Laboratorio di Basi di Dati

Squadra I

Gruppo A2

Progettazione Logica

Autori
Mariagiovanna Rotundo
Leonardo Vona

Docente
Giovanna Rosone

3 dicembre 2019

Versione 2.5

1. Traduzione schema da concettuale a logico

L'attività di traduzione dello schema concettuale verso quello logico prevede alcuni passi, eseguiti in ordine, che sono stati di seguito elencati:

1. Identificazione delle chiavi primarie;
2. Traduzione delle entità non coinvolte in gerarchie;
3. Traduzione delle gerarchie;
4. Traduzione degli attributi multivalore;
5. Traduzione delle associazioni molti a molti;
6. Traduzione delle associazioni uno a molti;
7. Traduzione delle associazioni uno a uno.

Legenda componenti grafici


Indica una chiave primaria.


Indica una chiave primaria che è anche chiave esterna.


Indica una chiave esterna.


Indica una chiave esterna con il vincolo NOT NULL.


Indica un attributo generico.


Indica un attributo generico con il vincolo NOT NULL.

1.1. Identificazione delle chiavi primarie

Una chiave primaria, per essere definita tale, deve soddisfare alcune proprietà, quali unicità all'interno della tabella e minimalità (la chiave deve coinvolgere il minimo numero di attributi possibile, garantendo comunque la sua unicità).

Con lo scopo di adottare una chiave primaria uniforme tra le varie tabelle, si è deciso di aggiungere ad ogni entità un ulteriore attributo di tipo intero, denominato "pk_NomeEntità", che viene gestito ed auto incrementato dal sistema.

Fanno eccezione a questa regola due casi:

- Entità figlie all'interno di una gerarchia: in questo caso la chiave primaria corrisponde alla chiave esterna che relaziona l'entità figlia con quella padre.
- Entità nate da associazioni molti a molti: in questo caso la chiave primaria è composta dalle chiavi esterne che relazionano l'entità nata dall'associazione con le entità associate.

1.2. Creazione delle sequenze

Per poter assegnare una chiave univoca nella fase di inserimento dei dati all'interno delle tabelle si utilizza una SEQUENCE per ogni chiave primaria che non sia anche chiave esterna all'interno della stessa tabella. Per ogni sequenza il valore di partenza e il passo sono 1.

La verifica che la chiave da inserire sia univoca deve essere comunque effettuata esplicitamente.

Non è stato impostato un valore massimo per le SEQUENCE, perciò il valore limite che può assumere la sequenza è pari a NOMAXVALUE, che corrisponde a 10^{28} - 1.

Poiché il valore massimo della sequenza è molto alto e verosimilmente irraggiungibile nel sistema che si sta progettando, si prevede che le SEQUENCE non siano cicliche.

1.2. Traduzione delle entità non coinvolte in gerarchie

Per le entità non coinvolte in gerarchie si è proceduto con una semplice traduzione, ovvero con il riportare gli attributi elencati nello schema concettuale e l'identificazione del corretto tipo di dato per ogni attributo.

Entità non coinvolte in gerarchie:

- Abbonamenti
- Aree
- Box
- CodiciSconto
- Colonne
- Notifiche
- ParcheggiAutomatici
- Sanzioni
- Sedi
- StipendiErogati
- TipiAbbonamento
- TipiAssicurazione
- Turni
- TipiTurno
- Veicoli

1.3. Introduzione entità Utenti

Per motivi di efficienza e migliore gestione delle sessioni, si è deciso, in collaborazione con il gruppo dei sistemisti, di aggiungere una nuova entità all'interno dello schema logico denominata Utenti.

Tale entità fa riferimento a Persone ed ogni persona può avere 1 o 2 utenze (2 nel caso in cui una persona sia dipendente e anche cliente).

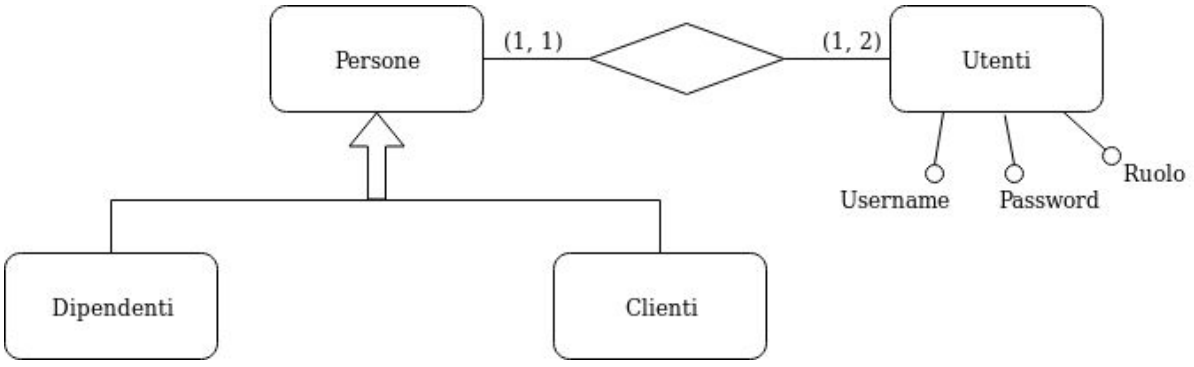
Lo username e la password sono stati spostati dalle entità Dipendenti e Clienti nella entità Utenti.

In aggiunta è presente anche un attributo ruolo (sempre per motivi di efficienza) di tipo INT, che mappa i possibili ruoli all'interno del sistema come segue:

- 1: Operatore
- 2: Amministratore
- 3: Responsabile
- 4: SuperUser
- 5: Cliente

Nota bene: l'attributo ruolo deve essere consistente con il ruolo effettivo della persona.
Nota bene 2: deve essere sempre garantito che per ogni persona ci possono essere al massimo due utenti.

