo.

		N°avventori (*integer*): intero positivo corrispondente alla quantità di avventori che hanno occupato il tavolo.
AVVENTORE	Descrittore di un avventore	**codCartaIdentità** (*string*): chiave tecnica. Stringa da 9 o 10 caratteri che identifica univocamente ciascuna istanza di Avventore. **Nome** (*string*): nome dell'Avventore. **Cognome** (*string*): cognome dell'Avventore. **N°telefono** (*string*): stringa preferibilmente in formato "+39 XXX XXX XXXX", con le X corrispondenti a caratteri numerici, corrispondente al numero di telefono dell'avventore.
CLIENTELA	Elenca tutti gli avventori di un'istanza di Tavolata, per ciascuna Tavolata	
CAMERIERE	Descrittore di un cameriere di un Ristorante	**codCartaIdentità** (*string*): chiave tecnica. Stringa da 9 o 10 caratteri che identifica univocamente ciascuna istanza di Cameriere. **Nome** (*string*): nome del Cameriere. **Cognome** (*string*): cognome del Cameriere. **N°telefono** (*string*): stringa preferibilmente in formato "+39 XXX XXX XXXX", con le X corrispondenti a caratteri numerici, corrispondente al numero di telefono del cameriere. **dataDiNascita** (*date*): data di nascita del Cameriere **città** (*string*): città di residenza del Cameriere. **Indirizzo** (*string*): indirizzo di residenza del Cameriere **n°civico** (*string*): numero civico dell'indirizzo di residenza del Cameriere. **Sesso** (*genere*): sesso del Cameriere
SERVIZIO	Elenca tutte le Tavolate servite da un Cameriere, per ciascun Cameriere	

Tab 4.1: Dizionario delle classi

4.2 DIZIONARIO DELLE ASSOCIAZIONI

ASSOCIAZIONE	DESCRIZIONE	CLASSI COINVOLTE
GESTIONE	Esprime il legame di un ristorante con i camerieri che lavorano lì	**Cameriere [1]** *ruolo* (**lavora in**): indica i camerieri di un ristorante **Ristorante [1..*]** *ruolo* (**è gestito da**): indica il ristorante in cui lavorano i camerieri
SALE	Esprime le sale appartenenti ad un ristorante	**Ristorante [1..*]** *ruolo* (**possiede**): indica il ristorante in cui si trovano le sale **Sala [1]** *ruolo* (**appartiene a**): indica le sale di un ristorante
TAVOLI	Esprime i tavoli appartenenti ad una sala	**Sala [1..*]** *ruolo* (**possiede**): indica la sala in cui si trovano i tavoli **Tavolo [1]** *ruolo* (**appartiene a**): indica i tavoli di una sala
SERVIZIO	Esprime il legame tra le tavolate e i camerieri che le servono	**Cameriere [0..*]** *ruolo* (**serve**): indica i camerieri che servono una o più tavolate **Tavolata [1..*]** *ruolo* (**è servita da**): indica le tavolate servite da uno o più camerieri
CLIENTELA	Esprime il legame tra le tavolate e gli avventori che le occupano	**Tavolata [1..*]** *ruolo* (**è composta da**): indica le tavolate occupate da degli avventori **Avventore [1..*]** *ruolo* (**compone**): indica gli avventori che occupano, in giorni diversi, una o più tavolate
TAVOLIOCCUPATI	Esprime i tavoli occupati da tavolate	**Tavolata [1]** *ruolo* (**occupa**): indica la tavolata che occupa un certo tavolo **Tavolo [0..1]** *ruolo* (**è occupato da**): indica il tavolo occupato da una Tavolata, se è occupato
TAVOLIADIACENTI	Esprime il legame tra un tavolo ed un altro tavolo nel caso in cui siano adiacenti	**Tavolo [0..*]** *ruolo* (**adiacente a**): indica il tavolo adiacente ad un altro tavolo, se ce n'è uno **Tavolo [0..*]** *ruolo* (**adiacente a**): indica il tavolo adiacente ad un altro tavolo, se ce n'è uno

