UNIVERSITÀ DEGLI STUDI DI NAPOLI PARTHENOPE
DIPARTIMENTO DI SCIENZE E TECNOLOGIE

**CORSO DI BASI DI DATI E LABORATORIO**


# LoSfizioDB

**PROPONENTE:**

Franco Riformato 0124 / 001968

[Progetto Ridotto]

**DATA DI CONSEGNA:**

21/09/2020

**ANNO ACCADEMICO:**

2019 - 2020

**CATEGORIA:**

Ristorazione: Gestione di clienti, Gestione del personale

# Indice

## Progettazione



## Implementazione



## Note

# Elenco delle figure



# Elenco delle tabelle



# Elenco degli script



***"Popolamento LoSfizioDB.sql"* non è stato incluso nella documentazione.

Il popolamento di base è visionabile nel file appena citato, oppure direttamente nel file *"Ricostruzione completa DB.sql".*

# Progettazione

## Introduzione

Il ristorante "Lo Sfizio", luogo di ristoro di ridotte dimensioni, chiede di gestire al meglio l'afflusso di clienti, i loro ordini (in sala e da asporto) e i turni dei dipendenti, stando attenti a quando vengono rispettati.

Il locale è composto da un bancone con cassa (la quale viene utilizzata da un cameriere o, molto spesso, dallo stesso proprietario del ristorante), tre sale con dieci tavoli in totale (nella prima e nella terza sala serve un solo cameriere, mentre nella seconda sono necessari due camerieri), una cucina (nella quale operano i due cuochi).

## Analisi dei requisiti

Si vuole progettare una Base Di Dati per il ristorante "Lo Sfizio", il quale richiede di tenere conto principalmente di:

- Distinguere gli ordini dei clienti, sapere cosa contengono e visualizzarli quando necessario;
- Avere un menù di prodotti attualmente in vendita aggiornabile dal proprietario;
- Gestire il conto, sia per i tavoli che per gli ordini da asporto;
- Conoscere quali tavoli sono occupati in un determinato momento;
- Gestire le prenotazioni dei clienti;
- Tenere traccia dei dipendenti, dei turni da loro seguiti e dei permessi richiesti;

Il ristorante è, di norma, aperto dalle 18:30 alle 00:00.
Il proprietario richiede di poter visualizzare tutti i dati, oltre che poter inserire i turni dei propri dipendenti.

I clienti non hanno bisogno di accesso diretto al database:

- Per effettuare l'ordine, i vari prodotti saranno inseriti direttamente dal cameriere che riporterà l’ordine al cassiere, sotto richiesta del cliente;
- Nell'ipotesi in cui volessero un riepilogo di cosa contiene il proprio ordine dovrebbero chiedere al cameriere;
- Per inserire o disdire una prenotazione, dovrebbero chiamare il ristorante oppure presentarsi alla cassa;

Le prenotazioni dei clienti prenotanti devono includere il numero dei partecipanti al tavolo e (in maniera opzionale) se il cliente preferisce un tavolo all’aperto o un tavolo all’interno.

I dipendenti devono avere un’informazione di contatto obbligatoria, che può essere una email o un numero di telefono.

Alcuni dipendenti possiedono delle qualifiche professionali delle quali tener conto.

I permessi vengono richiesti un giorno per volta dai dipendenti.

La sala n.1 è completamente all’esterno ed è consentito fumare, mentre nella n.2 non è consentito fumare e ha un tavolo su un piccolo balcone e gli altri tavoli all’interno, la n.3 ha uno spazio dedicato ad un solo tavolo all’interno ma ha maggiore spazio su una terrazza e, in questa sala, è consentito fumare.

# Glossario

Raccolta dei termini specifici del settore, affiancati dalla loro definizione.

Tabella 1: Glossario

| TERMINE | DEFINIZIONE |
|---|---|
| Cliente | Persona che si avvale delle prestazioni del ristorante, indipendentemente che sia saltuariamente o abitualmente. Può occupare un tavolo, oppure richiedere un asporto, cioè portare via l'ordine richiesto. |
| Ordine | La richiesta del cliente: pietanze preparate o bevande vendute dal ristorante. |
| Menù | Lista dei prodotti in vendita che possono essere ordinati dai clienti. Possono essere modificati dal proprietario. |
| Conto | Contributo da pagare da parte dei clienti per usufruire dei servizi del ristorante, sia da asporto che in sala. |
| Tavolo | Posizione attrezzata per ospitare i clienti. Ogni tavolo può ospitare un numero massimo di avventori. |
| Prenotazione | Richiesta al ristorante dal cliente di riservargli un tavolo, in una determinata data e ora, per un determinato numero di persone. |
| Dipendente | Persona che lavora sotto le direttive del proprietario del ristorante. E' tenuta a seguire determinati turni e a far sapere con anticipo se non si presenterà in tali date. Si dividono in Camerieri e Cuochi per il ristorante "Lo Sfizio". |
| Turno | Periodo di tempo assegnato all'attività lavorativa di un Cameriere o un Cuoco. Comprende una durata in ore, una specifica data e orario di inizio. |
| Permesso | Richiesta di non presentarsi all'attività lavorativa da parte di un Dipendente. |

# Diagramma EE/R

Il diagramma EE/R (visibile in [Figura 1] e nel file *“EER LoSfizioDB.png”*) permette di esprimere la situazione del ristorante, con le entità, le associazioni e gli attributi necessari alla creazione del suo database.

L’entità “Tavolo” è la lista dei tavoli presenti nel ristorante “Lo Sfizio”, essa è l’entità di maggiore rilevanza per l’individuazione degli ordini: sulla base del numero del tavolo (NumTavolo), possiamo definire “Ordine” come entità debole (identificata da NumTavolo + DataOraOrdine).

Se un cliente dovesse presentarsi due volte nella stessa sera, sarebbe considerato come un cliente nuovo, al quale viene assegnato un nuovo ordine (con corrispettivi NumTavolo e DataOraOrdine).

