Progetto Basi di Dati

InfoAutobus

Furlan Ivan 1661622

A.A. 2018/2019

1. Abstract

Negli ultimi decenni, la società moderna e la vita dell'uomo sono state radicalmente modificate dall'informatizzazione e dalla successiva digitalizzazione, che hanno saputo stravolgere quasi ogni aspetto della vita quotidiana. Questi processi hanno permesso una notevole semplificazione di numerose problematiche in molti ambiti diversi. Un settore che si è scoperto completamente rivoluzionato è quello dei trasporti pubblici, che ad oggi adotta un'organizzazione interna ed esterna completamente diverse. Lo scopo di **InfoAutobus** è quello di offrire una piattaforma in grado di semplificare la gestione dei dati riguardanti i mezzi pubblici urbani e non di una qualsiasi provincia, permettendo all'azienda responsabile del servizio di avere un database che serva sia all'organizzazione dell'azienda stessa sia al cliente, in grado di accedere ai dati di suo interesse come orari e costi al fine di migliorare il servizio offerto. L'idea nasce dalla volontà di concedere la possibilità di registrare le informazioni in maniera semplice, chiara ed accessibile, ma soprattutto efficace ed utile al cliente. **InfoAutobus** permette infatti di racchiudere tutte le informazioni utili, andando ad associare sia quelle puramente amministrative come l'elenco dei dipendenti e dei mezzi, ad altre necessarie per il cliente come le tratte e gli orari e collegando il tutto.

2. Analisi dei requisiti

Si vuole realizzare una base di dati per un'azienda di autobus che permetta di gestire le informazioni partendo dai singoli viaggi percorsi.
Si vuole far si che il sistema sia utile sia all'azienda ed i suoi dipendenti, sia agli utenti del servizio, per questo è necessario che la basi di dati permetta di svolgere o consultare le seguenti operazioni e dati:

- Consultare per ogni fermata che linee passano e con che orari.
- Consultare le tappe che fa una linea e quindi capire quale percorso farà e in che destinazione arriverà.
- Gestire i biglietti elettronici che vengono venduti con la possibilità da parte del controllore di verificare se siano ancora validi.
- Aggiungere/modificare/eliminare/consultare le varie linee, le tariffe, i biglietti, i mezzi e i dipendenti.
- Possibilità di tenere traccia di tutte le corse che vengono fatte potendo risalire al conducente ed al mezzo utilizzato in un preciso momento se ce ne fosse la necessità.
- Possibilità di gestire le ore svolte dai dipendenti con il calcolo dello stipendio in automatico.

Ogni **viaggio** ha :

- un codice univoco
- una linea
- una data
- un numero che lo identifica tra le più corse che si possono effettuare sulla stessa linea in un giorno
- una stazione di partenza
- una stazione di arrivo
- un conducente
- un mezzo

Ogni **linea** ha :

- un codice identificativo
- una fermata di partenza
- una fermata di arrivo
- una breve descrizione che la identifica univocamente

Ogni **fermata** ha :

- un codice identificativo
- un paese
- un nome

Ogni **orario** ha :

- una linea
- una fermata
- un numero che identifica la corsa del giorno
- un orario
- la tipologia di giorno a cui l'orario si riferisce (*festivo, prefestivo o feriale*)

I conducenti sono un sottoinsieme dei dipendenti. Ogni **dipendente** ha:

- un codice fiscale
- un nome
- un cognome
- una data di nascita
- un sesso
- un ruolo all'interno dell'azienda
- un numero di telefono
- un indirizzo email
- una tariffa di retribuzione oraria

I dipendenti lavorano a turni. Ogni **turno** ha:

- un codice identificativo
- il codice fiscale di un dipendente
- un orario di inizio
- un orario di fine

Ogni **mezzo** ha :

- un numero di matricola
- un tipo *(esempio: autobus, minibus, pulmino...)*
- una marca
- un sistema di alimentazione
- una anno di produzione
- un numero di posti a sedere
- un numero di posti per disabili
- una lunghezza del veicolo
- una larghezza del veicolo
- un'altezza del veicolo
- un luogo di deposito

Ogni **deposito** ha :

- un identificativo
- un paese
- un indirizzo
- un numero di parcheggi

Ogni **biglietto** ha :

- un identificativo
- una tipologia in base alla tratta *(urbano o extraurbano)*
- una tariffa
- un fermata di partenza *
- una fermata di arrivo *
- un orario di convalida

** non necessarie se il biglietto è urbano*

Ogni **tariffa** ha :

- un nome
- un prezzo
- un classe per i chilometri *(se è una tariffa extraurbana)*
- un tempo di utilizzo del biglietto *(se è una tariffa urbana)*

