PEGASO
Università Telematica

TEMPLATE
PROJECT WORK

Corso di Studio	INFORMATICA PER LE AZIENDE DIGITALI (L-31)
Dimensione dell'elaborato	Minimo 6.000 – Massimo 10.000 parole *(pari a circa Minimo 12 – Massimo 20 pagine)*
Formato del file da caricare in piattaforma	PDF
Nome e Cognome	**Maddalena Pagliarulo**
Numero di matricola	**0312201391**
Tema n. (Indicare il numero del tema scelto):	Tema n. 1
Titolo del tema (Indicare il titolo del tema scelto):	La digitalizzazione dell'impresa
Traccia del PW n. (Indicare il numero della traccia scelta):	4
Titolo della traccia (Indicare il titolo della traccia scelta):	Progettazione dello schema di persistenza dei dati a supporto dei servizi di un'azienda nel settore dei trasporti
Titolo dell'elaborato (Attribuire un titolo al proprio elaborato progettuale):	**VIAGGIARE IN TRENO**

PARTE PRIMA – DESCRIZIONE DEL PROCESSO

Utilizzo delle conoscenze e abilità derivate dal percorso di studio
(Descrivere quali conoscenze e abilità apprese durante il percorso di studio sono state utilizzate per la redazione dell'elaborato, facendo eventualmente riferimento agli insegnamenti che hanno contribuito a maturarle):

Conoscenze

- Visione di insieme delle risorse di un sistema di elaborazione
- concetti e modelli per l'organizzazione di una base di dati.
- Modellazione dei dati,
- il modello E/R,
- entità, attributi,associazioni,
- associazione ricorsiva.
- Concetti di base del modello relazionale,
- derivazione del modello logico dal modello concettuale,
- operazioni relazionali,
- normalizzazione, integrità dei dati.
- Caratteristiche generali di un linguaggio per basi di dati,
- parole chiave e sintassi del linguaggio SQL,
- codifica delle operazioni relazionali.

Abilità

- Individuare le caratteristiche di un sistema di gestione di basi di dati.
- Individuare le entità e gli attributi della realtà osservata,

- classificare le associazioni tra entità,
- disegnare il modello E/R di un problema,
- verificare la correttezza del modello attraverso le regole di lettura,
- sviluppare i passi dell'analisi di un problema, rappresentare nel modello le associazioni ricorsive.
- Usare le regole di derivazione delle tabelle dal modello E/R,
- applicare le operazioni relazionali per interrogare un database,
- normalizzare le relazioni,impostare i controlli per l'integrità dei dati.
- Applicare i principi del modello relazionale,
- utilizzare i comandi del linguaggio SQL per la definizione delle tabelle, le operazioni di manipolazione dei dati e le interrogazioni.
- Utilizzare funzioni e clausole per calcoli, raggruppamenti, ordinamenti e ricerche avanzate.
- Codificare le viste.

Riferimento agli insegnamenti che hanno contribuito a maturarle:

Corso - BASI DI DATI. Prof.re - Filippo Sciarrone, nonché al mio lavoro come ingegnante tecnico pratico (classe di concorso (B016) presso un'istituto tecnico tecnologico.

Fasi di lavoro e relativi tempi di implementazione per la predisposizione dell'elaborato
(Descrivere le attività svolte in corrispondenza di ciascuna fase di redazione dell'elaborato. Indicare il tempo dedicato alla realizzazione di ciascuna fase, le difficoltà incontrate e come sono state superate):

Fasi di lavoro e tempi di implementazione

Analisi dei requisiti

Durata stimata: 1 settimana
In questa fase si è svolta la raccolta e l'analisi delle esigenze funzionali e non funzionali dell'azienda ferroviaria. Sono stati identificati i processi chiave, gli attori coinvolti (passeggeri, personale, sistema informativo) e le principali funzionalità da supportare (prenotazione, acquisto, validazione biglietti, gestione tratte).

Progettazione concettuale

Durata stimata: 2 giorni
È stato realizzato il modello Entità-Relazione (E/R) per rappresentare le entità principali (Passeggero, Biglietto, Tratta, Stazione, Mezzo) e le associazioni tra di esse. Si è posta particolare attenzione alla corretta rappresentazione delle associazioni e delle eventuali relazioni ricorsive.