Visione concettuale modificata:



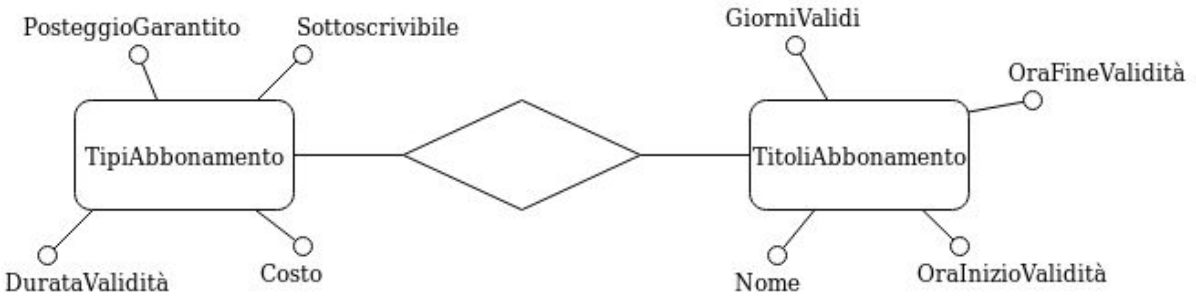
1.4. Introduzione entità TitoliAbbonamento

Per motivi di efficienza delle operazioni da effettuare, si è deciso, in collaborazione con il gruppo di caricamento dati, di aggiungere una nuova entità all'interno dello schema logico denominata TitoliAbbonamento..

Tale entità nasce da una suddivisione dell'entità TipiAbbonamento, separando il nome dell'abbonamento e dei giorni validi rispetto al resto degli attributi.

L'attributo Nome è di tipo char(1), quindi è un solo carattere che rappresenta il nome del tipo di abbonamento. Attualmente si prevedono i tipi di abbonamento A, B e C (come descritti in house of cars). Data la flessibilità della soluzione è comunque possibile aggiungere facilmente in futuro nuovi tipi di abbonamento.

Visione concettuale modificata:



1.5. Traduzione delle gerarchie

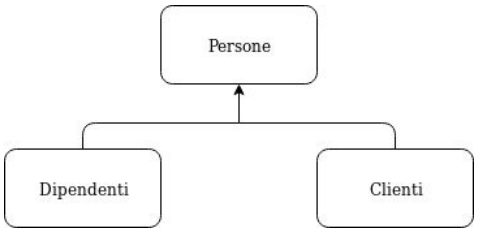
Relativamente alla traduzione delle gerarchie è possibile scegliere fra tre alternative:

- Traduzione della sola entità padre, accorpando le figlie nel padre.
- Traduzione delle sole entità figlie, accorpando il padre nelle figlie.
- Tradurre sia l'entità padre che le entità figlie, collegandole con delle chiavi esterne.

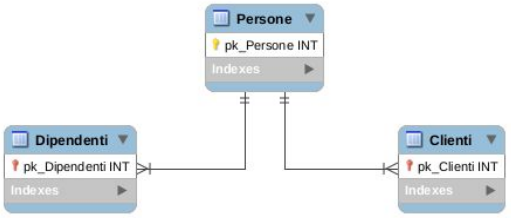
Si è scelto di tradurre sia l'entità padre che la/le entità figlia/e con lo scopo di mantenere la struttura più uniforme e leggibile possibile, oltre che a garantire flessibilità in caso di future modifiche. Le entità figlie, come già definito nel paragrafo 1.1, possiedono una chiave esterna al padre, che svolge anche la funzione di chiave primaria.

Nello schema concettuale sono presenti tre gerarchie.

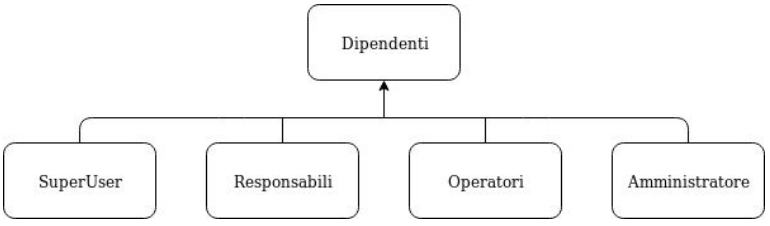
Gerarchia Persone - Dipendenti, Clienti



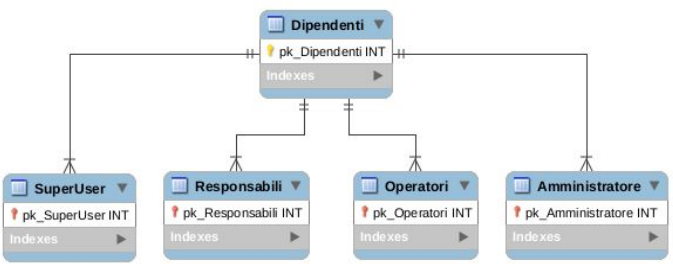
Che è stata tradotta in:



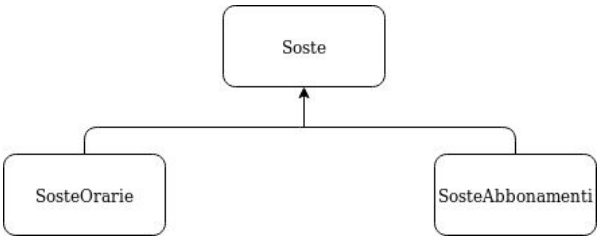
Gerarchia Dipendenti - Superuser, Responsabili, Operatori, Amministratore



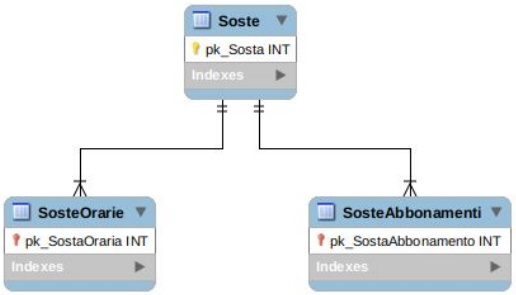
Che è stata tradotta in:



Gerarchia Dipendenti - Superuser, Responsabili, Operatori, Amministratore



Che è stata tradotta in:

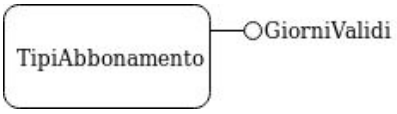


1.6. Traduzione degli attributi multivalore

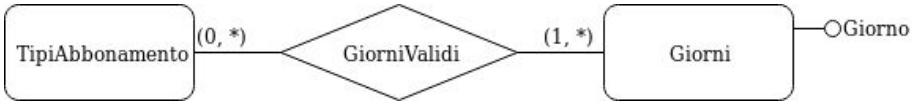
Per la traduzione degli attributi multivalore si è scelto di creare una nuova entità distinta per l'attributo e di collegarla all'entità originale con una associazione.

Attributo GiorniValidi in TipiAbbonamento

Nell'entità TIpiAbbonamento è presente l'attributo GiorniValidi, che indica a seconda del tipo di abbonamento, i giorni della settimana nei quali l'abbonamento è valido.