2.1 Glossario dei termini

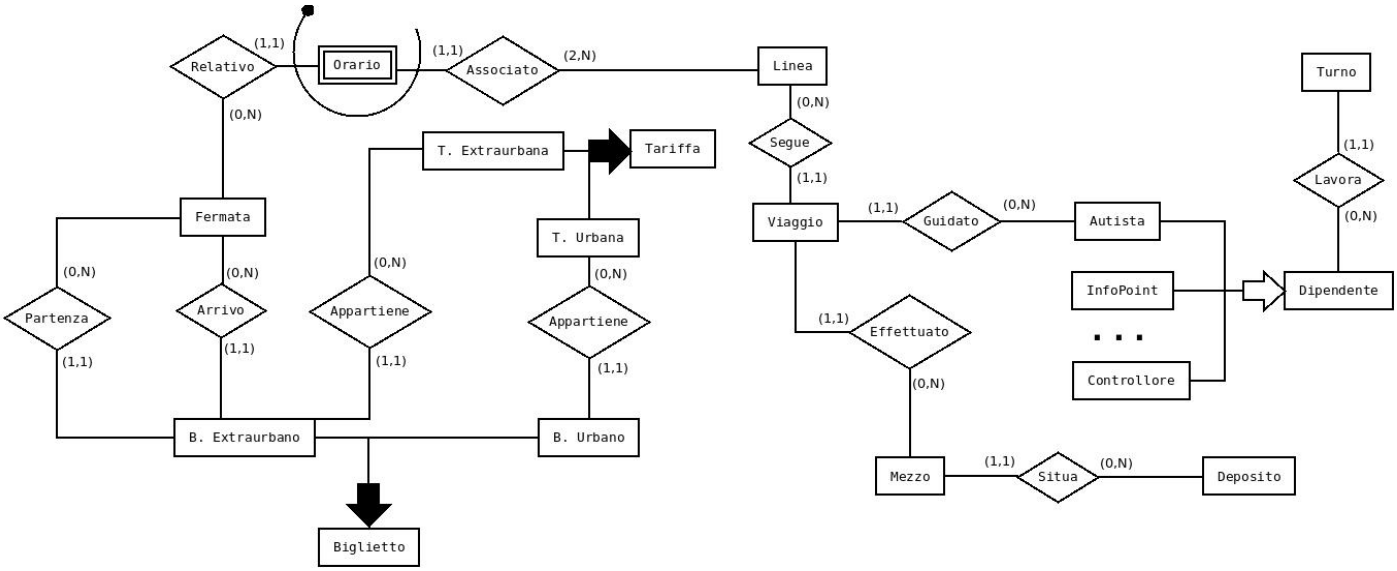
Termini	Descrizione	Sinonimi	Collegamenti
Viaggio	Singolo viaggio effettuato un determinato giorno. *(Sono considerati differenti due viaggi sulla stessa linea nello stesso giorno)*		Orario, Fermata, Mezzo, Dipendente
Tipologia di giorno	Indica se un giorno è festivo, prefestivo o feriale.	Tipo di giorno	Viaggio, Orario
Corsa	Identifica la corsa di una linea effettuata lo stesso giorno sullo stesso tragitto indipendentemente dal tipo di giorno, ossia verranno conteggiati anche le corse che un giorno potranno non essere effettuati in quanto quel giorno sarà festivo, prefestivo, o feriale. La prima corsa su una linea in un giorno sarà la *1,* di conseguenza la corsa successiva sarà la *2*. Nel caso la *3* fosse festiva, ma si sia in un giorno feriale, allora l'effettiva corsa successiva che verrà effettuata sarà la *4*.	Viaggio del giorno	Viaggio, Orario
Orario	L'orario per le varie fermate di una linea, in base alla tipologia di giorno (feriale, prefestivo o festivo) con anche a che corsa ci si riferisce		Viaggio, Fermata, Linea
Fermata	Identifica una fermata che può appartenere a più linee, o nessuna nel caso sia una fermata non ancora o non più servita. Le eventuali stazioni di partenza e arrivo sono considerate fermate.	Stazione	Biglietto, Orario, Viaggio
Biglietto	Il singolo biglietto che è stato acquistato		Fermata, Tariffa
Tariffa	Costo delle varie possibili tipologie di biglietti		Biglietto
Mezzo	Identifica un mezzo adibito al trasposto di passeggeri che viene utilizzato per effettuare il servizio pubblico	Veicolo	Viaggio, Deposito
Deposito	Luogo di parcheggio dei mezzi quando non vengono utilizzati		Mezzo
Dipendente	Lavoratore dell'azienda		Viaggio, Turno
Turno	Indica il singolo turno di lavoro effettuato da un dipendente un determinato giorno		Dipendente

Ruolo	Indica i ruoli che ha un dipendente all'interno dell'azienda. Può essere un set di ruoli in quanto un dipendente potrebbe svolgere diverse mangioni o essere qualificato a farne più di una.		Dipendente

2.2 Operazioni

- **Operazione 1 (***nuovaCorsa***)**: permette di aggiungere una nuova corsa per quella linea, copiandone una già esistente con un orario di partenza diverso. Automaticamente copia la linea con i nuovi orari calcolati sulla differenza di tempo tra una fermata e l'altra della linea già presente
- **Operazione 2 (***eliminaCorsa***)**: permette di eliminare una corsa di una linea, modificando il codice giornaliero degli altri viaggi della stessa linea
- **Operazione 3 (***mostraLinea***)**: Data una linea in ingresso mostra le fermate che fa e quante corse ci sono ogni giorno in base se il giorno sia feriale, prefestivo, o festivo
- **Operazione 4 (***mostraOrariViaggio***)**: Funzione che verrà usata dal conducente del mezzo prima di partire. Dato l'id del viaggio mostra il percorso che dovrà fare con le varie fermate e i rispettivi orari per ognuna di esse
- **Operazione 5 (***tabellonePartenze***)**: Dato il codice di una fermata e un orario (xx:xx:xx), mostra tutte le corse delle linee passanti per quella fermata dopo tale orario, la destinazione finale di ogni linea, e se passa in un giorno feriale, prefestivo, o festivo
- **Operazione 6 (***calcoloStipendiDipendente***)**: Dato il Codice fiscale di un dipendente, un mese e un anno, restituisce lo stipendio di tale dipendente nel mese indicato.
- **Operazione 7 (***oreMeseSolareDipendente***)**: Dato il CF di un dipendente, un mese e un anno, restituisce le ore di lavoro di tale dipendente nel mese indicato
- **Operazione 8 (***stazioneArrivoLinea***)**: Data una linea restituisce la fermata di arrivo
- **Operazione 9 (***stazionePartenzaLinea***)**: Data una linea restituisce la fermata di partenza
- **Operazione 10 (***validitaBigliettoUrbano***)**: Dato in ingresso l'identificativo di un biglietto URBANO, restituisce TRUE se il biglietto è ancora valido (non ancora convalidato, oppure non ancora scaduto il tempo), altrimenti FALSE