Tab 4.2: Dizionario delle associazioni

4.3 DIZIONARIO DEI VINCOLI

NOME VINCOLO	TIPO	DESCRIZIONE
UNICITÀ POSIZIONE RISTORANTE	Di unicità	Non esistono due istanze della tabella Ristorante aventi la seguente stessa tripla: (città, indirizzo, n°civico)
TAVOLI ADIACENTI STESSA SALA	Inter-relazionale	Se due tavoli sono Adiacenti allora fanno parte della stessa Sala
TAVOLI ADIACENTI PROPRIETÀ SIMMETRICA	Inter-relazionale	Se un tavolo A è adiacente ad un tavolo B, allora anche B è adiacente ad A
NUMERO MASSIMO DI AVVENTORI PER TAVOLATA	Inter-relazionale di dominio	Il numero di avventori di un'istanza di Tavolata non deve superare il numero massimo di avventori dell'istanza di Tavolo corrispondente
UNICITÀ TAVOLATA	Di unicità	Non esistono due istanze della tabella Tavolata aventi la seguente stessa dupla: (dataDiArrivo, codT). Perché in una stessa giornata un tavolo può essere occupato al massimo una volta (vedi pag 1)
UNICITÀ NUMERO DI TELFONO AVVENTORE	Di unicità	Non esistono due istanze della tabella Avventore aventi lo stesso n°telefono. Ogni persona ha un numero di telefono diverso
UNICITÀ NUMERO DI TELEFONO CAMERIERE	Di unicità	Non esistono due istanze della tabella Cameriere aventi lo stesso n°telefono. Ogni persona ha un numero di telefono diverso
UNICITÀ CLIENTE	Di unicità	Non esistono due istanze della tabella Clientela aventi la seguente stessa dupla: (codCartaIdentità, dataDiArrivo). Uno stesso avventore non può trovarsi in due posti contemporaneamente
SERVIZIO CAMERIERE COERENZA	Inter-relazionale	Un cameriere può solo servire i tavoli del ristorante di cui fa parte

Tab 4.3: Dizionario dei vincoli

5 SCHEMA LOGICO

Ristorante(codR, nome, città, indirizzo, n°civico, n°sale, oraApertura, oraChiusura)

Sala(codS, n°tavoli, codR)

Sala(codR) → Ristorante(codR)

Tavolo(codT, n°maxAvventori, codS)

Tavolo(codS) → Sala(codS)

TavoliAdiacenti(tavolo1, tavolo2)

TavoliAdiacenti(tavolo1) → Tavolo(codT)

TavoliAdiacenti(tavolo2) → Tavolo(codT)

Tavolata(dataDiArrivo, n°avventori, codT)

Tavolata(codT) → Tavolo(codT)

Avventore(codCartaIdentità, nome, cognome, n°telefono)

Clientela(codCartaIdentità, dataDiArrivo, codT)

Clientela(codCartaIdentità) → Avventore(codCartaIdentità)

Clientela(dataDiArrivo, codT) → Tavolata(dataDiArrivo, codT)

Cameriere(codCartaIdentità, nome, cognome, n°telefono, dataDiNascita, città, indirizzo, n°civico, sesso, codR)

Cameriere(codR) → Ristorante(codR)

Servizio(codCameriere, dataDiArrivo, codT)

Servizio(codCameriere) → Cameriere(codCartaIdentità)

Servizio(dataDiArrivo, codT) → Tavolata(dataDiArrivo, codT)