Possiamo essere a conoscenza degli alimenti presenti in un ordine tramite l’associazione “Contiene”.

“Conto” risulta essere la lista delle fatture emesse, quindi, con l’associazione “Paga”, abbiamo gli ordini che sono stati pagati.

“ClientePrenotante” è un’entità utile per conservare i dati di contatto dei clienti che effettuano una prenotazione, cioè un ordine con data, ora, numero del tavolo assegnati durante il dialogo con il cliente.

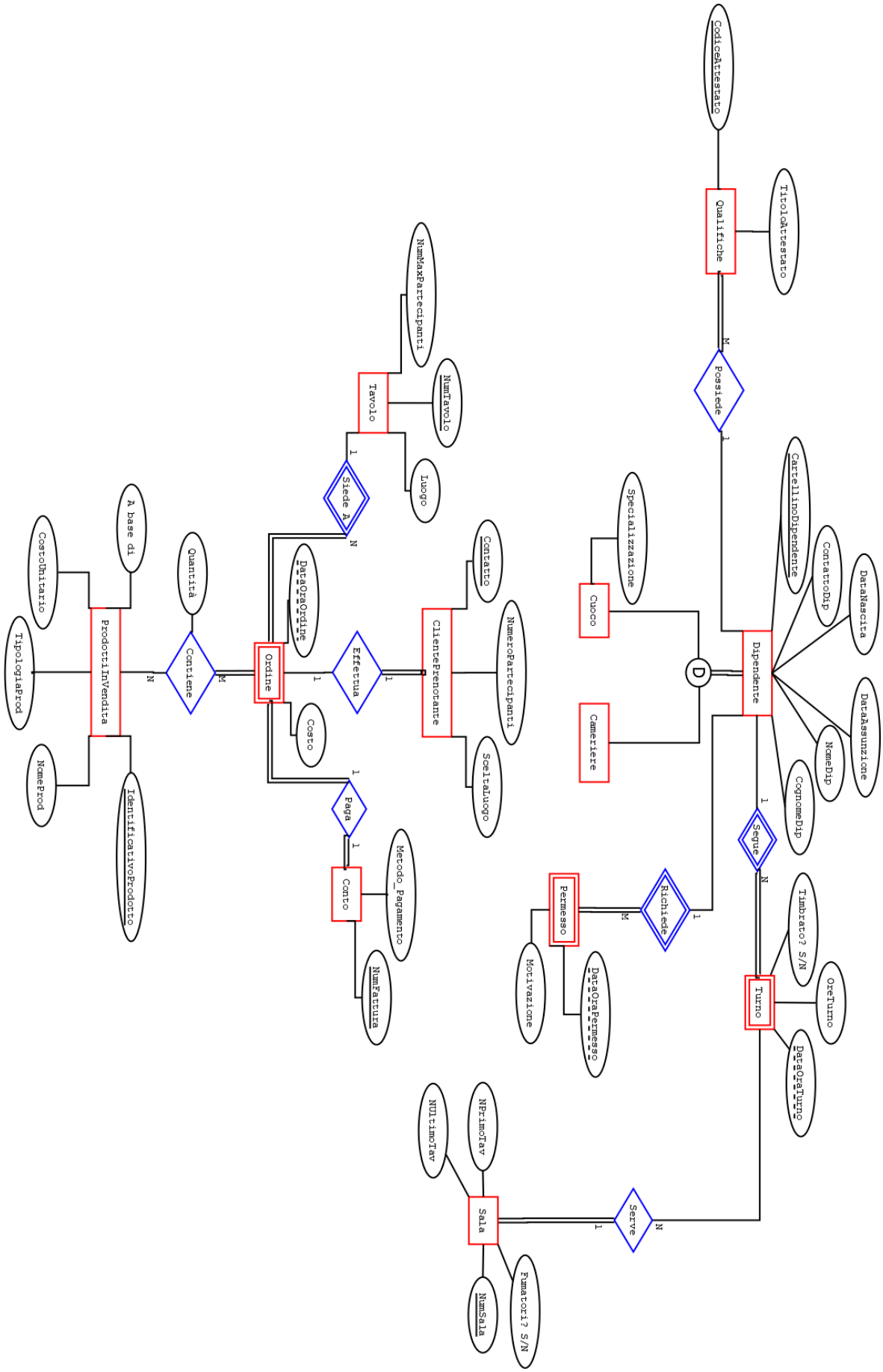
L’entità “Sala” dà informazioni sulla disposizione dei tavoli.

Per la gestione dei dipendenti, abbiamo:

- l’entità “Dipendente” con tutti gli attributi che è importante conservare relativi al lavoratore;
- l’entità debole “Turno”, cioè gli orari di lavoro dei dipendenti, definita da CartellinoDipendente+DataOraTurno;
- l’entità debole “Permesso”, cioè l’elenco dei permessi richiesti dai dipendenti, definita da CartellinoDipendente+DataOraPermesso;
- l’entità “Qualifiche”, che permette di conservare l’elenco degli attestati dei dipendenti;
- l’entità “Serve”, la quale permette di associare ad ogni turno di un cameriere una specifica sala.

Inoltre, l’entità “Dipendente” viene specializzata in “Cuoco” e “Cameriere”, dato che bisogna registrare il fatto che ogni cuoco ha la propria specializzazione culinaria.

N.B. Le considerazioni su Totalità e Molteplicità e la loro traduzione da EE/R a relazionale sono incluse nella documentazione a pagina 18-19 e nel file “*Sulle Totalità e Molteplicità.txt*”.