3. Progettazione Concettuale



3.1 Lista delle entità

Gli attributi di colore *viola* possono essere *NULL*

Entità	Descrizione	Attributi	Identificatore
Dipendente	Persona che lavora all'interno dell'azienda di trasporti	CF: **char(16)** Nome: **varchar(30)** Cognome: **varchar(30)** *Data_nascita:* ***date*** Sesso: **char(1)** *Telefono:* ***varchar(14)*** *Email:* ***varchar(100)*** costoOrario: **decimal(10,2)**	CF
Conducente	Dipendente abilitato alla guida di un mezzo per il trasporto pubblico	Eredita da Dipendente	CF
Turno	Data e orario di inizio e fine lavoro di un dipendente	Id: **int(11)** Inizio: **datetime** *Fine:* ***datetime***	Id
Mezzo	Il mezzo con cui si effettua il servizio pubblico (autobus, minibus, filobus...)	Matricola: **int(11)** Tipo: **varchar(15)** Marca: **varchar(20)** Alimentazione: **enum(...)** *Anno:* ***year(4)*** Posti_sedere: **int(11)** Posti_sediaRotelle: **int(11)** Lunghezza_veicolo_mt: **decimal(4,2)** Larghezza_veicolo_mt: **decimal(4,2)** Altezza_veicolo_mt: **decimal(4,2)**	Matricola
Deposito	Indica il posto dove solitamente sostano i mezzi per il trasporto pubblico quando non vengono utilizzati	Id: **int(11)** Paese: **varchar(40)** Indirizzo: **varchar(50)** Parcheggi: **int(11)**	Id
Viaggio	Il singolo viaggio che effettua una corsa in un determinato giorno su una linea	Id: **int(11)** Linea: **char(5)** Corsa: **int(11)** Giorno: **date**	Id
Linea	La linea dove possono venir effettuate più corse, e vengono percorse dai vari viaggi.	Codice: **char(5)** Descrizione: **varchar(50)**	Codice
Fermata	La fermata dove passano le varie linee che la servono. Comprende le stazioni di partenza e arrivo.	Id: **int(11)** Paese: **varchar(30)** Descrizione: **varchar(50)**	Id
Orario	L'ora che una linea passa ad una fermata, e se quel orario è festivo, prefestivo, o feriale	*Linea*: **char(5)** Corsa: **int(11)** Id_fermata: **int(11)** *O*ra: **time** Tipo: **enum(...)**	*Linea*, viaggio_giorno, *id_fermata*
Biglietto	Biglietto che è già stato venduto a un cliente	Id: **int(11)** *Ora_convalidato:* ***datetime***	Id

Entità	Descrizione	Attributi	Identificatore
B. Extraurbano	Biglietto per le linee extraurbane	Eredita da Biglietto	Id
B. Urbano	Biglietto per le linee urbane	Eredita da Biglietto	Id
Tariffa	Indica la tariffa di un biglietto	Nome: **char(5)** Prezzo: **float**	Nome
T. Extraurbana	Indica la tariffa di un biglietto extraurbano	Eredita da Tariffa Classe_km: **float**	Nome
T. Urbana	Indica la tariffa di un biglietto urbano	Eredita da Tariffa Tempo: **time**	Nome

3.2 Lista delle relazioni con relative cardinalità

Relazione	Descrizione cardinalità	Entità coinvolte
Lavorato	Un dipendente ha lavorato 0, 1 o più turni (0,N) Ogni turno è lavorato da un dipendente (1,1)	Dipendente, Turno
Guidato	Ogni viaggio è svolto da un autista (1,1) Un autista può svolgere più viaggi (0,N)	Viaggio, Autista
Effettuato	Ogni viaggio è effettuato con un mezzo (1,1) Ogni mezzo può effettuare più viaggi (0,N)	Viaggio, Mezzo
Percorso	Ogni viaggio ha percorso una linea (1,1) Ogni linea può essere percorsa da più viaggi (0,N)	Viaggio, Linea
Situa	Ogni mezzo situa in un deposito (1,1) In ogni deposito possono essere situati dei mezzi (0,N)	Mezzo, Deposito
Associato	Ogni orario è associato ad una linea (1,1) Ad ogni linea vengono associati almeno 2 orari (2,N)	Orario, Linea
Relativo	Ogni orario è relativo ad una fermata (1,1) Una fermata può avere dei relativi orari (0,N)	Orario, Fermata
Partenza	Ogni Biglietto Extraurbano ha una fermata di partenza (1,1) Una fermata può essere di partenza per uno o più biglietti extraurbani (0,N)	B. Extraurbano, Fermata
Arrivo	Ogni Biglietto Extraurbano ha una fermata di arrivo (1,1) Una fermata può essere di arrivo per uno o più biglietti extraurbani (0,N)	B. Extraurbano, Fermata
Appartiene	Ad ogni B.E. appartiene una Tariffa E. (1,1) Una Tariffa E. può appartenere ad un B.E. (0,N)	B. Extraurbano, T. Extraurbana
Appartiene	Ad ogni B.U. appartiene una Tariffa U. (1,1) Una Tariffa U. può appartenere ad un B.U. (0,N)	B. Urbano, T. Urbana

4. Progettazione Logica

4.1 Ristrutturazione ed eliminazione delle generalizzazioni

Nello schema sono presenti le seguenti generalizzazioni che abbiamo analizzato.