6 PROGETTAZIONE FISICA

6.1 ATTRIBUTO CALCOLATO DI RISTORANTE: N°SALE

È possibile ottenere l'attributo n°sale contando il numero di istanze della tabella Sala; dunque, è stato scelto di calcolarlo automaticamente con un trigger dopo un qualsiasi inserimento o cancellazione di Sala, nel seguente modo:

```
1. CREATE TABLE Ristorante
2. (
3.       codR SERIAL PRIMARY KEY,
4.       nome VARCHAR(50) NOT NULL,
5.       città VARCHAR(50) NOT NULL,
6.       indirizzo VARCHAR(100) NOT NULL,
7.       n°civico VARCHAR(5) NOT NULL,
8.       n°sale INTEGER,
9.       oraApertura INTEGER NOT NULL DEFAULT 20,
10.      oraChiusura INTEGER NOT NULL DEFAULT 22,
11.      UNIQUE(città, indirizzo, n°civico)
12.);
13.
14.--Per far sì che n°sale sia un attributo calcolato:
15.CREATE OR REPLACE FUNCTION CalcolaNumeroSaleDi(codRistorante
   Ristorante.codR%TYPE)
16.RETURNS INTEGER
17.AS $$
18.DECLARE ret INTEGER;
19.BEGIN
20.      SELECT N_S.count INTO ret
21.      FROM (
22.            SELECT S.codR, COUNT(S.codS)
23.            FROM Ristorante AS R NATURAL JOIN Sala AS S
24.            GROUP BY S.codR
25.      ) AS N_S
26.      WHERE N_S.codR = codRistorante;
27.      IF ret IS NULL THEN
28.            ret:=0;
29.      END IF;
30.      RETURN ret;
31.END;
32.$$ LANGUAGE plpgsql;
33.
34.CREATE OR REPLACE FUNCTION AggiornaNumeroSale()
35.RETURNS TRIGGER
36.AS $$
37.BEGIN
38.      UPDATE Ristorante
39.      SET n°sale = (SELECT CalcolaNumeroSaleDi(codR))
```

```
40.         WHERE codR = NEW.codR OR codR = OLD.codR;
41.         RETURN NEW;
42. END;
43. $$ LANGUAGE plpgsql;
44.
45. CREATE OR REPLACE TRIGGER AggiornaNumeroSale
46. AFTER INSERT OR DELETE ON Sala
47. FOR EACH ROW
48. EXECUTE FUNCTION AggiornaNumeroSale();
```

6.2 ATTRIBUTO CALCOLATO DI SALA: N°TAVOLI

È possibile ottenere l'attributo n°tavoli contando il numero di istanze della tabella Tavolo; dunque, è stato scelto di calcolarlo automaticamente con un trigger dopo un qualsiasi inserimento o cancellazione di Tavolo, nel seguente modo:

```
1. CREATE TABLE Sala
2. (
3.       codS SERIAL PRIMARY KEY,
4.       n°tavoli INTEGER,
5.       codR INTEGER NOT NULL,
6.       FOREIGN KEY (codR) REFERENCES Ristorante(codR)
7. );
8.
9. --Per far sì che n°tavoli sia un attributo calcolato:
10.CREATE OR REPLACE FUNCTION CalcolaNumeroTavoliDi(codSala Sala.codS%TYPE)
11.RETURNS INTEGER
12.AS $$
13.DECLARE ret INTEGER;
14.BEGIN
15.      SELECT N_T.count INTO ret
16.      FROM (
17.            SELECT S.codS, COUNT(T.codT)
18.            FROM Sala AS S NATURAL JOIN Tavolo AS T
19.            GROUP BY S.codS
20.      ) AS N_T
21.      WHERE N_T.codS = codSala;
22.      IF ret IS NULL THEN
23.            ret:=0;
24.      END IF;
25.      RETURN ret;
26.END;
27.$$ LANGUAGE plpgsql;
28.
29.CREATE OR REPLACE FUNCTION AggiornaNumeroTavoli()
30.RETURNS TRIGGER
31.AS $$
32.BEGIN
33.      UPDATE Sala
34.      SET n°tavoli = (SELECT CalcolaNumeroTavoliDi(codS))
35.      WHERE codS = NEW.codS OR codS = OLD.codS;
36.      RETURN NEW;
37.END;
38.$$ LANGUAGE plpgsql;
39.
40.CREATE OR REPLACE TRIGGER AggiornaNumeroTavoli
41.AFTER INSERT OR DELETE ON Tavolo
42.FOR EACH ROW
43.EXECUTE FUNCTION AggiornaNumeroTavoli()
```

6.3 DATO RIDONDANTE: TAVOLO ADIACENTE CON SÉ STESSO IN TAVOLIADIACENTI

Un tavolo tecnicamente è sempre adiacente con sé stesso, ma è un'informazione inutile da salvare nel database, quindi è stato deciso di ignorare un qualsiasi inserimento di tale dato, in modo anche da poter evitare di dover trattare questo caso. Per farlo è stato realizzato un trigger che, al posto di un qualsiasi inserimento di una coppia identica, non fa nulla:

```
1. CREATE TABLE TavoliAdiacenti
2. (
3.       tavolo1 INTEGER NOT NULL,
4.       tavolo2 INTEGER NOT NULL,
5.       FOREIGN KEY(tavolo1) REFERENCES Tavolo(codT),
6.       FOREIGN KEY(tavolo2) REFERENCES Tavolo(codT),
7.       UNIQUE(tavolo1, tavolo2)
8. );
9.
10.--per ignorare gli inserimenti di coppie identiche:
11.CREATE OR REPLACE FUNCTION niente()
12.RETURNS TRIGGER
13.AS $$
14.BEGIN
15.      RETURN NULL;
16.END;
17.$$ LANGUAGE plpgsql;
18.
19.CREATE OR REPLACE TRIGGER RidondanzaTavoloAdiacenteConSeStesso
20.BEFORE INSERT OR UPDATE ON TavoliAdiacenti
21.FOR EACH ROW
22.WHEN (NEW.Tavolo1=NEW.Tavolo2)
23.EXECUTE FUNCTION niente()
```