Progettazione logica e normalizzazione

Durata stimata: 2 giorni
Il modello concettuale è stato trasformato in modello relazionale, definendo tabelle, chiavi primarie e chiavi esterne. Si è applicata la normalizzazione fino alla terza forma normale (3NF) per assicurare l'assenza di ridondanze e anomalie.

Implementazione fisica

Durata stimata: 1 settimana
È stata realizzata la creazione delle tabelle in Xampp/phpMyAdmin, la definizione di vincoli di integrità, la predisposizione degli indici per ottimizzare le query più frequenti. È stato creato uno script SQL per il popolamento iniziale del database.

Sviluppo delle query e testing

Durata stimata: 2 giorni
Sono state sviluppate e testate le query SQL più rilevanti per il sistema: ricerca tratte, storico prenotazioni, verifica validità biglietto, elenco passeggeri e report vendite. È stata effettuata una fase di collaudo per verificarne la correttezza e le prestazioni.

Documentazione e consegna

Durata stimata: 1 settimana
Preparazione della documentazione tecnica, inclusi diagrammi ER, descrizione del modello relazionale e script SQL. Infine, consegna del progetto tramite repository GitHub.

Totale durata stimata: 4 settimane

Risorse e strumenti impiegati
(Descrivere quali risorse - bibliografia, banche dati, ecc. - e strumenti - software, modelli teorici, ecc. - sono stati individuati ed utilizzati per la redazione dell'elaborato. Descrivere, inoltre, i motivi che hanno orientato la scelta delle risorse e degli strumenti, la modalità di individuazione e reperimento delle risorse e degli strumenti, le eventuali difficoltà affrontate nell'individuazione e nell'utilizzo di risorse e strumenti ed il modo in cui sono state superate):

Per la realizzazione del progetto sono state utilizzate diverse risorse, scelte principalmente per facilitare lo sviluppo, il testing e la prototipazione del sistema di persistenza dati in un ambiente locale e accessibile.

Strumenti software:

- XAMPP la piattaforma che ci permette di avviare un server locale sul nostro computer è stata utilizzata per la creazione, popolamento e interrogazione del database.
- phpMyAdmin che ci permette di interfacciarci con un database MySQL senza doverlo fare attraverso la linea di comando. Grazie ad una comoda interfaccia phpMyAdmin ci permette, infatti, di gestire il nostro database MySQL in maniera davvero semplice.
- **Strumenti di modellazione ER:** software come **draw.io** utilizzato per la realizzazione del modello Entità-Relazione in formato grafico.
- **Git e GitHub:** utilizzati per la condivisione del repository.

Risorse Hardware

- **Ambiente di sviluppo locale**: configurazione standard con CPU multi-core e RAM sufficiente per eseguire XAMPP (Apache, MySQL/MariaDB) e per effettuare test e simulazioni. Questa configurazione è adeguata per un prototipo di medie dimensioni.

Documentazione e Fonti

- **Documentazione ufficiale di MySQL/MariaDB e XAMPP**: consultata per approfondire funzionalità, configurazione e ottimizzazione del database e dell'ambiente server.
- DATABASE SQL & PHP per il quinto anno degli istituti tecnici tecnologici.
- Camagni Paolo; Nikolassy Riccardo - Editore: HOEPLI

PARTE SECONDA – PREDISPOSIZIONE DELL'ELABORATO

Obiettivi del progetto
(Descrivere gli obiettivi raggiunti dall'elaborato, indicando in che modo esso risponde a quanto richiesto dalla traccia):

Il presente lavoro ha l'obiettivo di progettare un sistema di persistenza dei dati a supporto del processo di vendita dei biglietti per un'azienda operante nel settore ferroviario. Attraverso un approccio metodologico rigoroso, si intende costruire un modello dati ottimizzato, normalizzato e scalabile, in grado di garantire integrità, coerenza e affidabilità delle informazioni nel tempo.

Per strutturare i dati in modo coerente e comprensibile, ho adottato un modello Entità/Relazioni (E/R), che riflette le principali informazioni gestite dal sistema, come passeggeri, tratte, stazioni, biglietti e mezzi utilizzati.

L'interazione tra queste entità consente di coprire le principali funzionalità operative del processo, tra cui:

- la gestione delle tratte ferroviarie e degli orari;
- la prenotazione e l'acquisto dei biglietti da parte dei passeggeri;
- la registrazione dell'utilizzo e dell'annullamento dei titoli di viaggio;
- la tracciabilità dei mezzi coinvolti su ciascuna tratta;

Il modello, una volta tradotto in struttura relazionale, è stato implementato tramite script SQL che consentono la creazione delle tabelle, il popolamento iniziale dei dati e l'eventuale estensione futura con nuove funzionalità (es. scontistiche, fasce orarie, abbonamenti).