Dipendente: potenzialmente un dipendente può avere più di un ruolo, e creare un entità per ognuno di essi potrebbe portare ad un elevata ridondanza, inoltre l'unica entità figlia che si relaziona con il resto dello schema sarebbe Autista. È stato perciò deciso di creare un unica entità Dipendente ed impostare un trigger che controlli all'aggiunta di una nuova relazione con Viaggio che il Dipendente sia un Autista, altrimenti impedisce l'operazione.

Biglietto: urbano o extraurbano non differiscono dall'entità padre per gli attributi che hanno, ma solo per le relazioni. Soltanto nello schema logico il biglietto extraurbano avrebbe due chiavi esterne in più che identificherebbero le fermate di partenza ed arrivo. Per gestire meglio la chiave primaria dei biglietti che deve essere univoca indipendentemente dal tipo di biglietto si è preferito accorpare le entità in quella padre aggiungendo un attributo che caratterizzi la tipologia di biglietto. Successivamente, nella implementazione, nel caso di biglietti urbani gli attributi che identificano le fermate di partenza e arrivo per gli extraurbani saranno impostati a NULL.

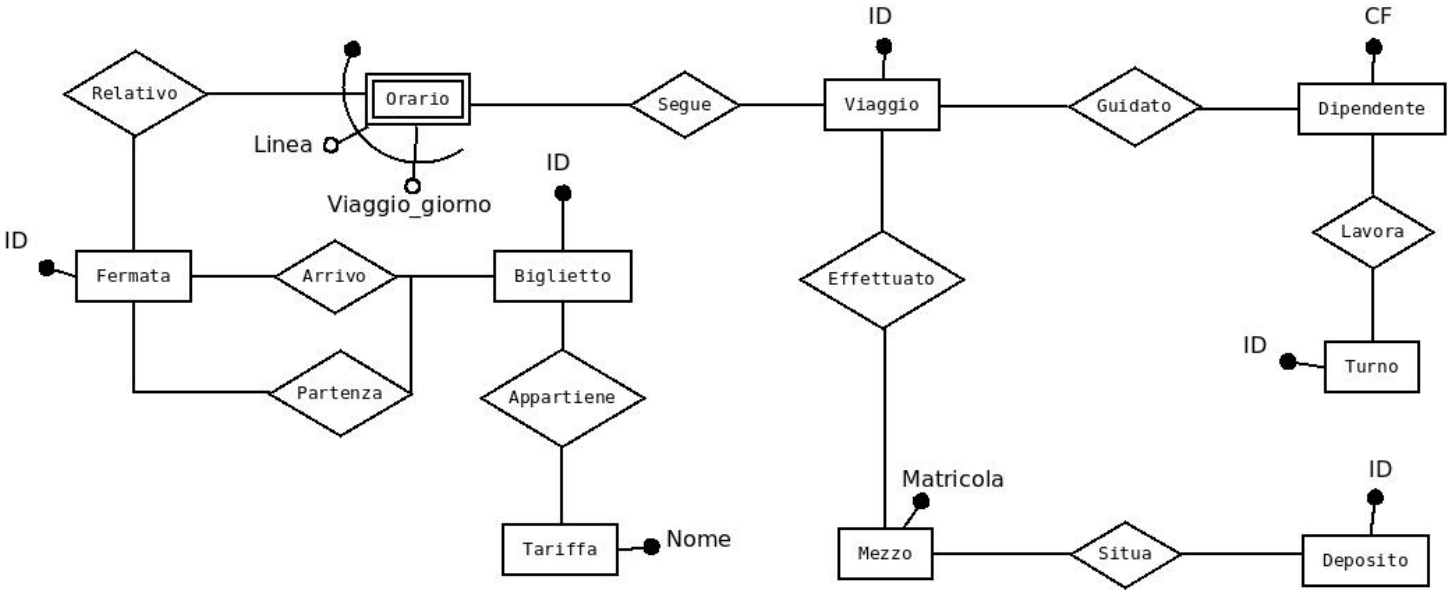
Tariffa: nell'implementazione della base di dati l'entità Biglietto conterrà un attributo per relazionarsi con la relativa tariffa in base a che sia urbano o extraurbano. Il mantenere separate le entità complicherebbe l'associare l'attributo nell'entità Biglietto con la giusta entità Tariffa, perciò si è preferito anche in questo caso accorpare tutte le entità figlie nel padre.

L'entità Linea che si relaziona con Orario e Viaggio non è necessaria per due motivi:

1. I dati in essa contenuti sono ridondanti in quanto si possono ricavare dall'entità Orario
2. Non vengono svolte operazioni che la interessano direttamente e/o univocamente

Di conseguenza è stato decidere di eliminarla mettendo direttamente in relazione Viaggio con Orario e creando una vista Linee che di fatto è quella che dovrebbe essere stata l'entità Linea.

4.2 Schema ER ristrutturato



4.3 Modello logico-relazionale

Dipendente(**CF**, Nome, Cognome, Data_Nascita, Sesso, Ruolo, Telefono, Email, costoOrario)
Turno(**id**, *cf_dipendente*, inizio, fine)
Viaggio(**id**, *linea*, *corsa*, giorno, *conducente*, *id_mezzo*)
Mezzo(**matricola**, tipo, marca, alimentazione, anno, posti_sedere, posti_sediaRotelle, lunghezza_mt, larghezza_mt, altezza_mt, *deposito*)
Deposito(**id**, paese, indirizzo, parcheggi)
Orario(**linea**, **corsa, id_fermata**, ora, tipo)
Fermata(**id,** paese, descrizione)
Biglietto(**id**, tipologia, *nome_tariffa*, *fermata_partenza*, *fermata_arrivo*, ora_convalidato)
Tariffa(**nome**, classe_km, tempo, prezzo)

5. Query significative (Viste e Procedure), Funzioni, Trigger

5.1 Query

Per questioni di spazio non sempre viene riportato l'output di tutte le tuple, e nella vista/query "*dettaglioViaggi*" considerato l'elevato numero di attributi vengono riportati solo i principali, sufficienti a descrivere e comprendere la query.