[Figura 1: Diagramma EE/R]

# Diagramma relazionale

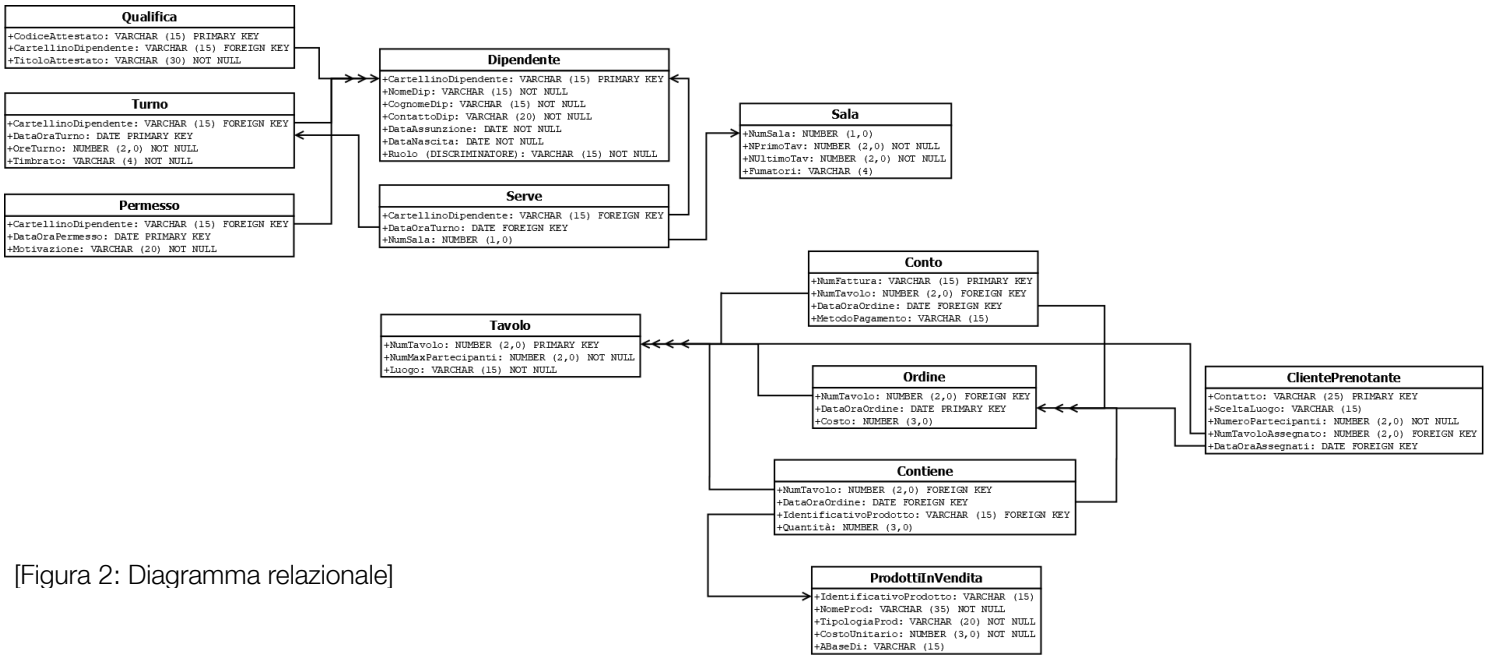
Il diagramma relazionale è presente in [Figura 2] e visionabile nel file *“TradRelazionale LoSfizioDB.png”.*

In questo diagramma, vi sono i nomi delle entità usate per la codifica in SQL e i nomi dei loro attributi, specificandone il tipo e sottolineando le chiavi primarie e le chiavi esterne.

Possiamo notare che non sono state tradotte in tabelle le entità:

- “Siede A”, al fine di tradurre la totalità “tutti gli Ordini siedono ad un Tavolo” (gli ordini da asporto siedono, per convenzione, al tavolo numero 0);
- “Paga”, per la totalità “tutti i Conti sono pagati da Ordini”;
- “Effettua”, per la totalità “tutti i ClientiPrenotanti effettuano un Ordine”;
- “Segue”, per tradurre la totalità “tutti i Turni sono seguiti da Dipendenti”;
- “Richiede”, dato che “tutti i Permessi sono richiesti da Dipendenti”;
- “Possiede”, dato che “tutte le Qualifiche sono possedute da Dipendenti”.

La specializzazione di “Dipendente” è stata tradotta con l’attributo “Ruolo” su Dipendente, evitando di creare due tabelle (una per “Cuoco”, l’altra per “Cameriere”).




[Figura 2: Diagramma relazionale]

# Utenti e categorie

Le operazioni dei Clienti (prenotazione di un tavolo, effettuare un ordine ecc..) avvengono tutte tramite l'utente "Cassiere" (il quale ha un volume di 4 per distinguere i 4 diversi camerieri che si alternano alla cassa) oppure tramite l'utente "Proprietario", il quale molto spesso si dedica alla gestione della cassa.

L'utente "Proprietario" possiede tutti i permessi, in quanto potrebbe voler modificare sia le tabelle gestite da "Cassiere" sia quelle relative alla gestione del personale sia quella relativa al menù e quella relativa alla disposizione dei tavoli.

Gli utenti vengono descritti nella [Tabella 2].

| UTENTE | TIPO | VOLUME | PERMESSI |
|---|---|---|---|
| DB_LoSfizio | Amministratore | 1 | ALL |
| Proprietario | Comune | 1 | ALL |
| Cassiere | Comune | 4 | ALL ON TAVOLO,<br>ALL ON ORDINE,<br>ALL ON CONTO,<br>ALL ON CONTIENE,<br>ALL ON CLIENTEPRENOTANTE |

[Tabella 2: Utenti]

# Operazioni degli utenti

La maggior parte delle operazioni è eseguibile con comandi DML, alcuni esempi:

- Entra un cliente ed effettua un ordine, sedendosi al tavolo: INSERT su ORDINE, INSERT su CONTIENE;
- Un cliente prenota un tavolo: viene chiesto al cliente per quando prenotare, per quante persone e se preferisce un posto all'aperto o all'interno, quindi INSERT su ORDINE (il costo di default sarà 0€, sarà poi aggiornato quando il cliente si presenterà per la consumazione) e su CLIENTEPRENOTANTE;
- Un cliente effettua un ordine, da asporto: INSERT su ORDINE (con NumTavolo=0) e su CONTIENE;
- Un cliente paga l'ordine: INSERT su Conto;
- Cambiare il prezzo di un prodotto sul menù: UPDATE PRODOTTIINVENDITA SET CostoUnitario = <valore desiderato> WHERE IdentificativoProdotto = <nome prodotto del quale modificare il prezzo>;
- Aggiungere un turno per un dipendente: INSERT su Turno;
- Un dipendente richiede un permesso: INSERT su PERMESSO;
- Aggiungere un tavolo ad una sala: UPDATE su Sala, stando attenti che il NumSala può essere solo un numero compreso tra 1 e 3. Sarà possibile visualizzare il numero dei tavoli in una sala con una vista descritta successivamente.