L'architettura proposta si presta a essere modularmente estendibile ed è pensata per garantire buone performance anche in scenari ad alta affluenza, con potenziale integrazione futura in sistemi real-time o web-based.

Contestualizzazione
(Descrivere il contesto teorico e quello applicativo dell'elaborato realizzato):

Il trasporto ferroviario rappresenta una scelta concreta per promuovere la mobilità sostenibile, grazie al suo ridotto impatto ambientale e alla buona efficienza energetica rispetto ad altri mezzi di trasporto. Esso assicura servizi su lunga, media e breve percorrenza, soddisfacendo un ampio spettro di esigenze di spostamento, sia in ambito urbano che extraurbano.

In un contesto in continua evoluzione, le aziende ferroviarie moderne si trovano ad affrontare sfide complesse legate alla digitalizzazione e all'automazione dei processi interni. Tali sfide comprendono la necessità di migliorare l'efficienza operativa, ottimizzare l'uso delle risorse disponibili e garantire una gestione efficace delle informazioni, assicurando nel contempo alti standard di qualità del servizio e tracciabilità dei dati.

Tra i processi centrali in questo ambito, la vendita e la gestione dei biglietti rappresentano un nodo cruciale per il funzionamento dell'intero sistema. La corretta gestione di queste attività influisce direttamente su aspetti organizzativi come il monitoraggio delle tratte, la pianificazione delle corse e il controllo dei flussi di passeggeri. Tali esigenze richiedono l'adozione di sistemi informativi solidi, strutturati e ben progettati, capaci di garantire la coerenza, la disponibilità e la sicurezza dei dati.

In quest'ottica, risulta fondamentale disporre di architetture dati flessibili e scalabili, supportate da modelli concettuali in grado di rappresentare fedelmente la realtà operativa. Il ricorso a un database ben progettato consente non solo di gestire in modo efficiente le informazioni relative ai biglietti, ai passeggeri, ai mezzi e alle tratte, ma anche di fornire un supporto concreto alle decisioni e alla crescita futura del servizio.

Descrizione dei principali aspetti progettuali
(Sviluppare l'elaborato richiesto dalla traccia prescelta):

Nel problema ho individuato le seguenti entità , attributi e associazioni:

- **Passeggero** per l'anagrafica del cliente con i seguenti attributi (Id,nome, cognome ed email);
- **Biglietto** per rappresentare l'acquisto effettuato da un passeggero per una tratta specifica con i seguenti attributi(Id, passeggero_id, tratta_id, data_acquisto e stato);
- **Mezzo** treno utilizzato per la tratta con gli attributi (id, tipo, codice_identificativo);
- **Stazione** per nodo di partenza o arrivo con gli attributi(id, nome, città);
- **Tratta** identifica un collegamento tra due stazioni con un orario e un mezzo, con gli attributi (id, stazione_partenza, stazione_arrivo, orario_partenza, orario_arrivo, mezzo_id);

Tra l'entità **Passeggero** e l'entità **Biglietto** esiste un'associazione uno a molti perché un passeggero può acquistare uno o più biglietti, ma ogni biglietto deve essere inteso come unico per le sue caratteristiche riferito ad un preciso passeggero.
Tra l'entità **Biglietto** e l'entità **Tratta** si stabilisce un'associazione uno a molti in quanto un biglietto può riguardare una o più tratte.
Tra l'entità **Mezzo** e l'entità **Tratta** esiste un'associazione uno a molti in quanto un mezzo può servire una o più tratte.
Tra l'entità **Tratta** e l'entità **Stazione** esiste un'associazione uno a uno, in quanto una specifica tratta è associata ad una stazione di partenza ed a una stazione di arrivo; nell'entità Stazione sono presenti due attributi nome e città, poiché in una città possono esserci più stazioni.