Vista dettaglioViaggi

Mostra ogni singolo viaggio con tutti i dettagli principali che lo riguardano, come i dati sulla corsa, le fermati di arrivo e partenza, e i dati potenzialmente utili del conducente e del mezzo utilizzato.

IdViaggio	GiornoViaggio	TipoGiorno	Linea	Corsa	IdFermataP	oraP	IdFermataA	OraA	CFConducente	MatricolaVeicolo
1	2019-01-14	Feriale	E99A	1	14	7:00:00	15	8:00:00	FRLVNI98T09F770V	2
13	2019-01-21	Feriale	U03A	3	1	6:35:00	13	6:57:00	WTFYSV73C11S161T	16

```
CREATE VIEW `dettaglioViaggi` AS SELECT `V`.`id` AS `idViaggio`, `V`.`giorno` AS `giornoViaggio`, `OP`.`tipo` AS `tipoGiorno`, `V`.`linea` AS `linea`, `V`.`viaggio_giorno` AS `viaggio_giorno`, `FP`.`id` AS `idFermataPartenza`, `FP`.`Paese` AS `PaesePartenza`, `FP`.`descrizione` AS `FermataPartenza`, `OP`.`ora` AS `OraPartenza`, `FA`.`id` AS `idFermataArrivo`, `FA`.`Paese` AS `PaeseArrivo`, `FA`.`descrizione` AS `FermataArrivo`, `OA`.`ora` AS `OraArrivo`, `D`.`CF` AS `CFConducente`, concat(`D`.`Nome`, ' ', `D`.`Cognome`) AS `NomeConducente`, `M`.`tipo` AS `Veicolo`, `M`.`marca` AS `marcaVeicolo`, `M`.`alimentazione` AS `alimentazioneVeicolo`, `M`.`posti_sedere` AS `posti_sedere`, `M`.`posti_sediaRotelle` AS `posti_sediaRotelle`, `M`.`lunghezza_veicolo_mt` AS `lunghezza_veicolo_mt`, `M`.`larghezza_veicolo_mt` AS `larghezza_veicolo_mt`, `M`.`altezza_veicolo_mt` AS `altezza_veicolo_mt`
FROM ((((((`Viaggio` `V` JOIN `Orario` `OP`) JOIN `Orario` `OA`) JOIN `Fermata` `FP`) JOIN `Fermata` `FA`) JOIN `Dipendente` `D`) JOIN `Mezzo` `M`)
WHERE ((`V`.`conducente` = `D`.`CF`) AND (`V`.`id_mezzo` = `M`.`matricola`) AND (`V`.`linea` = `OP`.`linea`) AND (`V`.`viaggio_giorno` = `OP`.`viaggio_giorno`) AND (`V`.`linea` = `OA`.`linea`) AND (`V`.`viaggio_giorno` = `OA`.`viaggio_giorno`) AND (`OA`.`id_fermata` = `FA`.`id`) AND (`OP`.`id_fermata` = `FP`.`id`)
AND (`V`.`id`, `OA`.`ora`) IN (SELECT `VV`.`id` AS `id`, max(`OO`.`ora`) AS `ora` FROM (`Viaggio` `VV` JOIN `Orario` `OO`) WHERE ((`VV`.`linea` = `OO`.`linea`) AND (`VV`.`viaggio_giorno` = `OO`.`viaggio_giorno`)) GROUP BY `VV`.`id`)
AND (`V`.`id`,`OP`.`ora`) IN (SELECT `VV`.`id` AS `id`, min(`OO`.`ora`) AS `ora` FROM (`Viaggio` `VV` JOIN `Orario` `OO`) WHERE ((`VV`.`linea` = `OO`.`linea`) AND (`VV`.`viaggio_giorno` = `OO`.`viaggio_giorno`)) GROUP BY `VV`.`id`))
ORDER BY `V`.`giorno`, `OP`.`ora`, `V`.`id`;
```

Vista dipendentiPerRuolo

Mostra per ogni ruolo all'interno dell'azienda quanti dipendenti ci sono, in modo da capire come è distribuito il personale nei vari ambiti.

Ruolo	numeroDipendenti
Dipendente	33
Conducente, Dipendente	23
Controllore, Dipendente	21
Conducente, Controllore, Dipendente	1
InfoPoint, Dipendente	24

```
CREATE VIEW `dipendentiPerRuolo` AS select `Dipendente`.`Ruolo` AS `Ruolo`,count(0) AS `numeroDipendenti` from `Dipendente` group by `Dipendente`.`Ruolo` order by `Dipendente`.`Ruolo` ;
```

Vista Linee

Mostra per ogni linea tutti i dettagli come partenza ed arrivo. Le linee vengono ricavate dalla tabella orario. Utilizza le funzioni successivamente riportate stazionePartenzaLinea e stazioneArrivoLinea.