# Volumi

La tabella dei volumi [Tabella 3] permette di rappresentare quante tuple ci si aspetta che siano inserite nelle principali tabelle del DB, una volta che questo sia in pieno regime di funzionamento.

Si potrebbe decidere di eliminare le tuple obsolete ogni settimana (sulle tabelle ORDINE, CONTIENE, CONTO, CLIENTEPRENOTANTE) in caso non ci sia bisogno di un monitoraggio continuo dell'attività del ristorante.

Mentre invece potrebbe essere utile tenere traccia del contenuto delle tabelle DIPENDENTE, TURNO, SERVE, PERMESSO (per sapere se i dipendenti si presentano regolarmente al lavoro e quanti permessi richiede ogni dipendente)

Le tabelle QUALIFICA e SALA rimarranno pressoché invariate nel tempo, tranne qualche saltuario aggiornamento in caso in cui i dipendenti decidano di seguire corsi di qualificazione professionale o in caso in cui si decida di spostare dei tavoli.

| TABELLA | TIPO | VOLUME | INCREMENTO | PERIODO |
|---|---|---|---|---|
| ORDINE | Entità | 5800* annuo | 450** | mese |
| CONTO | Entità | 5800* annuo | 450** | mese |
| CONTIENE | Associazione | 14.400*** annuo | 1200*** | mese |
| CLIENTEPRENOTANTE | Entità | 2900**** annuo | 240**** | Mese |

[Tabella 3: Volumi]

*ca. 20 Clienti per serata *288 Giorni lavorativi in un anno = 5760
**ca. 20 Clienti per serata * 24 Giorni lavorativi in un mese = 480
***se ogni sera, ci sono circa 50 INSERT su Contiene
****ca., se ogni giorno dovessero prenotare 10 ClientiPrenotanti * 288 = 2880
****ca., se ogni giorno dovessero prenotare 10 ClientiPrenotanti * 24 = 240

Alla luce di quanto mostrato nella [Tabella 3] è fortemente consigliato eliminare le tuple non più utili del DB ogni settimana.

In particolare, si consiglia di eliminare settimanalmente (o anche ogni 3-4 giorni) le tuple da: ORDINE, CONTIENE, CONTO, CLIENTEPRENOTANTE (considerato che sono dati utili soltanto la sera stessa e nell'arco di tempo in cui il cliente consuma il suo ordine).

Settimanalmente (in 6 giorni lavorativi), si avrebbero ca. 120 tuple su ORDINE e su CONTO, 300 tuple su CONTIENE e 60 tuple su CLIENTEPRENOTANTE.

# Vincoli di integrità

Servono per limitare i valori che alcuni attributi possono assumere.
I vincoli di integrità implementati sono di tipo “statico”:

- Per l’attributo “Luogo” su “Tavolo” è consentito solo “Interno” o “Esterno”;
- Per l’attributo “MetodoPagamento” su “Ordine” è consentito solo “Contanti” o “Bancomat”;
- Per l’attributo “TipologiaProdotto” su “ProdottiInVendita” sono consentiti “Antipasto”, “Primo”, “Secondo”, “Dolce”, “Vino Bianco”, “Vino Rosso”, “Bevanda”;
- Per l’attributo “SceltaLuogo” su “ClientePrenotante” è consentito solo “Interno” o “Esterno”;
- Per l’attributo “Ruolo” su “Dipendente” è consentito solo “Cameriere” o “Cuoco”;
- Per l’attributo “Timbrato” su “Turno” è consentito solo “Si” o “No”;
- Per l’attributo “Motivazione” su “Permesso” è consentito “Salute” o “Personale”.

Tutti gli attributi sopraelencati sono anche dichiarati come NOT NULL, eccetto che “SceltaLuogo”.

A. Altri attributi NOT NULL sono:
(TAVOLO) “NumMaxPartecipanti”;
(PRODOTTIINVENDITA) “NomeProd”, “CostoUnitario”;
(CLIENTEPRENOTANTE) “NumeroPartecipanti”;
(DIPENDENTE) “NomeDip”, “CognomeDip”, “ContattoDip”, “DataAssunzione”, “DataNascita”;
(TURNO) “OreTurno”;
(SALA) “NPrimoTav”, “NUltimoTav”;
(QUALIFICA) “TitoloAttestato”.

B. Inoltre, per alcuni valori è stato impostato un valore di DEFAULT:
(ORDINE) “DataOraOrdine” ha come default la data di sistema (SYSDATE) nel momento in cui viene inserita la tupla, “Costo” ha come default “0”;
(CONTIENE) “Quantita” ha come default “1”.

C. Tutte le chiavi esterne (foreign keys) hanno come politica di reazione “ON DELETE CASCADE”, in modo che - una volta eliminato il valore dalla tabella alla quale si riferiscono - vengano eliminati tutti i valori associati nelle varie tabelle in cui è presente quella chiave esterna.