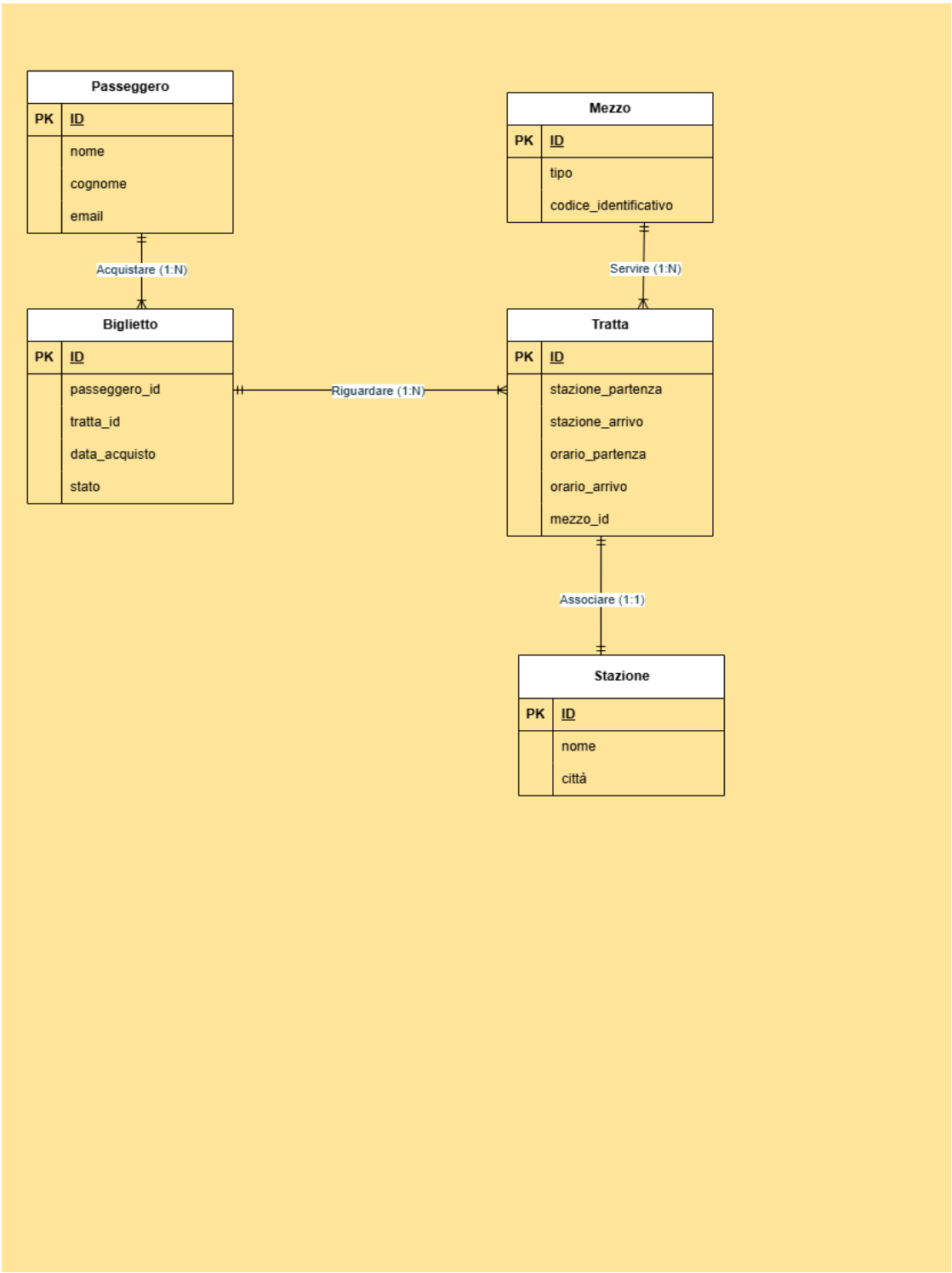
Glossario termini

Termine	Descrizione	Collegamenti
Passeggero	Id, nome, cognome, email	Biglietto
Biglietto	Id, passeggero_id, tratta_id, data_acquisto, stato	Passeggero Tratta
Mezzo	Id, tipo, codice_identificativo	Tratta
Tratta	Id, stazione_partenza, stazione_arrivo, orario_arrivo, orario_partenza, mezzo_id	Mezzo Stazione
Stazione	Id, nome, città	Tratta

Tabella relazioni/associazioni

E1 → E2	Molteplicità Min	Molteplicità Max	Nome
Passeggero - Biglietto	1	N	Acquistare
Mezzo - Tratta	1	N	Servire
Biglietto - Tratta	1	N	Riguardare
Tratta - Stazione	1	1	Associare

Modello concettuale (E/R diagramma)



Modello relazionale

Le entità vengono mappate come segue:
- Passeggero(id, nome, cognome, email);
- Stazione(id, nome, città);
- Mezzo(id, tipo, codice_identificativo);
- Tratta(id, stazione_partenza, stazione_arrivo, orario_partenza, orario_arrivo, mezzo_id);
- Biglietto(id, passeggero_id, tratta_id, data_acquisto, stato);

Chiavi primarie :ID
Chiavi esterne:passeggero_id e tratta_id in Biglietto.