Linea	Descrizione	IdStazionePartenza	StazionePartenza	IdStazioneArrivo	stazioneArrivo
U03A	Padova Ferrovia – Padova Lion di Albignasego	1	Padova Ferrovia	13	Padova Lion di Albignasego
E99A	Rovigo Stazione - Venezia Mestre Stazione	14	Rovigo Stazione	15	Venezia Mestre Stazione
E99C	Rovigo Stazione - Venezia Piazzale Roma	14	Rovigo Stazione	16	Venezia Piazzale Roma

```
CREATE VIEW `Linee` AS select distinct `O`.`linea` AS `linea`, concat(`FP`.`Paese`, ' ', `FP`.`descrizione`, ' - ', `FA`.`Paese`, ' ', `FA`.`descrizione`) AS `Descrizione`, `FP`.`id` AS `idStazionePartenza`, concat(`FP`.`Paese`,' ',`FP`.`descrizione`) AS `stazionePartenza`,`FA`.`id` AS `idStazioneArrivo`,concat(`FA`.`Paese`,' ',`FA`.`descrizione`) AS `stazioneArrivo` from ((`Orario` `O` join `Fermata` `FA`) join `Fermata` `FP`) where ((`O`.`id_fermata` = `stazionePartenzaLinea`(`O`.`linea`)) and (`O`.`id_fermata` = `FP`.`id`) and (`FA`.`id` = `stazioneArrivoLinea`(`O`.`linea`))) ;
```

Vista StipendiUltimoMese

Mostra gli stipendi di tutti i dipendenti nel mese precedente a quello attuale. Utilizza le funzioni successivamente riportate oreMeseSolareDipendente e calcoloStipendioDipendente.

CF	Nome	Ruolo	Mese	CostoOrario	Ore	Stipendio
CHCMXB33Z49B533G	Rollins Giuseppe	InfoPoint, Dipendente	January	7,5	120	900
FRLVNI98T09F770V	Furlan Ivan	Conducente, Controllore, Dipendente	January	9,5	180	1710

```
CREATE VIEW `StipendiUltimoMese` AS select `D`.`CF` AS `CF`,concat(`D`.`Cognome`,' ', `D`.`Nome`) AS `Nome`,`D`.`Ruolo` AS `Ruolo`,date_format((curdate() - interval 1 month),'%M') AS `Mese`, `D`.`costoOrario` AS `costoOrario`, `oreMeseSolareDipendente`(`D`.`CF`, extract(month from (curdate() - interval 1 month)), extract(year from (curdate() - interval 1 month))) AS `Ore`,`calcoloStipendioDipendente`(`D`.`CF`,extract(month from (curdate() - interval 1 month)),extract(year from (curdate() - interval 1 month))) AS `Stipendio` from `Dipendente` `D` order by `oreMeseSolareDipendente`(`D`.`CF`,extract(month from (curdate() - interval 1 month)), extract(year from (curdate() - interval 1 month))), `calcoloStipendioDipendente`(`D`.`CF`, extract(month from (curdate() - interval 1 month)),extract(year from (curdate() - interval 1 month))) desc ;
```

Procedura tabellonePartenze

Ricevendo in ingresso l'id di una fermata (e opzionalmente un ora di partenza) mostra tutte le linee che passano per di lì, con la relativa ora, la destinazione del viaggio, e il tipo di giorno in cui passa.

CALL `tabellonePartenze`(1, '');

Linea	Destinazione	OraPassaggio	Giorno
U03A	0	6:20:00	0
E99A	0	7:30:00	0
E99C	Venezia - Piazzale Roma	12:30:00	Prefestivo

```
CREATE PROCEDURE `tabellonePartenze` (IN `id_fermata` INT, IN `ora` TIME)  NO SQL
BEGIN
	SELECT O.linea, concat(F.Paese, ' - ', F.descrizione) AS Direzione, O.ora AS OraPassaggio, O.tipo AS Giorno
	from Orario O, (
		SELECT O.linea, O.corsa, O.id_fermata
		FROM Orario O
		WHERE (O.linea, O.corsa, O.ora) in (
		  SELECT O.linea, O.corsa, MAX(O.ora) AS ora
		  FROM Orario O
		  GROUP BY O.linea, O.corsa)
		) L, Fermata F
	WHERE O.id_fermata=id_fermata AND O.ora>=ora AND O.linea = L.linea AND O.corsa = L.corsa AND L.id_fermata = F.id AND F.id <> id_fermata
	ORDER BY OraPassaggio,linea,Direzione;

END$$
```

Procedura eliminaCorsa

Ricevuto in ingresso una linea già esistente e una specifica corsa, elimina tale corsa aggiornando il numero delle altre eventualmente rimanenti. Mostra la corsa eliminata.

CALL `eliminaCorsa`('U03A', '6');

Linea	Corsa	IdFermata	Ora	Tipo
U03A	6	1	06:55:00	0
U03A	6	2	06:56:00	Festivo
…	…	…	…	…
U03A	6	13	07:17:00	Festivo

```
CREATE PROCEDURE `eliminaCorsa` (IN `linea` CHAR(5), IN `corsa_` INT)  NO SQL
BEGIN
	SELECT * FROM Orario WHERE Orario.linea=linea AND Orario.corsa=corsa_;

	DELETE FROM Orario WHERE Orario.linea=linea AND Orario.corsa=corsa_;

	UPDATE Orario SET Orario.corsa=Orario.corsa - 1 WHERE Orario.linea = linea AND Orario.corsa > corsa_;
END$$
```

Procedura nuovaCorsa

Ricevuto in ingresso una linea già esistente, aggiunge una nuova corsa con il nuovo orario di partenza inserito, e il tipo di giorno in cui verrà effettuata. Gli orari tra le varie fermate sono gli stessi della corsa già presente dalla quale "copia" le informazioni. Mostra la nuova corsa aggiunta.