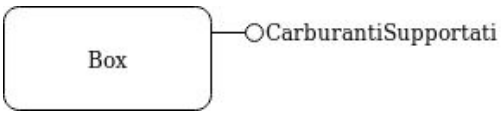
Che è stato modificato in:



Che è stato tradotto in:

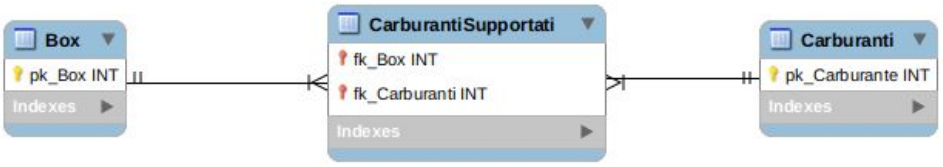
Attributo CarburantiSupportati in Box

Nell'entità Box è presente l'attributo CarburantiSupportati, che indica l'insieme dei tipi di alimentazione dei veicoli supportati per un dato Box.



Che è stato modificato in:

Che è stato tradotto in:



1.7. Traduzione delle associazioni molti a molti

Le associazioni molti a molti richiedono di essere tradotte in una nuova entità collegata alle entità associate con delle chiavi esterne, le quali svolgono anche la funzione di chiave primaria, come già affermato nel paragrafo 1.1.

Nel caso in cui le chiavi esterne non garantiscano l'unicità all'interno dell'entità, è necessario estendere la chiave primaria anche ad almeno uno degli attributi dell'associazione molti a molti.

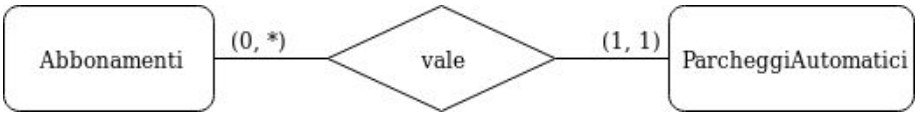
Associazione "usa" tra Clienti e Veicoli

Che è stata tradotta in:

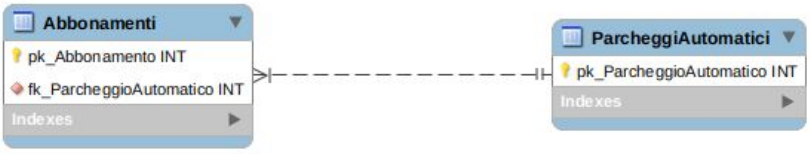
1.8. Traduzione delle associazioni uno a molti

L'attività di traduzione delle associazioni uno a molti prevede l'inserimento di una chiave esterna nell'entità "lato molti".

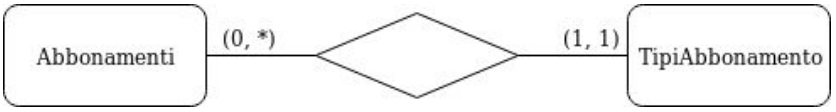
Associazione "vale" tra Abbonamenti e ParcheggiAutomatici



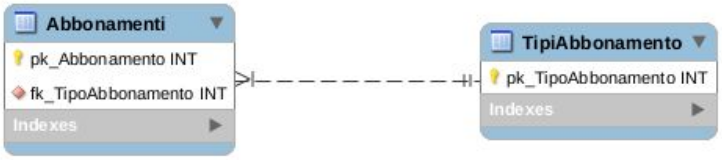
Traduzione:



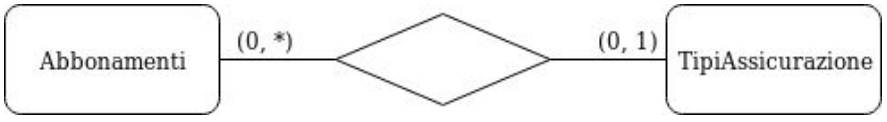
Associazione tra Abbonamenti e TipiAbbonamento



Traduzione:



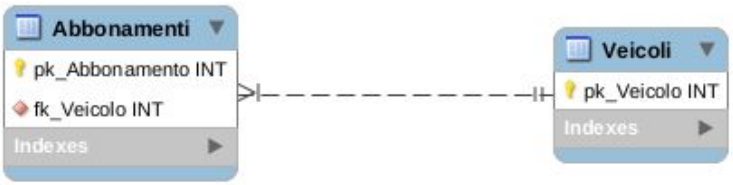
Associazione tra Abbonamenti e TipiAssicurazione



Traduzione:

Associazione “associato” tra Abbonamenti e Veicoli

Traduzione:



Associazione “raggruppato” tra Box e Aree

Traduzione:

Associazione “appartiene” tra Box e Colonne

Traduzione:

Associazione “fornisce” tra CodiciSconto e ParcheggiAutomatici

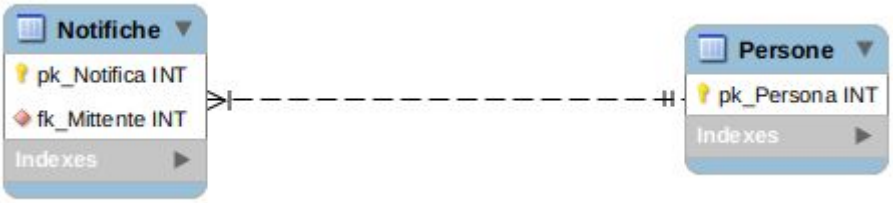
Traduzione:

Associazione “composto” tra Colonne e ParcheggiAutomatici

Traduzione:

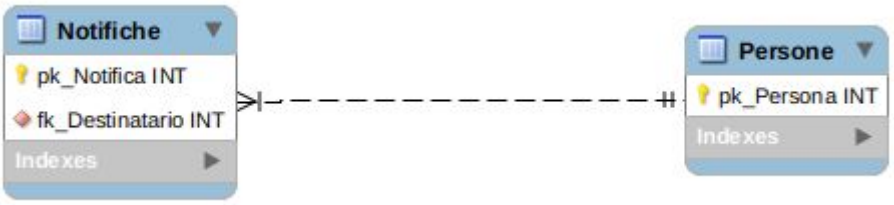
Associazione “manda” tra Notifiche e Persone

Traduzione:



Associazione “riceve” tra Notifiche e Persone

Traduzione:

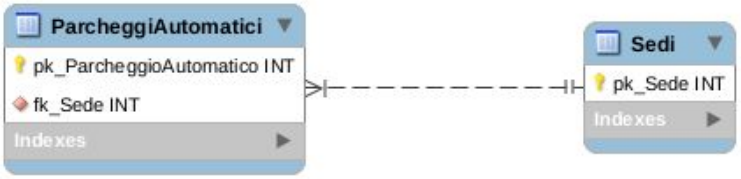


Associazione “lavora” tra Operatori e ParcheggiAutomatici

Traduzione:

Associazione “associato” tra ParcheggiAutomatici e Sedi

Traduzione:



Associazione “gestisce” tra Responsabili e Sedi

Traduzione:

Associazione “segnala” tra Sanzioni e Operatori

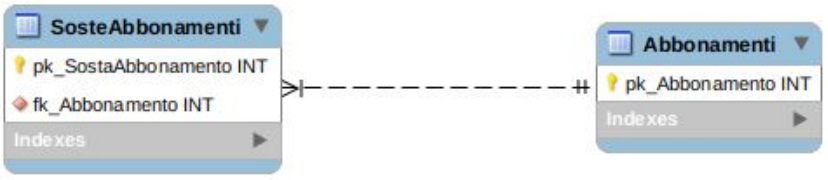
Traduzione:

Associazione “assegnata” tra Sanzioni e Veicoli

Traduzione:

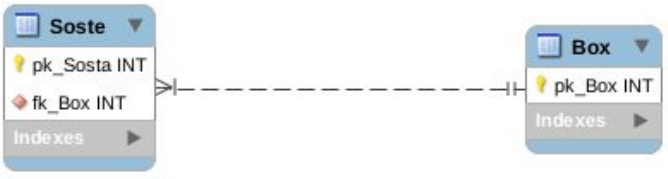
Associazione “usa” tra SosteAbbonamenti e Abbonamenti

Traduzione:



Associazione “assegnato” tra Soste e Box

Traduzione:



Associazione “effettua” tra Soste e Veicoli

Traduzione:

Associazione “ottiene” tra StipendiErogati e Dipendenti

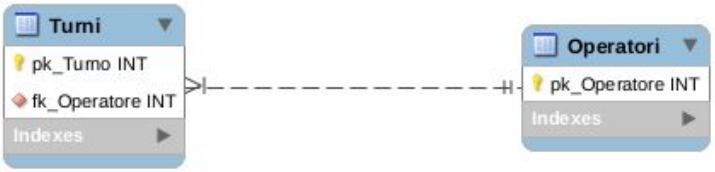
Traduzione:

Associazione “relativo” tra TipiAbbonamento e Aree

Traduzione:

Associazione “ricopre” tra Turni e Operatori

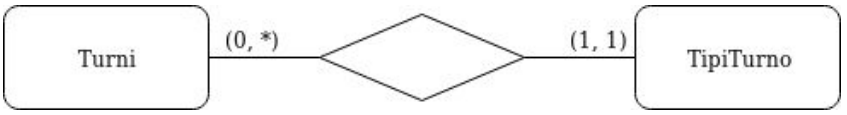
Traduzione:



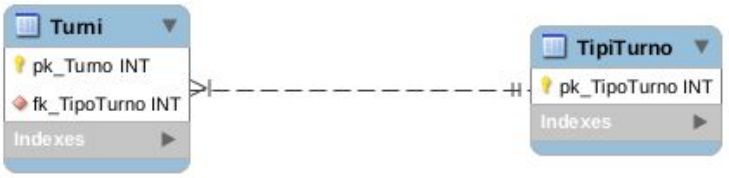
Associazione “prevede” tra Turni e ParcheggiAutomatici

Traduzione:

Associazione tra Turni e TipiTurno

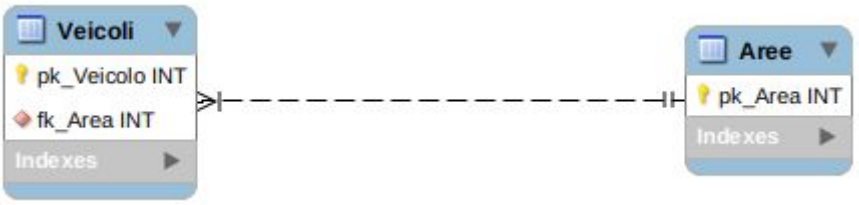


Traduzione:

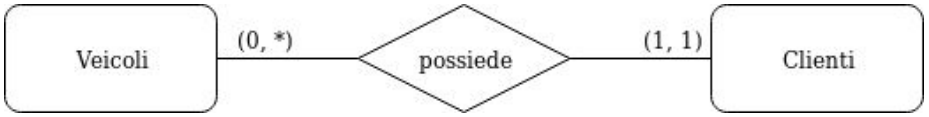


Associazione “appartiene” tra Veicoli e Aree

Traduzione:



Associazione “possiede” tra Veicoli e Clienti



Traduzione:

1.9. Traduzione delle associazioni uno a uno

Per le associazioni uno a uno ci sono due scelte a disposizione:

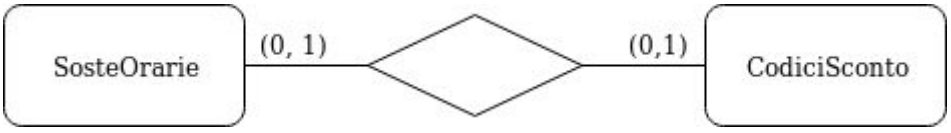
- definire una chiave esterna nell'entità più opportuna.
- definire una nuova entità dall'associazione, collegata tramite chiavi esterne alle entità associate; anche in questo caso, le chiavi esterne svolgono anche la funzione di chiave primaria.

Si è deciso, in generale, di utilizzare la prima scelta e quindi di collegare le due entità tramite una chiave esterna inserita nell'entità ritenuta più opportuna.

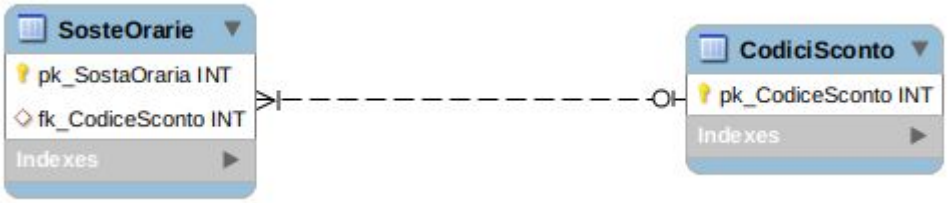
Associazione tra Abbonamenti e CodiciSconto

Traduzione:

Associazione tra SosteOrarie e CodiciSconto



Traduzione:



1.10. Determinazione dei tipi di dato

Dato che in Oracle PL/SQL non è definito un tipo standard boolean per gli attributi di una tabella, si è deciso di convertire gli attributi booleani in attributi di tipo CHAR di dimensione 1 vincolando i valori accettati a 0,1.
La scelta è ricaduta sul tipo CHAR perché richiede minor spazio rispetto ad altri tipi come NUMBER o SMALLINT.