Normalizzazione e progettazione efficiente dei dati

Durante la progettazione di un database relazionale, è fondamentale evitare che i dati siano ridondanti o inconsistenti. Per ottenere questo risultato, si applica un processo chiamato normalizzazione, che consiste nel suddividere le tabelle in modo logico e ordinato.
L'obiettivo è quello di garantire:

- che i dati non vengano duplicati inutilmente,
- che le modifiche siano facili da gestire senza errori,
- che ogni informazione sia collocata nel posto corretto, in base alla sua funzione.

Naturalmente, nel fare questo bisogna assicurarsi che il significato originario dei dati non venga perso e che le relazioni tra le informazioni rimangano integre e tracciabili.

Prima Forma Normale (1NF)

Per soddisfare la 1NF, ogni riga della tabella deve descrivere un solo oggetto, con valori semplici. In particolare:

- tutte le righe devono avere lo stesso numero di colonne,
- ogni colonna deve contenere un solo valore (nessun dato aggregato),
- ogni riga deve essere unica (grazie alla chiave primaria).

Seconda Forma Normale (2NF)

Una tabella è in 2NF se è in 1NF e se tutti gli attributi non chiave dipendono dall'intera chiave primaria, non da una sola parte. Questo vale soprattutto se la chiave è composta da più campi.

Terza Forma Normale (3NF)

In 3NF, tutti gli attributi non chiave devono dipendere direttamente dalla chiave primaria, evitando dipendenze indirette o transitive.
Nel mio progetto, ho costruito il modello relazionale rispettando tutte le tre forme normali, per garantire coerenza, efficienza e flessibilità nel trattamento dei dati.

Implementazione fisica

```
CREATE TABLE Passeggero (
   ID INT AUTO_INCREMENT PRIMARY KEY,
   nome VARCHAR(50) NOT NULL,
   cognome VARCHAR(50) NOT NULL,
   email VARCHAR(100) UNIQUE
);

CREATE TABLE Biglietto (
   ID INT AUTO_INCREMENT PRIMARY KEY,
   passeggero_id INT NOT NULL,
   tratta_id INT NOT NULL,
   data_acquisto DATE NOT NULL,
   stato VARCHAR(20),
   FOREIGN KEY (passeggero_id) REFERENCES Passeggero(ID),
   FOREIGN KEY (tratta_id) REFERENCES Tratta(ID)
);

CREATE TABLE Tratta (
   ID INT AUTO_INCREMENT PRIMARY KEY,
   stazione_partenza INT NOT NULL,
   stazione_arrivo INT NOT NULL,
   orario_partenza DATETIME NOT NULL,
   orario_arrivo DATETIME NOT NULL,
   mezzo_id INT NOT NULL,
   FOREIGN KEY (stazione_partenza) REFERENCES Stazione(ID),
   FOREIGN KEY (stazione_arrivo) REFERENCES Stazione(ID),
   FOREIGN KEY (mezzo_id) REFERENCES Mezzo(ID)
);

CREATE TABLE Stazione (
   ID INT AUTO_INCREMENT PRIMARY KEY,
   nome VARCHAR(100) NOT NULL,
   città VARCHAR(100) NOT NULL
);

CREATE TABLE Mezzo (
   ID INT AUTO_INCREMENT PRIMARY KEY,
   tipo VARCHAR(50) NOT NULL CHECK (tipo = 'Treno'),
   codice_identificativo VARCHAR(100) UNIQUE NOT NULL
);

CREATE INDEX idx_tratta_partenza ON Tratta(stazione_partenza);
CREATE INDEX idx_tratta_arrivo ON Tratta(stazione_arrivo);
CREATE INDEX idx_biglietto_passeggero ON Biglietto(passeggero_id);
CREATE INDEX idx_biglietto_tratta ON Biglietto(tratta_id);
```