CALL `nuovaCorsa`('U03A', '19:15:00','festivo');

Linea	Corsa	IdFermata	Ora	Tipo	Paese	Descrizione
U03A	7	1	19:15:00	0	0	0
U03A	7	2	19:16:00	Festivo	Padova	Corso del Popolo
…	…	…	…	…	…	...
U03A	7	13	19:37:00	Festivo	Padova	Lion di Albignasego

```
CREATE PROCEDURE `nuovaCorsa` (IN `da_linea` CHAR(5), IN `new_ora` TIME, IN `new_tipo_giorno` VARCHAR(15)) BEGIN
	DROP TABLE IF EXISTS `percorso`;
	CREATE TABLE `percorso` (
	  `linea` char(5) NOT NULL,
	  `corsa` int(11) NOT NULL,
	  `id_fermata` int(11) NOT NULL,
	  `ora` time NOT NULL,
	  `tipo` varchar(15) NOT NULL DEFAULT 'feriale'
	);
	SET @daVG = 1;
	INSERT INTO `percorso` SELECT * FROM `Orario` WHERE `linea` = da_linea AND `corsa`=@daVG ORDER BY `ora`;
	SET @vg = (SELECT MAX(`corsa`) FROM `Orario` WHERE `linea`=da_linea) + 1;
	UPDATE `percorso` SET `corsa` = @vg;
	IF(new_tipo_giorno<>'') THEN
		UPDATE `percorso` SET `tipo` = new_tipo_giorno;
	END IF;
	SET @temp = (SELECT P.ora FROM percorso P ORDER BY P.ora LIMIT 1);
	IF (@temp < new_ora) THEN
		SET @dif = TIME_TO_SEC(TIMEDIFF(new_ora,@temp));
		UPDATE `percorso` SET `ora` = ADDTIME(`ora`,SEC_TO_TIME(@dif));
	ELSE
		SET @dif = TIME_TO_SEC(TIMEDIFF(@temp,new_ora));
		UPDATE `percorso` SET `ora` = ADDTIME(`ora`,SEC_TO_TIME(@dif));
	END IF;
	INSERT INTO `Orario` SELECT * FROM `percorso`;
	DROP TABLE `percorso`;
	SELECT O.*, F.Paese, F.descrizione FROM Orario O, Fermata F WHERE O.id_fermata = F.id AND O.linea=da_linea AND O.corsa=@vg ORDER BY O.ora;
END$$
```

Procedura mostraLinea

Riceve come parametro il nome della linea e mostra in ordine le fermate che fa e quante corse feriali, prefestive e festive giornaliere ci sono.

CALL `mostraLinea`('E99A');

Linea	IdFermata	Paese	FermaIn	CorseFeriali	CorsePrefestive	CorseFesive
E99A	14	Rovigo	Stazione	3	1	1
E99A	1	Padova	Ferrovia	3	1	1
E99A	15	Venezia	Mestre Stazione	3	1	1

```
CREATE PROCEDURE `mostraLinea` (IN `linea` CHAR(5)) BEGIN
	SELECT OO.linea, FF.*
	from(
	    SELECT F.id as idFermata, F.Paese, F.descrizione as FermaIn, sum(case when O.tipo = 'feriale' then 1 else 0 end) AS ViaggiFerialiGiornalieri, sum(case when O.tipo = 'prefestivo' then 1 else 0 end) AS ViaggiPrefestiviGiornalieri, sum(case when O.tipo = 'festivo' then 1 else 0 end) AS ViaggiFestiviGiornalieri
	      FROM  Orario O, Fermata F
	      WHERE O.linea=linea AND O.id_fermata=F.id
	      GROUP BY idFermata, F.Paese, FermaIn
	      ) FF, Orario OO
	WHERE FF.idFermata=OO.id_fermata AND OO.corsa=1 AND OO.linea=linea
	ORDER BY OO.ora;
END$$
```

Procedura mostraOrariViaggio

Procedura che viene principalmente utilizzata dal conducente di un viaggio e visualizza una volta dato in input l'id del viaggio tutte le fermate che deve fare e il relativo orario. Può servire anche all'utente per visualizzare gli orari delle fermate del Viaggio in cui viaggia.

CALL `mostraOrariViaggio`('5');

IdFermata	Paese	FermaIn	oraArrivo
14	Rovigo	Stazione	7:00:00
1	Padova	Ferrovia	7:30:00
15	Venezia	Mestre Stazione	8:00:00

```
CREATE PROCEDURE `mostraOrariViaggio` (IN `idViaggio` INT)  NO SQL
BEGIN
	SELECT F.id as idFermata, F.Paese, F.descrizione as FermaIn, O.ora AS oraArrivo
	FROM  Orario O, Fermata F, Viaggio V
	WHERE V.id=idViaggio AND O.linea=V.linea AND O.id_fermata=F.id AND O.corsa=V.corsa
	ORDER BY O.ora;
END$$
```

5.2 Funzioni

StazioneArrivoLinea

Inserendo come parametro il codice di una linea restituisce la relativa fermata di arrivo.

```
CREATE  FUNCTION `stazioneArrivoLinea` (`linea` CHAR(5)) RETURNS INT(11) NO SQL
   DETERMINISTIC
BEGIN
	DECLARE stazioneArrivo INT;
	DECLARE vg INT;
	SET stazioneArrivo = 0;
	SET vg = 1;
	SELECT O.id_fermata INTO stazioneArrivo
	FROM Orario O
	WHERE O.linea=linea AND O.corsa=vg
	ORDER BY O.ora DESC
	LIMIT 1;
	RETURN stazioneArrivo;
END$$
```

OreMeseSolareDipendente

Restituisce quante ore di lavoro ha svolto un dipendente nel mese inserito tra i parametri.

```
CREATE  FUNCTION  `oreMeseSolareDipendente`  (`cf_dipendente`  CHAR(16),  `Mese`  TINYINT,
`Anno` SMALLINT) RETURNS DECIMAL(10,2) NO SQL
   DETERMINISTIC
BEGIN
	DECLARE mes TINYINT;
	DECLARE ann SMALLINT;
	DECLARE ore DECIMAL(10,2);
	IF(Mese>0 AND Anno>1900) THEN
		SET mes = Mese;
		SET ann =Anno;
	ELSE
		SET mes = MONTH(CURRENT_DATE);
		SET ann = YEAR(CURRENT_DATE);
	END IF;
	SET ore =0;
	SELECT TIME_TO_SEC(SUM(TIMEDIFF(T.fine,T.inizio)))/3600 INTO ore
	FROM Turno T
	WHERE  T.cf_dipendente=  cf_dipendente  AND  EXTRACT(MONTH  FROM  T.inizio)  =  mes
AND EXTRACT(YEAR FROM T.inizio) = ann;
	RETURN ore;
END$$
```