6.4 TABELLA CALCOLATA: TAVOLATA

È possibile ottenere tutti i dati presenti nella tabella Tavolata a partire dalla tabella Clientela, contando il numero di istanze di una stessa coppia (codT, dataDiArrivo), per ognuna di queste coppie, per contare il numero di avventori di ogni tavolata. Dunque è stato scelto di popolarla automaticamente con un trigger dopo un qualsiasi statement di inserimento o cancellazione di Clientela. Quindi non serve che l'utente popoli o aggiorni questa tabella.

```
1. CREATE TABLE Tavolata
2. (
3.       dataDiArrivo DATE NOT NULL,
4.       n°avventori INTEGER NOT NULL,
5.       codT INTEGER NOT NULL,
6.       FOREIGN KEY(codT) REFERENCES Tavolo(codT),
7.       PRIMARY KEY(DataDiArrivo, codT)
8. );
9. CREATE TABLE Clientela
10.(
11.      codCartaIdentità VARCHAR(12) NOT NULL,
12.      dataDiArrivo DATE NOT NULL,
13.      codT INTEGER NOT NULL,
14.      FOREIGN KEY(CodCartaIdentità) REFERENCES Avventore(codCartaIdentità),
15.      UNIQUE(codCartaIdentità, dataDiArrivo)
16.);
17.
18.--Per far sì che Tavolata sia una tabella calcolata:
19.CREATE OR REPLACE FUNCTION AggiornaTavolata()
20.RETURNS TRIGGER
21.AS $$
22.BEGIN
23.      ALTER TABLE Servizio
24.      DROP CONSTRAINT servizio_datadiarrivo_codt_fkey;
25.      DELETE FROM Tavolata;
26.      INSERT INTO Tavolata(dataDiArrivo, n°avventori, codT) (
27.            SELECT dataDiArrivo, COUNT(codCartaIdentità), codT
28.            FROM Clientela
29.            GROUP BY dataDiArrivo, codT
30.      );
31.      ALTER TABLE Servizio
32.      ADD CONSTRAINT servizio_datadiarrivo_codt_fkey FOREIGN KEY(dataDiArrivo,
   codT) REFERENCES Tavolata(dataDiArrivo, codT);
33.      RETURN NEW;
34.END;
35.$$ LANGUAGE plpgsql;
36.
37.CREATE OR REPLACE TRIGGER AggiornaTavolata
38.AFTER INSERT OR DELETE OR UPDATE OF dataDiArrivo, codT ON Clientela
39.FOR EACH STATEMENT
40.EXECUTE FUNCTION AggiornaTavolata();
```

6.5 VINCOLO: NUMERO MASSIMO DI AVVENTORI PER TAVOLATA

Trigger che lancia un'eccezione alla prima istanza della tabella Tavolata avente n°avventori > n°maxAvventori del tavolo corrispondente:

```
1. CREATE TABLE Tavolata
2. (
3.       dataDiArrivo DATE NOT NULL,
4.       n°avventori INTEGER NOT NULL,
5.       codT INTEGER NOT NULL,
6.       FOREIGN KEY(codT) REFERENCES Tavolo(codT),
7.       PRIMARY KEY(DataDiArrivo, codT)
8. );
9.
10.CREATE TABLE Tavolo
11.(
12.      codT SERIAL PRIMARY KEY,
13.      n°maxAvventori INTEGER NOT NULL,
14.      codS INTEGER NOT NULL,
15.      FOREIGN KEY (codS) REFERENCES Sala(codS)
16.);
17.
18.--Per realizzare il vincolo:
19.CREATE OR REPLACE FUNCTION n°MassimoDiAvventoriSuperato()
20.RETURNS TRIGGER
21.AS $$
22.BEGIN
23.      IF NEW.n°avventori > (SELECT n°maxAvventori FROM Tavolo WHERE codT =
   NEW.codT) THEN
24.            RAISE EXCEPTION 'Il tavolo di codice % contiene un
25.            numero insufficiente di posti
26.            per contenere % avventori', NEW.codT, NEW.n°avventori;
27.      END IF;
28.      RETURN NEW;
29.END;
30.$$ LANGUAGE plpgsql;
31.
32.CREATE OR REPLACE TRIGGER n°MassimoDiAvventoriSuperato
33.BEFORE INSERT OR UPDATE ON Tavolata
34.FOR EACH ROW
35.EXECUTE PROCEDURE n°MassimoDiAvventoriSuperato();
36.
```