# Implementazione

## Creazione degli utenti e permessi

Sulla base di quanto richiesto dal proprietario del ristorante, sono stati creati i seguenti utenti:

- “db_LoSfizio”, utilizzato per la creazione della base di dati;
- “Proprietario”, con tutti i permessi, in modo che possa modificare tutte le tabelle e avere accesso senza vincoli al DB: di fatto il proprietario utilizzerà sia le tabelle per la gestione dei clienti quando sarà alla cassa e sia quelle per la gestione dei dipendenti;

```
-- Creazione Utenti e Password --

create user db_LoSfizio identified by admin;

create user Proprietario identified by leader;

create user cassiere1 identified by acpi1;

create user cassiere2 identified by uefi;

create user cassiere3 identified by kexts;

create user cassiere4 identified by gpt;

grant all privileges to db_LoSfizio;

grant all privileges to Proprietario;


-- Creazione ruolo: cashier --

create role cashier;

grant all on TAVOLO to cashier;

grant all on ORDINE to cashier;

grant all on CONTO to cashier;

grant all on CONTIENE to cashier;

grant all on CLIENTEPRENOTANTE to cashier;


-- Utilizzo ruolo cashier --

grant cashier to cassiere1;

grant cashier to cassiere2;

grant cashier to cassiere3;

grant cashier to cassiere4;
```

- “cassiere”, da 1 a 4, dato che il ristorante avrà sempre quattro camerieri che, di volta in volta, potranno loggarsi. Si hanno quattro utenti diversi perché è necessario distinguere le azioni delle diverse persone alla cassa in caso di problemi.

Nella seconda parte del codice, viene definito “cashier” come role, in modo da poter dare i permessi ai quattro cassieri senza dover ripetere le 5 righe di codice per ognuno (usando 6+4 righe di codice invece che 20).

# Creazione delle tabelle

Le tabelle vengono create mediante istruzioni “create table”, proseguendo poi specificando le chiavi primarie e secondarie, vincoli statici “check”, le politiche di reazione e i valori di default, secondo l’organizzazione del ristorante, dall’utente db_LoSfizio.

La prima istruzione è utile al fine di stabilire il formato della data che sarà utilizzato durante il popolamento. La creazione è ottenibile dal file “*Creazione LoSfizioDB.sql”.*

```
create table TAVOLO (
NumTavolo          NUMBER(2,0)    PRIMARY KEY,
NumMaxPartecipanti     NUMBER(2,0)    NOT NULL,
Luogo        VARCHAR(15)    NOT NULL    check (Luogo in ('Esterno','Interno'))
);


create table ORDINE (
NumTavolo   NUMBER(2,0),
DataOraOrdine   DATE      DEFAULT SYSDATE,
Costo     NUMBER(3,0) DEFAULT 0,
CONSTRAINT FKOrdine foreign key (NumTavolo) references TAVOLO(NumTavolo) ON DELETE CASCADE,
CONSTRAINT PKOrdine primary key (DataOraOrdine, NumTavolo)
);


create table CONTO (
NumFattura    VARCHAR(15) PRIMARY KEY,
NumTavolo   NUMBER(2,0)     NOT NULL,
DataOraOrdine     DATE    NOT NULL,
MetodoPagamento   VARCHAR(15)   NOT NULL    check (MetodoPagamento in ('Contanti','Bancomat')),
CONSTRAINT FKConto1 foreign key (NumTavolo) references TAVOLO(NumTavolo) ON DELETE CASCADE,
CONSTRAINT FKConto2 foreign key (DataOraOrdine, NumTavolo) references ORDINE(DataOraOrdine, NumTavolo) ON DELETE CASCADE,
CONSTRAINT UniConto UNIQUE(NumTavolo,DataOraOrdine)
);


create table PRODOTTIINVENDITA (
IdentificativoProdotto  VARCHAR(15) PRIMARY KEY,
NomeProd    VARCHAR(35) NOT NULL,
CostoUnitario   NUMBER(3,0) NOT NULL,
TipologiaProd   VARCHAR(20)   NOT NULL  check (TipologiaProd in ('Antipasto', 'Primo', 'Secondo', 'Dolce', 'Vino Bianco', 'Vino Rosso', 'Bevanda')),
ABaseDi     VARCHAR(15)
);


create table CONTIENE (
NumTavolo   NUMBER(2,0),
DataOraOrdine   DATE,
IdentificativoProdotto  VARCHAR(15),
Quantita    NUMBER(3,0) DEFAULT 1,
CONSTRAINT FKContiene1 foreign key (NumTavolo) references TAVOLO(NumTavolo) ON DELETE CASCADE,
CONSTRAINT FKContiene2 foreign key (DataOraOrdine, NumTavolo) references ORDINE(DataOraOrdine, NumTavolo) ON DELETE CASCADE,
CONSTRAINT PKContiene primary key (NumTavolo, DataOraOrdine, IdentificativoProdotto)
);

create table CLIENTEPRENOTANTE (
Contatto    VARCHAR(30) PRIMARY KEY,
SceltaLuogo   VARCHAR(15) check (SceltaLuogo in ('Esterno', 'Interno')),
NumeroPartecipanti  NUMBER(2,0) NOT NULL,
NumTavoloAssegnato  NUMBER(2,0),
DataOraAssegnati  DATE,
CONSTRAINT FKCliente1 foreign key (NumTavoloAssegnato) references TAVOLO(NumTavolo) ON DELETE SET NULL,
CONSTRAINT FKCliente2 foreign key (DataOraAssegnati, NumTavoloAssegnato) references ORDINE(DataOraOrdine, NumTavolo) ON DELETE CASCADE
);


create table DIPENDENTE (
CartellinoDipendente  VARCHAR(15) PRIMARY KEY,
NomeDip     VARCHAR(15) NOT NULL,
CognomeDip    VARCHAR(15) NOT NULL,
ContattoDip   VARCHAR(30) NOT NULL,
DataAssunzione    DATE    NOT NULL,
DataNascita   DATE    NOT NULL,
Ruolo     VARCHAR(15)   NOT NULL  check (Ruolo in ('Cameriere','Cuoco'))
);
```