CalcoloStipendioDipendente

Restituisce lo stipendio di un dipendente nel mese inserito tra i parametri. Utilizza la funzione oreMeseSolareDipendente.

```
CREATE FUNCTION `calcoloStipendioDipendente` (`cf_dipendente` CHAR(16), `Mese` TINYINT UNSIGNED, `Anno` SMALLINT UNSIGNED) RETURNS DECIMAL(10,2) NO SQL
   DETERMINISTIC
BEGIN
        DECLARE mes TINYINT;
        DECLARE ann SMALLINT;
        DECLARE stipendio DECIMAL(10,2);
        IF(Mese>0 AND Anno>1900) THEN
                SET mes = Mese;
                SET ann =Anno;
        ELSE
                SET mes = MONTH(curdate() - interval 1 month);
                SET ann = YEAR(curdate() - interval 1 month);
        END IF;
        SET stipendio =0;
        Select D.costoOrario*oreMeseSolareDipendente(cf_dipendente, mes, ann) INTO stipendio
        FROM Dipendente D
        WHERE D.CF=cf_dipendente;
        RETURN stipendio;
END$$
```

ValiditàBigliettoUrbano

Ricevendo in input l'id di un biglietto urbano verifica se al momento dell'invocazione della funzione esso sia ancora valido.

```
CREATE FUNCTION `validitaBigliettoUrbano` (`idBigliettoUrbano` INT) RETURNS TINYINT(1) NO SQL
   DETERMINISTIC
BEGIN
        DECLARE scadenza DATETIME DEFAULT '1800-01-01 00:00:00';
        SELECT ADDTIME(IFNULL(B.ora_convalidato, CURRENT_TIME),T.tempo) INTO scadenza FROM Biglietto B, Tariffa T WHERE B.id=idBigliettoUrbano AND B.nome_tariffa=T.nome AND T.nome LIKE 'U%';
        CALL debug(scadenza);
        IF (scadenza <= CURRENT_TIMESTAMP) THEN
                RETURN false;
        ELSE
                RETURN true;
        END IF;
END$$
```

5.3 Trigger

CheckConducente

Non permette l'inserimento in un viaggio di un dipendente che non sia anche conducente. Viene eseguito prima dell'inserimento di un nuovo viaggio.

```
CREATE TRIGGER `CheckConducente` BEFORE INSERT ON `Viaggio` FOR EACH ROW BEGIN
	DECLARE dip CHAR(16) DEFAULT "";
	SELECT D.CF INTO dip FROM Dipendente D WHERE D.CF=NEW.conducente AND FIND_IN_SET('Conducente',D.Ruolo);

	IF(dip <> NEW.conducente) THEN
		SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "Il dipendente non è anche un conducente, o non esiste";
	END IF;
END$$
```

AggiuntaTurno

Aggiunge un nuovo turno al conducente che effettua un viaggio nel caso non abbia nessun turno già inserito che copra l'orario della corsa. Se ci fosse già un turno che però copre soltanto parzialmente la corsa, allora provvede ad anticipare o posticipare rispettivamente l'ora di inizio o di fine di esso.

```
CREATE TRIGGER `AggiuntaTurno` AFTER INSERT ON `Viaggio` FOR EACH ROW BEGIN
	DECLARE part TIME;
	DECLARE arr TIME;
	DECLARE idTurnoP INT;
	DECLARE idTurnoA INT;
	SELECT O.ora INTO part FROM Orario O WHERE O.linea = NEW.linea AND O.corsa = NEW.corsa AND O.id_fermata = stazionePartenzaLinea(NEW.linea);
	SELECT O.ora INTO arr FROM Orario O WHERE O.linea = NEW.linea AND O.corsa = NEW.corsa AND O.id_fermata = stazioneArrivoLinea(NEW.linea);
	SET idTurnoP =0;
	SELECT T.id INTO idTurnoP
	FROM Turno T
	WHERE NEW.conducente = T.cf_dipendente
	AND T.inizio BETWEEN
	   TIMESTAMP(
		NEW.giorno,part
	   ) AND TIMESTAMP(
		NEW.giorno,arr
	   );
	IF idTurnoP <> 0 THEN
		UPDATE Turno T SET T.inizio = TIMESTAMP(NEW.giorno,part) WHERE T.id = idTurnoP;
	END IF;
	SET idTurnoA = 0;
	SELECT T.id INTO idTurnoA
	FROM Turno T
	WHERE NEW.conducente = T.cf_dipendente
	AND T.inizio BETWEEN
	   TIMESTAMP(
		NEW.giorno,part
	   ) AND TIMESTAMP(
		NEW.giorno,arr
	   );
	IF idTurnoA <> 0 THEN
		UPDATE Turno T SET T.inizio = TIMESTAMP(NEW.giorno,part) WHERE T.id = idTurnoA;
	END IF;
	IF idTurnoP = 0 AND idTurnoA = 0 THEN
		INSERT INTO Turno(`cf_dipendente`, `inizio`, `fine`) VALUES (NEW.conducente, TIMESTAMP(NEW.giorno, part), TIMESTAMP(NEW.giorno,arr));
	END IF;
END$$
```