Gli attributi che rappresentano una data e un'ora sono stati tradotti in un attributo di tipo TIMESTAMP.

Gli attributi che rappresentano un orario in ore e minuti sono stati tradotti come attributi di tipo TIMESTAMP, dove le informazioni ulteriori alle ore ed ai minuti non sono da considerare.

Gli attributi che rappresentano uno specifico giorno senza l'orario sono stati tradotti come attributi di tipo DATE, dove le informazioni aggiuntive rispetto a giorno, mese e anno non sono da considerare.

2. Schema Relazionale

Legenda

- chiave_primaria: le chiavi primarie sono indicate con attributi sottolineati.
- chiave_esterna*: le chiavi esterne sono indicate con una * alla destra dell'attributo.
- PK: primary key
- NN: not null
- U: unique
- FK: foreign key

2.1. Persone

`Persone(pk_Persona, CodiceFiscale, Nome, Cognome, Indirizzo, Citta, DataNascita, LuogoNascita, Sesso, Telefono, Email)`

- pk_Persona: chiave primaria, intero progressivo;
- CodiceFiscale: stringa che indica il codice fiscale di una persona;
- Nome: stringa che indica il nome di una persona;
- Cognome: stringa che indica il cognome di una persona;
- Indirizzo: stringa che indica un indirizzo;
- Citta: stringa che indica una città;
- DataNascita: data di nascita di una persona;
- LuogoNascita: stringa che indica il luogo di nascita di una persona;
- Sesso: carattere che indica il sesso di una persona;
- Telefono: stringa che rappresenta un numero di telefono;
- Email: stringa che rappresenta un'email.

NOME	TIPO	PK	NN	U	FK
pk_Persona	INT	X	X	X	
CodiceFiscale	CHAR(16)		X	X	
Nome	VARCHAR(45)		X		
Cognome	VARCHAR(45)		X		
Indirizzo	VARCHAR(60)		X		
Citta	VARCHAR(45)		X		
DataNascita	DATE		X		
LuogoNascita	VARCHAR(45)		X		
Sesso	CHAR(1)		X		
Telefono	VARCHAR(45)		X		
Email	VARCHAR(70)		X		

Vincoli

- Sesso può assumere uno dei valori tra 'M','F','A';
- Email ha il formato '%@%.%' dove '%' è una qualsiasi stringa.

2.2. Dipendenti

`Dipendenti(pk_Dipendente*, Assunzione, Licenziamento, CodiceIBAN)`

- pk_Dipendente: chiave primaria ed esterna che riferisce la chiave primaria di Persone. Intero progressivo;
- Assunzione: data che indica quando il dipendente è stato assunto;
- Licenziamento:data che indica quando il dipendente è stato licenziato;
- CodiceIBAN: stringa che rappresenta un codice IBAN.

NOME	TIPO	PK	NN	U	FK
pk_Dipendente	INT	X	X	X	X
Assunzione	DATE		X		
Licenziamento	DATE				
CodiceIBAN	VARCHAR(30)		X		

Vincoli

- Licenziamento >= Assunzione.

2.3. StipendiErogati

`StipendiErogati(pk_StipendioErogato, Data, Importo, fk_Dipendente*)`

- pk_StipendioErogato: chiave primaria, intero progressivo;
- Data: data in cui viene erogato lo stipendio;
- Importo: numero che rappresenta l'importo dello stipendio;
- fk_Dipendente: chiave esterna, riferisce la chiave primaria di Dipendenti.

NOME	TIPO	PK	NN	U	FK
pk_StipendioErogato	INT	X	X	X	
Data	DATE		X		
Importo	FLOAT		X		
fk_Dipendente	INT		X		X

Vincoli

- Importo>0.

Nota bene L'inserimento dei valori nella tabella StipendiErogati deve essere effettuato attraverso un trigger ogni giorno 1 del mese, valutando eventuali assenze o straordinari dalla tabella Turni.

2.4. SuperUser

`SuperUser(pk_SuperUser*,StipendioMensile)`

- pk_SuperUser: chiave primaria ed esterna che riferisce la chiave primaria di Dipendenti. Intero progressivo;
- StipendioMensile: numero che indica lo stipendio mensile del SuperUser.

NOME	TIPO	PK	NN	U	FK
pk_SuperUser	INT	X	X	X	X
StipendioMensile	FLOAT		X		

Vincoli

- StipendioMensile>0.

2.5. Amministratori

`Amministratori(pk_Amministratore*,StipendioMensile)`

- pk_Amministratore: chiave primaria ed esterna che riferisce la chiave primaria di Dipendenti. Intero progressivo;
- StipendioMensile: numero che indica lo stipendio mensile dell'amministatore.

NOME	TIPO	PK	NN	U	FK
pk_Amministratore	INT	X	X	X	X
StipendioMensile	FLOAT		X		

Vincoli

- StipendioMensile>0.

2.6. Responsabili

`Responsabili(pk_Responsabile*, StipendioMensile, fk_Sede*)`

- pk_Responsabile: chiave primaria ed esterna che riferisce la chiave primaria di Dipendenti. Intero progressivo;
- StipendioMensile: numero che indica lo stipendio mensile del responsabile.
- fk_Sede: chiave esterna che riferisce la chiave primaria di Sedi. Intero Progressivo. Indica la sede di cui è responsabile.

NOME	TIPO	PK	NN	U	FK
pk_Responsabile	INT	X	X	X	X
StipendioMensile	FLOAT		X		
fk_Sede	INT		X		X

Vincoli

- StipendioMensile>0.

2.7. Clienti

`Clienti(pk_Cliente*, BlackList, Stato)`

- pk_Cliente: chiave primaria ed esterna che riferisce la chiave primaria di Persone. Intero progressivo;
- BlackList: stringa che indica se un cliente è in lista nera oppure no;
- Stato : intero che indica se l'account del cliente è attivo oppure no.

NOME	TIPO	PK	NN	U	FK
pk_Clienti	INT	X	X	X	X
BlackList	CHAR(1)		X		
Stato	CHAR(1)		X		

Vincoli

- BlackList può assumere come valore 0 (non in blacklist) oppure 1 (in blacklist);
- Stato può assumere come valore 0 (disattivo) oppure 1 (attivo).