Popolamento data base:

```
INSERT INTO Passeggero (nome, cognome, email) VALUES
('Maddalena', 'Pagliarulo', 'maddalena.pagliarulo@gmail.com),
('Maria', 'Cipriano', 'maria.cipriano@gmail.com),
('Gerardo', 'Pizzulo', 'gerardo.pizzulo@gmail.com),
('Rocco', 'Russo', 'rocco.russo@gmail.com),
('Ilaria', 'Saporito', 'ilaria.saporito@gmail.com.com');


INSERT INTO Biglietto (passeggero_id, tratta_id, data_acquisto, stato) VALUES
(1, 1, '2025-07-01', 'confermato'),
(2, 2, '2025-07-02', 'in attesa'),
(3, 3, '2025-07-03', 'confermato'),
(4, 4, '2025-07-04', 'annullato'),
(5, 5, '2025-07-05', 'confermato');

INSERT INTO Tratta (stazione_partenza_id, stazione_arrivo_id, orario_partenza, orario_arrivo,
mezzo_id) VALUES
(1, 2, '2025-06-10 08:00:00', '2025-06-10 12:30:00', 1),
(2, 3, '2025-06-15 08:00:00', '2025-06-15 13:30:00', 2),
(3, 4, '2025-06-25 10:00:00', '2025-06-25 14:30:00', 3),
(4, 5, '2025-07-04 08:00:00', '2025-07-04 12:30:00', 4),
(5, 1, '2025-07-10 12:00:00', '2025-07-10 16:30:00', 5);

INSERT INTO Stazione (nome, citta) VALUES
 ('Stazione Santa Lucia', 'Venezia'),
('Stazione Centrale', 'Napoli'),
('Stazione Porta Nuova', 'Torino'),
('Stazione Centrale', 'Milano'),
('Stazione Termini', 'Roma');

INSERT INTO Mezzo (tipo, codice_identificativo) VALUES
('Treno', 'TR101'),
('Treno', 'TR123'),
('Treno', 'TR783'),
('Treno', 'TR040'),
('Treno', 'TR303');
```

Interrogazioni/Query SQL

1. **Ricerca tratte disponibili:**

Visualizzo tutte le tratte con dettagli stazioni, orari e mezzo

```
SELECT
    tratta.ID AS id_tratta,
    stazione_partenza.nome AS stazione_di_partenza,
    stazione_partenza.città AS città_di_partenza,
    stazione_arrivo.nome AS stazione_di_arrivo,
    stazione_arrivo.città AS città_di_arrivo,
    tratta.orario_partenza,
    tratta.orario_arrivo,
    mezzo.tipo AS tipo_mezzo,
    mezzo.codice_identificativo AS codice_mezzo
FROM
    Tratta
JOIN
    Stazione AS stazione_partenza ON Tratta.stazione_partenza = stazione_partenza.ID
JOIN
    Stazione AS stazione_arrivo ON Tratta.stazione_arrivo = stazione_arrivo.ID
JOIN
    Mezzo ON Tratta.mezzo_id = Mezzo.ID;
```

risultato dopo esecuzione:

ID_tratta	stazione_di_partenza	città_di_partenza	stazione_di_arrivo	città_di_arrivo	orario_partenza	orario_arrivo	tipo_mezzo	codice_mezzo
1	Stazione Santa Lucia	Venezia	Stazione Centrale	Napoli	2025-06-10 08:00:00	2025-06-10 12:30:00	Treno	TR101
2	Stazione Centrale	Napoli	Stazione Porta Nuova	Torino	2025-06-15 08:00:00	2025-06-15 13:30:00	Treno	TR123
3	Stazione Porta Nuova	Torino	Stazione Centrale	Milano	2025-06-25 10:00:00	2025-06-25 14:30:00	Treno	TR783
4	Stazione Centrale	Milano	Stazione Termini	Roma	2025-07-04 08:00:00	2025-07-04 12:30:00	Treno	TR040
5	Stazione Termini	Roma	Stazione Santa Lucia	Venezia	2025-07-10 12:00:00	2025-07-10 16:30:00	Treno	TR303