*Editor di testo a colori utilizzato: Atom for macOS, UI Theme: One Light / Syntax Theme: Base16 Tomorrow Light

```
create table TURNO (
CartellinoDipendente  VARCHAR(15),
DataOraTurno    DATE,
OreTurno    NUMBER(2,0) NOT NULL,
Timbrato    VARCHAR(4)  NOT NULL    check (Timbrato in ('Si','No')),
CONSTRAINT FKTurno1 foreign key (CartellinoDipendente) references DIPENDENTE(CartellinoDipendente) ON DELETE CASCADE,
CONSTRAINT PKTurno primary key (DataOraTurno, CartellinoDipendente)
);


create table SALA (
NumSala     NUMBER(1,0) PRIMARY KEY   check (NumSala BETWEEN 1 and 3),
NPrimoTav   NUMBER(2,0) NOT NULL,
NUltimoTav    NUMBER(2,0) NOT NULL,
Fumatori    VARCHAR(4)      check (Fumatori in ('Si','No'))
);


create table SERVE (
CartellinoDipendente  VARCHAR(15),
DataOraTurno    DATE,
NumSala     NUMBER(1,0),
CONSTRAINT FKServe1 foreign key (CartellinoDipendente) references DIPENDENTE(CartellinoDipendente) ON DELETE CASCADE,
CONSTRAINT FKServe2 foreign key (DataOraTurno, CartellinoDipendente) references TURNO(DataOraTurno, CartellinoDipendente) ON DELETE CASCADE,
CONSTRAINT FKServe3 foreign key (NumSala) references SALA(NumSala) ON DELETE CASCADE,
CONSTRAINT PKServe primary key (CartellinoDipendente, DataOraTurno, NumSala)
);
```

```
create table PERMESSO (
CartellinoDipendente  VARCHAR(15),
DataOraPermesso   DATE,
Motivazione     VARCHAR(20) NOT NULL  check (Motivazione in ('Salute','Personale')),
CONSTRAINT FKPermesso1 foreign key (CartellinoDipendente) references DIPENDENTE(CartellinoDipendente) ON DELETE CASCADE,
CONSTRAINT PKPermesso primary key (CartellinoDipendente, DataOraPermesso)
);


create table QUALIFICA (
CodiceAttestato   VARCHAR(15) PRIMARY KEY,
CartellinoDipendente  VARCHAR(15),
TitoloAttestato   VARCHAR(40) NOT NULL,
CONSTRAINT FKQualifica1 foreign key (CartellinoDipendente) references DIPENDENTE(CartellinoDipendente) ON DELETE CASCADE
);
```

Successivamente, si può procedere al popolamento tramite il file “*Popolamento LoSfizioDB.sql*”.
A questo punto, è già possibile utilizzare il DB per eseguire qualsiasi delle operazioni base descritte a pagina 9.

*Editor di testo a colori utilizzato: Atom for macOS, UI Theme: One Light / Syntax Theme: Base16 Tomorrow Light

# Viste

Le viste ci permettono un accesso più rapido per la visualizzazione dei dati.
Sono state implementate le seguenti viste:

- *conti_nonpagati*

Questa vista molto semplice consente la visualizzazione di quali Ordini non sono ancora stati pagati dai clienti.
Viene implementata per comodità, essendo un'azione che può essere richiesta molto spesso.

```
CREATE OR REPLACE VIEW conti_nonpagati AS
SELECT ORDINE.DataOraOrdine, ORDINE.NumTavolo
FROM ORDINE
WHERE not exists ( SELECT Conto.NumTavolo, CONTO.DataOraOrdine
        FROM CONTO);
```

- *dipendenti_nonautorizzati*

La vista consente di visionare quali dipendenti non hanno seguito il loro turno, senza richiedere un permesso, il giorno lasciato scoperto e di quale ruolo avrebbero dovuto occuparsi.

```
CREATE OR REPLACE VIEW dipendenti_nonautorizzati AS
SELECT DIPENDENTE.CartellinoDipendente, DIPENDENTE.Ruolo, TURNO.DataOraTurno
FROM (DIPENDENTE join TURNO on DIPENDENTE.CartellinoDipendente=TURNO.CartellinoDipendente)
WHERE TURNO.Timbrato = 'No'
AND
not exists ( SELECT PERMESSO.CartellinoDipendente, PERMESSO.DataOraPermesso
       FROM PERMESSO
             WHERE TURNO.CartellinoDipendente = PERMESSO.CartellinoDipendente
                    AND
                   PERMESSO.DataOraPermesso = TURNO.DataOraTurno);
```

- *npermessi_dipendente*

La vista riguarda la visualizzazione di quanti permessi ha richiesto ogni dipendente. Potrebbe essere utile al proprietario per guardare quanti permessi ha richiesto un dipendente prima di concederne un altro.

```
CREATE OR REPLACE VIEW npermessi_dipendente AS
SELECT    DIPENDENTE.CartellinoDipendente, COUNT(PERMESSO.CartellinoDipendente) AS NPermessiRichiesti
FROM    DIPENDENTE join PERMESSO on DIPENDENTE.CartellinoDipendente=PERMESSO.CartellinoDipendente
GROUP BY  DIPENDENTE.CartellinoDipendente;
```

- *ntavolipersala*

La vista permette di vedere quanti tavoli sono presenti in ogni sala (considerato che vi sono tre sale). Continua ad essere valida anche nel caso in cui si cambino i valori di “NPrimoTav” e “NUltimoTav” su “Sala”. Utile per gestire il carico del lavoro sui camerieri e in caso di spostamento di gruppi ordinati di tavoli.

```
CREATE OR REPLACE VIEW ntavolipersala AS
SELECT SALA.NumSala AS SalaN, SALA.NPrimoTav AS PrimoTavolo, SALA.NUltimoTav AS UltimoTavolo, SUM(SALA.NUltimoTav - SALA.NPrimoTav + 1) AS NTavoliNellaSala
FROM SALA
WHERE
     (
  ( SALA.NPrimoTav = (SELECT SALA.NPrimoTav FROM SALA WHERE SALA.NumSala=1) )
  OR
  ( SALA.NPrimoTav = (SELECT SALA.NPrimoTav FROM SALA WHERE SALA.NumSala=2) )
  OR
  ( SALA.NPrimoTav = (SELECT SALA.NPrimoTav FROM SALA WHERE SALA.NumSala=3) )
     )
GROUP BY NumSala, NPrimoTav, NUltimoTav
ORDER BY NumSala;
```