6.6 VINCOLO: TAVOLI ADIACENTI STESSA SALA

Trigger che lancia un'eccezione alla prima istanza della tabella TavoliAdiacenti aventi due tavoli con due sale diverse:

```
1. CREATE TABLE TavoliAdiacenti
2. (
3.       tavolo1 INTEGER NOT NULL,
4.       tavolo2 INTEGER NOT NULL,
5.       FOREIGN KEY(tavolo1) REFERENCES Tavolo(codT),
6.       FOREIGN KEY(tavolo2) REFERENCES Tavolo(codT),
7.       UNIQUE(tavolo1, tavolo2)
8. );
9.
10.CREATE TABLE Tavolo
11.(
12.      codT SERIAL PRIMARY KEY,
13.      n°maxAvventori INTEGER NOT NULL,
14.      codS INTEGER NOT NULL,
15.      FOREIGN KEY (codS) REFERENCES Sala(codS)
16.);
17.
18.--Per realizzare il vincolo:
19.CREATE OR REPLACE FUNCTION TavoliAdiacentiDevonoAppartenereAllaStessaSala()
20.RETURNS TRIGGER
21.AS $$
22.BEGIN
23.      IF (SELECT codS FROM Tavolo WHERE codT=NEW.tavolo1) <> (SELECT codS FROM
   Tavolo WHERE codT=NEW.tavolo2) THEN
24.            RAISE EXCEPTION 'Il tavolo % non può essere adiacente al tavolo
   %', NEW.tavolo1, NEW.tavolo2;
25.      END IF;
26.      RETURN NEW;
27.END;
28.$$ LANGUAGE plpgsql;
29.
30.CREATE OR REPLACE TRIGGER TavoliAdiacentiDevonoAppartenereAllaStessaSala
31.BEFORE INSERT OR UPDATE ON TavoliAdiacenti
32.FOR EACH ROW
33.EXECUTE PROCEDURE TavoliAdiacentiDevonoAppartenereAllaStessaSala();
```

6.7 VINCOLO: TAVOLI ADIACENTI PROPRIETÀ SIMMETRICA

Trigger che lancia un'eccezione se data una coppia (tavolo1, tavolo2) della tabella TavoliAdiacenti, non esiste la coppia (tavolo2, tavolo1) nella tabella TavoliAdiacenti:

```
1. CREATE TABLE TavoliAdiacenti
2. (
3.       tavolo1 INTEGER NOT NULL,
4.       tavolo2 INTEGER NOT NULL,
5.       FOREIGN KEY(tavolo1) REFERENCES Tavolo(codT),
6.       FOREIGN KEY(tavolo2) REFERENCES Tavolo(codT),
7.       UNIQUE(tavolo1, tavolo2)
8. );
9.
10.--Per realizzare il vincolo:
11.CREATE OR REPLACE FUNCTION ProprietàSimmetricaTavoliAdiacenti()
12.RETURNS TRIGGER
13.AS $$
14.DECLARE
15.      tavoloA INTEGER;
16.      tavoloB INTEGER;
17.BEGIN
18.      SELECT tavolo1, tavolo2 INTO tavoloA, tavoloB
19.      FROM TavoliAdiacenti
20.      WHERE (tavolo2, tavolo1) NOT IN (SELECT* FROM TavoliAdiacenti);
21.      IF tavoloA IS NOT NULL THEN
22.            RAISE EXCEPTION 'Il tavolo % è adiacente al tavolo %, ma non
   viceversa', tavoloA, tavoloB;
23.      END IF;
24.      RETURN NEW;
25.END;
26.$$ LANGUAGE plpgsql;
27.
28.CREATE OR REPLACE TRIGGER ProprietàSimmetricaTavoliAdiacenti
29.AFTER INSERT OR UPDATE ON TavoliAdiacenti
30.FOR EACH STATEMENT
31.EXECUTE FUNCTION ProprietàSimmetricaTavoliAdiacenti();
```