2. **Storico prenotazioni di un passeggero:**

Mostro tutti i biglietti di un passeggero specifico (esempio: ID passeggero = 1), con dettagli della tratta e stato.

```
SELECT
    biglietto.ID AS id_biglietto,
    tratta.ID AS id_tratta,
    stazione_partenza.nome AS stazione_di_partenza,
    stazione_partenza.città AS città_di_partenza,
    stazione_arrivo.nome AS stazione_di_arrivo,
    stazione_arrivo.città AS città_di_arrivo,
    biglietto.data_acquisto,
    biglietto.stato
FROM
    Biglietto
JOIN
    Tratta ON Biglietto.tratta_id = Tratta.ID
JOIN
    Stazione AS stazione_partenza ON Tratta.stazione_partenza = stazione_partenza.ID
JOIN
    Stazione AS stazione_arrivo ON Tratta.stazione_arrivo = stazione_arrivo.ID
WHERE
    Biglietto.passeggero_id = 1;
```

risultato dopo esecuzione:

id_biglietto	id_tratta	stazione_di_partenza	città_di_partenza	stazione_di_arrivo	città_di_arrivo	data_acquisto	stato
1	1	Stazione Santa Lucia	Venezia	Stazione Centrale	Napoli	2025-07-01	confermato

3. **Verifica validità biglietto:**

Controllo se un biglietto è valido e confermato (esempio biglietto con ID 3):

```
SELECT
    Biglietto.ID AS id_biglietto,
    Passeggero.nome AS nome_passeggero,
    Passeggero.cognome AS cognome_passeggero,
    Biglietto.stato
FROM
    Biglietto
JOIN
    Passeggero ON Biglietto.passeggero_id = Passeggero.ID
WHERE
    Biglietto.ID = 3
```

```
    AND Biglietto.stato = 'confermato';
```

risultato dopo esecuzione:

ID_biglietto	nome	cognome	stato
3	Gerardo	Pizzulo	confermato

Questa query restituisce solo se il biglietto è confermato
Se lo stato è diverso (es. "annullato", "in attesa") non restituisce nulla.

4. **Elenco passeggeri su una tratta:**

Mostro i passeggeri che hanno prenotato un biglietto per una tratta specifica (esempio tratta ID 1):

```
SELECT  passeggero.ID AS ID_passeggero,
            passeggero.nome,
            passeggero.cognome,
            passeggero.email,
            biglietto.stato
FROM Biglietto
JOIN Passeggero passeggero ON biglietto.passeggero_id = passeggero.ID
WHERE biglietto.tratta_id = 1;
```

risultato dopo esecuzione:

ID_passeggero	nome	cognome	email	stato
1	Maddalena	Pagliarulo	maddalena.pagliarulo@gmail.com	confermato

5. **Report vendite per periodo:**

Conta i biglietti venduti (confermati) in un intervallo di date (esempio dal 2025-07-01 al 2025-07-31):

```
SELECT
    COUNT(*) AS numero_biglietti_venduti,
    Tratta.ID AS id_tratta,
    stazione_partenza.nome AS stazione_di_partenza,
    stazione_partenza.città AS città_di_partenza,
    stazione_arrivo.nome AS stazione_di_arrivo,
    stazione_arrivo.città AS città_di_arrivo
FROM
    Biglietto
JOIN
    Tratta ON Biglietto.tratta_id = Tratta.ID
JOIN
    Stazione AS stazione_partenza ON Tratta.stazione_partenza = stazione_partenza.ID
```

```
JOIN
    Stazione AS stazione_arrivo ON Tratta.stazione_arrivo = stazione_arrivo.ID
WHERE
    Biglietto.data_acquisto BETWEEN '2025-07-01' AND '2025-07-31'
    AND Biglietto.stato = 'confermato'
GROUP BY
    Tratta.ID,
    stazione_partenza.nome,
    stazione_partenza.città,
    stazione_arrivo.nome,
    stazione_arrivo.città;
```

risultato dopo esecuzione:

numero_biglietti_venduti	id_tratta	stazione_di_partenza	città_di_partenza	stazione_di_arrivo	città_di_arrivo
1	1	Stazione Santa Lucia	Venezia	Stazione Centrale	Napoli
1	3	Stazione Porta Nuova	Torino	Stazione Centrale	Milano
1	5	Stazione Termini	Roma	Stazione Santa Lucia	Venezia

Repository

Il progetto è disponibile al seguente repository GitHub:

https://github.com/maddalenapag/projectwork-biglietteria

Contiene il file `dump.sql` per la creazione del database e uno script per popolare i dati.