- *tavoli_occupati*

Visualizzazione di quali tavoli sono correntemente occupati, utile quando bisogna associare un tavolo ad un cliente.

```
CREATE OR REPLACE VIEW tavoli_occupati AS
SELECT ORDINE.NumTavolo, ORDINE.DataOraOrdine
FROM ORDINE
WHERE ( ORDINE.DataOraOrdine < (SYSDATE)
AND
 ( (TRUNC(ORDINE.DataOraOrdine)) = (TRUNC (SYSDATE) ) ) )
AND
 (ORDINE.NumTavolo != 0)
AND
ORDINE.DataOraOrdine != ALL (SELECT CONTO.DataOraOrdine FROM CONTO)
ORDER BY ORDINE.DataOraOrdine;
```

- *tavoli_pagati*

Vista tavoli dei quali si è provveduto al pagamento.

```
CREATE OR REPLACE VIEW tavoli_pagati AS
SELECT  ORDINE.NumTavolo, ORDINE.DataOraOrdine, CONTO.NumFattura
FROM  (ORDINE join CONTO on ORDINE.DataOraOrdine = CONTO.DataOraOrdine AND ORDINE.NumTavolo=CONTO.NumTavolo)
WHERE exists (SELECT CONTO.NumFattura FROM CONTO
              WHERE ORDINE.NumTavolo=CONTO.NumTavolo)
AND ORDINE.NumTavolo != 0;
```

- *tavoli_prenotati*

Visualizzazione dei tavoli prenotati e in quale data. Utile da consultare prima di prenotare un tavolo ad un nuovo cliente.

```
CREATE OR REPLACE VIEW tavoli_prenotati AS
SELECT ORDINE.NumTavolo, ORDINE.DataOraOrdine
FROM ( ORDINE join CLIENTEPRENOTANTE on ORDINE.DataOraOrdine = CLIENTEPRENOTANTE.DataOraAssegnati )
WHERE exists (SELECT CLIENTEPRENOTANTE.NumTavoloAssegnato, CLIENTEPRENOTANTE.DataOraAssegnati
        FROM CLIENTEPRENOTANTE
        WHERE ORDINE.DataOraOrdine = CLIENTEPRENOTANTE.DataOraAssegnati)
ORDER BY ORDINE.DataOraOrdine;
```

- *eta_dipendenti*

Visualizzazione dell'età (arrotondata) di ogni dipendente.

```
CREATE OR REPLACE VIEW eta_dipendenti AS
SELECT  DIPENDENTE.CartellinoDipendente, DIPENDENTE.NomeDip, DIPENDENTE.CognomeDip,
  ROUND(SUM(MONTHS_BETWEEN (SYSDATE, DIPENDENTE.DataNascita)/12), 0) AS EtaDip
FROM DIPENDENTE
GROUP BY DIPENDENTE.CartellinoDipendente, DIPENDENTE.NomeDip, DIPENDENTE.CognomeDip;
```

# Note

## Sulle Totalità e Molteplicità

Appunti di progettazione, contenuti anche nel file “*Sulle Totalità e Molteplicità.txt*”

**-- TOTALITA' --**

+ Posso inserire un ORDINE senza NumTavolo?

NO! -> "Tutti gli Ordini sono seduti a Tavoli" rispettata.

+ Posso inserire un CONTO senza correlazione ad un ORDINE?

NO! -> "Tutti i Conti sono pagati da Ordini" rispettata.

+ Posso inserire valori su CONTIENE senza correlazione ad un ORDINE?

NO! -> Tutti i valori di Contiene devono riferirsi ad un Ordine valido.

+ Posso inserire un ORDINE che non Contiene alcun prodotto?

SI. -> "Tutti gli Ordini contengono Prodotti In Vendita" non rispettata.

Lasciata in questo modo perché si potrebbe voler ordinare successivamente.

+ Posso inserire un CLIENTE PRENOTANTE senza relativo ORDINE?

SI. -> "Tutti i Clienti Prenotanti effettuano un ordine" non rispettata.

+ Posso inserire una QUALIFICA senza DIPENDENTE che la possiede?

NO! -> "Tutte le Qualifiche sono possedute da Dipendenti" rispettata.

+ Posso inserire un DIPENDENTE che non sia né "Cuoco" né "Cameriere"?

NO! -> "Tutti i Dipendenti si specializzano in Cuoco o Cameriere" rispettata.

Ma posso inserire un "Cameriere" con la specializzazione che dovrebbe essere propria “Cuoco".

=> Specializzazione non tradotta.

+ Posso inserire un TURNO senza che si riferisca ad un DIPENDENTE?

NO! -> "Tutti i Turni sono seguiti da Dipendenti" rispettata.

+ Posso inserire un PERMESSO senza che sia richiesto da un DIPENDENTE?

NO! -> "Tutti i Permessi sono richiesti da Dipendenti" rispettata.

+ Posso inserire una SALA senza che sia servita da alcun dipendente?

Teoricamente "SI", ma c'è un check su SALA "NumSala BETWEEN 1 and 3" e SERVE.NumSala è parte della PK di SERVE.

**-- MOLTEPLICITA' --**

+ "Ad un Tavolo siedono N Ordini", "Un Ordine siede ad un solo Tavolo"
rispettata.

+ "Un Ordine contiene N ProdottiInVendita", "Un ProdottoInVendita è contenuto in M Ordini"
rispettata.