2.8. Utenti

`Utenti(pk_Utente, UserName, Password, Ruolo, Stato, fk_Persona*)`

- pk_Utente: chiave primaria, intero progressivo;
- UserName: stringa che rappresenta un username;
- Password: stringa che rappresenta una password;
- Ruolo: numero che indica il ruolo dell'utente associato a quell'Utente;
- Stato: numero che indica lo stato dell'utente;
- fk_Persona: chiave esterna, riferisce la chiave primaria di Persone.

NOME	TIPO	PK	NN	U	FK
pk_Utente	INT	X	X	X	
UserName	VARCHAR(45)		X	X	
Password	VARCHAR(45)		X		
Ruolo	INT		X		
Stato	INT		X		
fk_Persona	INT		X		X

Vincoli

- Valori accettati di Ruolo: 1, 2, 3, 4, 5;
- All'interno della tabella ci possono essere al massimo due tuple con fk_Persona uguale. In tal caso, solo una delle due tuple può e deve avere Ruolo = 0.
- Stato può assumere i valori 0 o 1;

2.9. Veicoli

`Veicoli(pk_Veicolo, Targa, Larghezza, Lunghezza, Altezza, Peso, TipoCarburante, Modello, Cancellato, fk_Proprietario*, fk_Area*)`

- pk_Veicolo: chiave primaria, intero progressivo;
- Targa: stringa che rappresenta la targa di un veicolo;
- Larghezza: numero che indica la larghezza del veicolo in metri;
- Lunghezza: numero che indica la lunghezza del veicolo in metri;
- Altezza: numero che indica l'altezza del veicolo in metri;
- Peso: numero che indica il peso del veicolo in kg;
- TipoCarburante: stringa che indica il tipo di carburante del veicolo;
- Modello: stringa che indica il modello del veicolo;
- Cancellato: numero che indica se il veicolo non può più essere selezionato dal cliente;
- fk_Proprietario: chiave esterna, riferisce la chiave primaria di Clienti.
- fk_Area: chiave esterna, riferisce la chiave primaria di Aree.

NOME	TIPO	PK	NN	U	FK
pk_Veicolo	INT	X	X	X	
Targa	VARCHAR(10)		X	X	
Larghezza	FLOAT		X		
Lunghezza	FLOAT		X		
Altezza	FLOAT		X		
Peso	FLOAT		X		
TipoCarburante	VARCHAR(45)		X		
Modello	VARCHAR(45)				
Cancellato	CHAR(1)		X		
fk_Proprietario	INT		X		X
fk_Area	INT		X		X

Vincoli

- Larghezza > 0;
- Lunghezza >0;
- Altezza >0;
- Peso >0.
- Cancellato può assumere come valore 0 (non cancellato) oppure 1 (cancellato).

2.10. ClientiVeicoli

`ClientiVeicoli(fk_Cliente*, fk_Veicolo*)`

- fk_Cliente: chiave esterna che riferisce la chiave primaria di Clienti. Costituisce, insieme a fk_Veicolo, la chiave primaria. Intero progressivo;
- fk_Veicolo: chiave esterna che riferisce la chiave primaria di Veicoli. Costituisce, insieme a fk_Cliente, la chiave primaria. Intero progressivo.

NOME	TIPO	PK	NN	U	FK
pk_Cliente	INT	X	X		X
pk_Veicolo	INT	X	X		X

2.11. TipiAssicurazione

`TipiAssicurazione(pk_TipoAssicurazione, DanniCoperti, MassimaleCoperto, Costo, AgenziaAssicurativaErogatrice, Stipulabile)`

- pk_TipoAssicurazione: chiave primaria, intero progressivo;
- DanniCoperti: stringa che rappresenta i danni coperti dal tipo di assicurazione;
- MassimaleCoperto: intero che rappresenta il massimale coperto dal tipo di assicurazione;
- Costo: float che indica il costo di un'assicurazione di un certo tipo;
- AgenziaAssicurativaErogatrice: stringa che rappresenta l'agenzia assicurativa che eroga il servizio;
- Stipulabile: booleano che indica se l'assicurazione è stipulabile o meno.

NOME	TIPO	PK	NN	U	FK
pk_TipoAssicurazione	INT	X	X	X	
DanniCoperti	VARCHAR(100)		X		
MassimaleCoperto	INT		X		
Costo	FLOAT		X		
AgenziaAssicurativaErogatrice	VARCHAR(45)				
Stipulabile	CHAR(1)		X		

Vincoli

- MassimaleCoperto >0;
- Costo >0;
- Stipulabile può assumere come valore 0 (non stipulabile) o 1 (stipulabile).

2.12. Aree

`Aree(pk_Area, NomeArea, PesoSostenibile, TariffaOraria, LarghezzaMax, LunghezzaMax, AltezzaMax)`

- pk_Area: chiave primaria, intero progressivo;
- NomeArea: stringa che indica il nome di un'area;
- PesoSostenibile: numero che indica il peso in kg permesso ai veicoli in un'area;
- TariffaOraria: numero che indica la tariffa oraria di quell'area;
- LarghezzaMax: numero che indica la massima larghezza dei veicoli contenuti;
- LunghezzaMax: numero che indica la massima lunghezza dei veicoli contenuti;
- AltezzaMax: numero che indica la massima altezza dei veicoli contenuti.

NOME	TIPO	PK	NN	U	FK
pk_Area	INT	X	X	X	
NomeArea	VARCHAR(45)		X	X	
PesoSostenibile	FLOAT		X		
TariffaOraria	FLOAT		X		
LarghezzaMax	FLOAT		X		
LunghezzaMax	FLOAT		X		
AltezzaMax	FLOAT		X		

Vincoli

- PesoSostenibile > 0;
- TariffaOraria > 0;
- LarghezzaMax >0;
- LunghezzaMax > 0;
- AltezzaMax >0.

2.13. Sedi

`Sedi(pk_Sede, Indirizzo, Citta, Telefono, Stato)`

- pk_Sede: chiave primaria, intero progressivo;
- Indirizzo: stringa che rappresenta l'indirizzo della sede;
- Citta: stringa che rappresenta la città della sede;
- Telefono: stringa che rappresenta un numero di telefono.
- Stato: numero che indica lo stato della sede

NOME	TIPO	PK	NN	U	FK
pk_Sede	INT	X	X	X	
Indirizzo	VARCHAR(60)		X		
Citta	VARCHAR(45)		X		
Telefono	VARCHAR(45)		X		
Stato	CHAR(1)		X		

Vincoli

- Stato può assumere come valore 0 (disattiva) oppure 1 (attiva).