Campi di applicazione
(Descrivere gli ambiti di applicazione dell'elaborato progettuale e i vantaggi derivanti della sua applicazione):

Lo schema rappresenta un sistema di gestione di biglietti per un servizio di trasporto come un servizio ferroviario. Gli ambiti di applicazione includono:

- **Gestione dei passeggeri:** Tenere traccia dei dati personali dei passeggeri (nome, cognome, email).
- **Gestione dei biglietti:** Registrare l'acquisto dei biglietti da parte dei passeggeri, associando ogni biglietto alla tratta e allo stato del biglietto.
- **Gestione delle tratte:** Definire e tenere traccia delle tratte di viaggio, che comprendono stazioni di partenza e arrivo, orari e il mezzo di trasporto utilizzato.
- **Gestione dei mezzi di trasporto:** Registrare i diversi mezzi (tipo, codice identificativo) utilizzati per servire le tratte.
- **Gestione delle stazioni:** Mantenere informazioni sulle stazioni (nome, città), collegando le tratte alle stazioni di partenza e arrivo.

Vantaggi derivati

- **Centralizzazione delle informazioni:** Tutti i dati relativi a passeggeri, biglietti, tratte, mezzi e stazioni sono centralizzati in un unico sistema, facilitando la gestione e la consultazione.
- **Tracciabilità delle vendite:** Il sistema permette di tracciare ogni biglietto venduto, collegandolo al passeggero e alla specifica tratta, migliorando il controllo sulle vendite e l'analisi dei dati di utilizzo.

- **Gestione efficiente delle tratte:** Le tratte sono ben definite e collegate ai mezzi di trasporto e alle stazioni, permettendo una gestione precisa degli orari e dei percorsi.
- **Flessibilità:** Lo schema è strutturato per gestire diverse tipologie di mezzi di trasporto e più tratte, adattandosi a diversi contesti di trasporto pubblico o privato.
- **Riduzione degli errori:** La relazione tra biglietti, passeggeri e tratte limita errori nella gestione dei dati, grazie alla chiara definizione delle chiavi primarie e delle relazioni tra entità.
- **Miglioramento del servizio:** Permette di fornire informazioni accurate ai passeggeri e di ottimizzare le risorse di trasporto, migliorando l'esperienza utente.

Valutazione dei risultati
(Descrivere le potenzialità e i limiti ai quali i risultati dell'elaborato sono potenzialmente esposti):

Espandibilità del modello

Il modello è facilmente estendibile: si possono aggiungere nuove entità (es. tariffe, offerte, tipi di passeggeri, fermate intermedie) o attributi (es. numero di posto, classe) senza modificare radicalmente la struttura.

Automazione e digitalizzazione

Facilita l'automazione di operazioni manuali: vendita di biglietti, controllo degli accessi, gestione orari e mezzi, invio notifiche email ai passeggeri.

Monitoraggio e analisi dei dati

Il sistema può fornire statistiche sulle tratte più utilizzate, orari di punta, frequenza di acquisti da parte dei clienti, tassi di utilizzo dei mezzi, ecc.

Integrazione con altri sistemi

Può essere facilmente integrato con sistemi di pagamento elettronico e applicazioni mobili per utenti.

Personalizzazione dell'offerta

Avendo informazioni personali e comportamentali dei passeggeri, è possibile creare servizi personalizzati (es. suggerimenti di viaggio, sconti mirati).

Limiti del sistema

Assenza di storicizzazione dettagliata
Il modello non prevede un sistema per tracciare la storia delle tratte o dei mezzi (es. modifiche a orari o mezzi impiegati nel tempo). Ciò limita le analisi retroattive.

Relazioni semplificate tra entità

La relazione 1:1 tra "Tratta" e le stazioni di partenza/arrivo è rigida. Nella realtà, una tratta può avere più fermate intermedie, il che richiederebbe un modello più complesso (con una tabella intermedia).

Gestione limitata del biglietto

Il biglietto ha solo uno “stato” e manca di dettagli come:Prezzo, Metodo di pagamento, Posto assegnato, Codice QR o identificativo unico per la validazione

Sicurezza e privacy non considerate

Il modello non prevede nessun meccanismo per proteggere i dati sensibili dei passeggeri (es. crittografia dell’email) o per la gestione della privacy (consenso, diritto all’oblio).

Vincoli logistici non rappresentati

Il sistema non considera vincoli reali come: capacità dei mezzi, conflitti tra orari delle tratte, manutenzione dei mezzi o indisponibilità.

Firmato
Maddalena Pagliarulo