+ "Un Ordine paga un solo Conto", "Un Conto è pagato da un solo Ordine"
non rispettata, posso inserire NumFattura diversi per lo stesso Ordine.
=> Posso impedirlo mettendo UNIQUE la combinazione (NumTavolo,DataOraOrdine) su CONTO. =>
rispettata.

+ "Un Cliente Prenotante può effettuare un solo Ordine", "Un Ordine è effettuato da un solo CP"
rispettata, dato che "contatto" è chiave primaria di ClientePrenotante.

+ "Un Dipendente segue N Turni", "Un Turno è seguito da un solo Dipendente"
rispettata.

+ "Un Dipendente possiede M Qualifiche", "Una Qualifica è posseduta da un solo Dipendente"
rispettata.

+ "Un Turno serve una Sala", "Una Sala è servita da N Turni"
rispettata, ma non la prima parte dato che posso avere stesso Turno ma con Sale diverse.

## Sui Constraint

Appunti relativi ai vincoli implementati, contenuti anche nel file "*Sui Constraint.txt*".

**-- ORDINE --**

FKOrdine -> Vincolo di chiave esterna che lega "Ordine" a "Tavolo" tramite "NumTavolo".

ON DELETE: Cascade.

(Eliminando il NumTavolo, saranno eliminati in automatico tutti gli ordini che riguardano quel tavolo -> Se il tavolo non esiste più, non esisteranno più neanche gli ordini associati)

PKOrdine -> Vincolo di chiave primaria di "Ordine", dato che la primary key di "Ordine" è formata da (NumTavolo, DataOraOrdine)
[Ordine è un'entità debole di Tavolo]

**-- CONTO --**

FKConto1 -> Vincolo di chiave esterna che lega "Conto" a "Tavolo".

ON DELETE: Cascade.

(Eliminando il NumTavolo da Tavolo, saranno eliminati tutti i Conti ad esso associati)

FKConto2 -> Vincolo di chiave esterna che lega "Conto" ad "Ordine".

ON DELETE: Cascade.

(Eliminando una DataOraOrdine e un NumTavolo da "Ordine", avrò l'eliminazione della relativa fattura su "Conto")

UniConto -> Vincolo UNIQUE che rende unica la coppia (NumTavolo, DataOraOrdine).

Impedisce di avere più conti per una stessa istanza di "Ordine".

**-- CONTIENE --**

FKContiene1 -> Vincolo di chiave esterna che lega "Contiene" a "Tavolo".

ON DELETE: Cascade.

(Eliminando il NumTavolo da Tavolo, vengono eliminate le tuple su "Contiene" dove appare quel Tavolo)

FKContiene2 -> Vincolo di chiave esterna che lega "Contiene" ad "Ordine".

ON DELETE: Cascade.

(Eliminando una DataOraOrdine e un NumTavolo da "Ordine", avrò l'eliminazione di tutti i prodotti contenuti nell'Ordine)

PKContiene -> Vincolo di chiave primaria dell'associazione "Contiene".

Impedisce la ripetizione di (NumTavolo, DataOraOrdine, IdentificativoProdotto).

(Infatti, se dovessimo ordinare un prodotto più volte per uno stesso Ordine, c'è l'attributo "Quantita" su “Contiene").

**-- CLIENTEPRENOTANTE --**

FKCliente1 -> Vincolo di chiave esterna che lega "ClientePrenotante" a "Tavolo".

ON DELETE: Set NULL.

(Eliminando il NumTavolo da Tavolo, alle prenotazioni su quel tavolo vengono assegnati valori di "NumTavoloAssegnato" null, in modo che siano successivamente riallocate ad un tavolo esistente)

FKCliente2 -> Vincolo di chiave esterna che lega "ClientePrenotante" a "Ordine"

ON DELETE: Cascade.

(Eliminando una DataOraOrdine e un NumTavolo da "Ordine", che corrispondono ad un Ordine di un ClientePrenotante, anche la sua prenotazione verrà eliminata)

**-- TURNO --**

FKTurno1 -> Vincolo di chiave esterna che lega "Turno" a "Dipendente".

ON DELETE: Cascade.

(Eliminando un CartellinoDipendente, eliminerò anche tutti i turni ad esso associati)

PKTurno -> Vincolo di chiave primaria di "Turno".

"Turno" risulta essere un'entità debole di "Dipendente".

E' definita da (CartellinoDipendente, DataOraTurno).

**-- SERVE --**

FKServe1 -> Vincolo di chiave esterna che lega "Serve" a "Dipendente".

ON DELETE: Cascade.

(Eliminato il CartellinoDipendente, non è più utile sapere le sale che serve)

FKServe2 -> Vincolo di chiave esterna che lega "Serve" a "Turno".

ON DELETE: Cascade.

(Eliminato il relativo Turno, il Turno non serve più la Sala)

FKServe3 -> Vincolo di chiave esterna che lega "Serve" a "Sala".

ON DELETE: Cascade.

(Eliminata la Sala, non è più servita da alcun Turno)

PKServe -> Vincolo di chiave primaria dell'associazione "Serve".

Impedisce che venga ripetuta la combinazione di (CartellinoDipendente, DataOraTurno, NumSala).

Impedisce che vi siano dei valori su "Serve" con null su uno di questi tre attributi.

**-- PERMESSO --**

FKPermesso1 -> Vincolo di chiave esterna che collega "Permesso" a "Dipendente".

ON DELETE: Cascade.

(Eliminando un Dipendente, non ci interessano più i permessi che ha preso)

PKPermesso -> Vincolo di chiave primaria di "Permesso".

"Permesso" è entità debole di "Dipendente".

E' definita da (CartellinoDipendente, DataOraPermesso).

**-- QUALIFICA --**

FKQualifica1 -> Vincolo di chiave esterna che collega "Qualifica" a "Dipendente".

(Eliminando un Dipendente, non ci interessano più le qualifiche che aveva)