2.14. TitoliAbbonamento

`TitoliAbbonamento(pk_TitoloAbbonamento, Nome, OraInizioValidita, OraFineValidita)`

- pk_TitoloAbbonamento: chiave primaria, intero progressivo;
- Nome: carattere che identifica il tipo di abbonamento, univoco;
- OraInizioValidita: ora e minuti relativi all'inizio della validità;
- OraFineValidita: ora e minuti relativi alla fine della validità.

NOME	TIPO	PK	NN	U	FK
pk_TitoloAbbonamento	INT	X	X	X	
Nome	CHAR(1)		X	X	
OraInizioValidita	TIMESTAMP		X		
OraFineValidita	TIMESTAMP		X		

2.15. TipiAbbonamento

`TipiAbbonamento(pk_TipoAbbonamento, DurataValidita, Costo, PosteggioGarantito, Sottoscrivibile, fk_Area*, fk_TitoloAbbonamento*)`

- pk_TipoAbbonamento: chiave primaria, intero progressivo;
- DurataValidita: numero che indica la durata della validità (in settimane);
- Costo: numero che indica il costo di un abbonamento di un certo tipo;
- PosteggioGarantito: booleano per l'opzione plus;
- Sottoscrivibile : numero che indica se possono essere fatti abbonamenti di quel tipo;
- fk_Area: chiave esterna, riferisce la chiave primaria di Aree;
- fk_TitoloAbbonamento: chiave esterna, riferisce la chiave primaria di TitoliAbbonamento.

NOME	TIPO	PK	NN	U	FK
pk_TipoAbbonamento	INT	X	X	X	
DurataValidita	INT		X		
Costo	FLOAT		X		
PosteggioGarantito	CHAR(1)		X		
Sottoscrivibile	CHAR(1)		X		
fk_Area	INT		X		X
fk_TitoloAbbonamento	INT				X

Vincoli

- Costo >0;
- PosteggioGarantito può assumere uno dei valori tra 0 (non garantito) e 1 (garantito).
- Sottoscrivibile può assumere come valore 0 (non sottoscrivibile) oppure 1 (sottoscrivibile).

2.16. Abbonamenti

```
Abbonamenti(pk_Abbonamento, DataInizioValidita, DataFineValidita,
PrezzoPagato,TipoPagamento ,fk_Veicolo*, fk_TipoAssicurazione*,
fk_TipoAbbonamento*, fk_ParcheggioAutomatico*,fk_CodiceSconto*)
```

- pk_Abbonamento: chiave primaria, intero progressivo;
- DataInizioValidita: data in cui inizia ad essere valido l'abbonamento;
- DataFineValidita: data in cui smette di essere valido l'abbonamento;
- PrezzoPagato: numero che indica il prezzo pagato per l'abbonamento;
- TipoPagamento: stringa che indica il tipo di pagamento utilizzato;
- fk_Veicolo: chiave esterna, riferisce la chiave primaria di Veicoli;
- fk_TipoAssicurazione: chiave esterna, riferisce la chiave primaria di TipiAssicurazione;
- fk_TipoAbbonamento: chiave esterna, riferisce la chiave primaria di TipiAbbonamento.
- fk_ParcheggioAutomatico: chiave esterna, riferisce la chiave primaria di ParcheggiAutomatici;
- fk_CodiceSconto: chiave esterna, riferisce la chiave primaria di CodiciSconto;

NOME	TIPO	PK	NN	U	FK
pk_Abbonamento	INT	X	X	X	
DataInizioValidita	DATE		X		
DataFineValidita	DATE		X		
PrezzoPagato	FLOAT		X		
TipoPagamento	VARCHAR(45)		X		
fk_Veicolo	INT		X		X
fk_TipoAssicurazione	INT				X
fk_TipoAbbonamento	INT		X		X
fk_ParcheggioAutomatico	INT		X		X
fk_CodiceSconto	INT				X

Vincoli

- PrezzoPagato, se non nullo, dev'essere >= 0;
- DataFineValidita > DataInizioValidita.

2.17. ParcheggiAutomatici

`ParcheggiAutomatici(pk_ParcheggioAutomatico, Citta, Indirizzo, AssicurazioneParcheggio, Telefono, Zona, MaggiorazioneZona, Stato, fk_Sede*)`

- pk_ParcheggioAutomatico: chiave primaria, intero progressivo;
- Citta: stringa che rappresenta una città;
- Indirizzo: stringa che rappresenta un indirizzo;
- AssicurazioneParcheggio: stringa per l'assicurazione del parcheggio;
- Telefono: stringa che rappresenta un numero di telefono;
- Zona: numero che rappresenta la zona;
- MaggiorazioneZona: numero che indica la maggiorazione dovuta alla zona;
- Stato: stringa che rappresenta lo stato del parcheggio;
- fk_Sede: chiave esterna, riferisce la chiave primaria di Sedi.

NOME	TIPO	PK	NN	U	FK
pk_ParcheggioAutomatico	INT	X	X	X	
Citta	VARCHAR(45)		X		
Indirizzo	VARCHAR(60)		X		
AssicurazioneParcheggio	VARCHAR(45)				
Telefono	VARCHAR(45)		X		
Zona	INT		X		
MaggiorazioneZona	FLOAT		X		
Stato	CHAR(1)		X		
fk_Sede	INT		X		X

Vincoli

- Stato può assumere come valore 0 (disattivo) oppure 1 (attivo).
- La coppia Citta e indirizzo è unica

2.18. Operatori

`Operatori(pk_Operatore*, fk_ParcheggioAutomatico*)`

- pk_Operatore: chiave primaria ed esterna che riferisce la chiave primaria di Dipendenti. Intero progressivo;
- fk_ParcheggioAutomatico: chiave esterna, riferisce la chiave primaria di ParcheggiAutomatici.

NOME	TIPO	PK	NN	U	FK
pk_Operatore	INT	X	X	X	X
fk_ParcheggioAutomatico	INT		X		X

2.19. CodiciSconto

`CodiciSconto(pk_CodiceSconto, Codice, Sconto, DataScadenza, fk_ParcheggioAutomatico*)`

- pk_CodiceSconto: chiave primaria, intero progressivo;
- Codice: stringa che indica il codice del codice sconto;
- Sconto: numero per indicare lo sconto in denaro;
- DataScadenza: data di scadenza del codice sconto, se è NULL il codice non ha scadenza;
- fk_ParcheggioAutomatico: chiave esterna, riferisce la chiave primaria di ParcheggiAutomatici.