6.8 VINCOLO: SERVIZIO CAMERIERE COERENZA

Un cameriere, appartenente ad un certo ristorante di codice XX, può solo servire i tavoli aventi codR XX. Trigger che lancia un'eccezione alla prima istanza di Servizio aventi cameriere e tavolo con due codici del ristorante diversi:

```
1. CREATE TABLE Servizio
2. (
3.       codCameriere VARCHAR(12) NOT NULL,
4.       dataDiArrivo DATE NOT NULL,
5.       codT INTEGER NOT NULL,
6.       FOREIGN KEY(codCameriere) REFERENCES Cameriere(codCartaIdentità),
7.       CONSTRAINT servizio_datadiarrivo_codt_fkey FOREIGN KEY(dataDiArrivo,
   codT) REFERENCES Tavolata(dataDiArrivo, codT)
8. );
9.
10.CREATE TABLE Cameriere
11.(
12.      codCartaIdentità VARCHAR(12) PRIMARY KEY,
13.      nome VARCHAR(50) NOT NULL,
14.      cognome VARCHAR(50) NOT NULL,
15.      n°telefono VARCHAR(15) NOT NULL,
16.      dataDiNascita DATE NOT NULL,
17.      città VARCHAR(50),
18.      indirizzo VARCHAR(100),
19.      n°civico VARCHAR(5),
20.      sesso genere,
21.      codR INTEGER NOT NULL,
22.      FOREIGN KEY(codR) REFERENCES Ristorante(codR),
23.      UNIQUE(n°telefono)
24.);
25.
26.CREATE TABLE Ristorante
27.(
28.      codR SERIAL PRIMARY KEY,
29.      nome VARCHAR(50) NOT NULL,
30.      città VARCHAR(50) NOT NULL,
31.      indirizzo VARCHAR(100) NOT NULL,
32.      n°civico VARCHAR(5) NOT NULL,
33.      n°sale INTEGER,
34.      oraApertura INTEGER NOT NULL DEFAULT 20,
35.      oraChiusura INTEGER NOT NULL DEFAULT 22,
36.      UNIQUE(città, indirizzo, n°civico)
37.);
38.
39.CREATE TABLE Sala
40.(
41.      codS SERIAL PRIMARY KEY,
42.      n°tavoli INTEGER,
```

```
43.       codR INTEGER NOT NULL,
44.       FOREIGN KEY (codR) REFERENCES Ristorante(codR)
45.);
46.
47.CREATE TABLE Tavolo
48.(
49.       codT SERIAL PRIMARY KEY,
50.       n°maxAvventori INTEGER NOT NULL,
51.       codS INTEGER NOT NULL,
52.       FOREIGN KEY (codS) REFERENCES Sala(codS)
53.);
54.
55.--Per realizzare il vincolo:
56.CREATE OR REPLACE FUNCTION TavoliServitiDaCamerieraEstraneo()
57.RETURNS TRIGGER
58.AS $$
59.DECLARE
60.       codR_Cameriere INTEGER;
61.       codR_Tavolo INTEGER;
62.BEGIN
63.       SELECT codR INTO codR_Cameriere
64.       FROM Cameriere
65.       WHERE codCartaIdentità=NEW.codCameriere;
66.       SELECT codR INTO codR_Tavolo
67.       FROM Tavolo NATURAL JOIN Sala NATURAL JOIN Ristorante
68.       WHERE codT=NEW.codT;
69.       IF codR_Cameriere <> codR_Tavolo THEN
70.             RAISE EXCEPTION 'Il cameriere % (appartente al ristorante %) non
   può servire il tavolo % (appartenente
71.             al ristorante %)', NEW.codCameriere, codR_Cameriere, NEW.codT,
   codR_Tavolo;
72.       END IF;
73.       RETURN NEW;
74.END;
75.$$ LANGUAGE plpgsql;
76.
77.CREATE OR REPLACE TRIGGER TavoliServitiDaCamerieraEstraneo
78.BEFORE INSERT OR UPDATE ON Servizio
79.FOR EACH ROW
80.EXECUTE FUNCTION TavoliServitiDaCamerieraEstraneo();
```