NOME	TIPO	PK	NN	U	FK
pk_CodiceSconto	INT	X	X	X	
Codice	VARCHAR(15)		X	X	
Sconto	FLOAT		X		
DataScadenza	DATE		X		
fk_ParcheggioAutomatico	INT		X		X

Vincoli

- Sconto > 0.

2.20. TipiTurno

`TipiTurno(pk_TipoTurno, Nome, OraInizio, OraFine, RetribuzioneOraria)`

- pk_TipoTurno: chiave primaria, intero progressivo;
- Nome: stringa che rappresenta il nome del tipo di turno;
- OraInizio: ora e minuti di inizio del tipo di turno;
- OraFine: ora e minuti di fine del tipo di turno;
- RetribuzioneOraria: numero che indica la retribuzione oraria per il tipo di turno.

NOME	TIPO	PK	NN	U	FK
pk_TipoTurno	INT	X	X	X	
Nome	VARCHAR(45)		X	X	
OraInizio	TIMESTAMP		X		
OraFine	TIMESTAMP		X		
RetribuzioneOraria	FLOAT		X		

Vincoli

- RetribuzioneOraria >0.

2.21. Sanzioni

`Sanzioni(pk_Sanzione, MotivoSanzione, Rilevamento, Costo, StatoPagamento, DataEmissione, DataScadenza, fk_Operatore*, fk_Veicolo*)`

- pk_Sanzione: chiave primaria, intero progressivo;
- MotivoSanzione: stringa per indicare il motivo della sanzione;
- Rilevamento: indica la data e l'ora del rilevamento;
- Costo: numero che rappresenta il costo della sanzione;
- StatoPagamento: booleano che indica lo stato del pagamento;
- DataEmissione: data di emissione della sanzione;
- DataScadenza: data di scadenza della sanzione;
- fk_Operatore: chiave esterna, riferisce la chiave primaria di Operatori;
- fk_Veicolo: chiave esterna, riferisce la chiave primaria di Veicoli.

NOME	TIPO	PK	NN	U	FK
pk_Sanzione	INT	X	X	X	
MotivoSanzione	VARCHAR(100)		X		
Rilevamento	DATE		X		
Costo	FLOAT		X		
StatoPagamento	CHAR(1)		X		
DataEmissione	DATE				
DataScadenza	DATE				
fk_Operatore	INT		X		X
fk_Veicolo	INT		X		X

Vincoli

- Costo >0;
- StatoPagamento può assumere come valore solo 0 (non pagata) oppure 1 (pagata);
- DataEmissione >= Rilevamento;
- DataScadenza >= DataEmissione.

2.22. Colonne

`Colonne(pk_Colonna, NumeroPianiSopraelevati, NumeroPianiSottoterra, Stato, fk_ParcheggioAutomatico*)`

- pk_Colonna: chiave primaria, intero progressivo;
- NumeroPianiSopraelevati: numero che indica il numero di piani della colonna sopra il livello del suolo;
- NumeroPianiSottoterra: numero che indica il numero di piani della colonna sotto il livello del suolo;
- Stato: stringa che indica lo stato della colonna;
- fk_ParcheggioAutomatico chiave esterna, riferisce la chiave primaria di ParcheggiAutomatici.

NOME	TIPO	PK	NN	U	FK
pk_Colonna	INT	X	X	X	
NumeroPianiSopraelevati	INT		X		
NumeroPianiSottoterra	INT		X		
Stato	VARCHAR(45)				
fk_ParcheggioAutomatico	INT		X		X

Vincoli

- NumeroPianiSopraelevati >= 0;
- NumeroPianiSottoterra >= 0;

2.23. GiorniValidi

`GiorniValidi(fk_Giorno*, fk_TitoloAbbonamento*)`

- fk_Giorno: chiave esterna che riferisce la chiave primaria di Giorni. Costituisce, insieme a fk_TitoloAbbonamento, la chiave primaria. Intero progressivo;
- fk_TitoloAbbonamento: chiave esterna che riferisce la chiave primaria di TitoloAbbonamento. Costituisce, insieme a fk_Giorno, la chiave primaria. Intero progressivo.

NOME	TIPO	PK	NN	U	FK
fk_Giorno	INT	X	X		X
fk_TitoloAbbonamento	INT	X	X		X

2.24. Box

`Box(pk_Box, Piano, Stato, fk_Area*, fk_Colonna*)`

- pk_Box: chiave primaria, intero progressivo;
- Piano: numero che indica il piano a cui si trova il box;
- Stato: stringa che indica lo stato del box;
- fk_Area: chiave esterna, riferisce la chiave primaria di Aree;
- fk_Colonna: chiave esterna, riferisce la chiave primaria di Colonne.

NOME	TIPO	PK	NN	U	FK
pk_Box	INT	X	X	X	
Piano	INT		X		
Stato	CHAR(1)		X		
fk_Area	INT		X		X
fk_Colonna	INT		X		X

Vincoli

- Stato può assumere come valore 0 (disattivo) oppure 1 (attivo).

2.25. Giorni

`Giorni(pk_Giorno, Giorno)`

- pk_Giorno: chiave primaria, intero progressivo;
- Giorno: stringa che rappresenta un giorno.

NOME	TIPO	PK	NN	U	FK
pk_Giorno	INT	X	X	X	
Giorno	VARCHAR(9)		X	X	

Vincoli

- Giorno può assumere uno tra i valori 'Lunedì', 'Martedì', 'Mercoledì', 'Giovedì', 'Venerdì', 'Sabato', 'Domenica', ‘Festivo’.

2.26. Turni

`Turni(pk_Turno, Data, Inizio, Fine, fk_ParcheggioAutomatico*, fk_Operatore*, fk_TipoTurno*)`

- pk_Turno: chiave primaria, intero progressivo;
- Data: data in cui è previsto l'inizio del turno;
- Inizio: data e ora di inizio effettivo del turno;
- Fine: data e ora di fine effettivo del turno;
- fk_ParcheggioAutomatico: chiave esterna, riferisce la chiave primaria di ParcheggiAutomatici;
- fk_Operatore: chiave esterna, riferisce la chiave primaria di Operatori;
- fk_TipoTurno: chiave esterna, riferisce la chiave primaria di TipiTurno.

NOME	TIPO	PK	NN	U	FK
pk_Turno	INT	X	X	X	
Data	DATE		X		
Inizio	TIMESTAMP				
Fine	TIMESTAMP				
fk_ParcheggioAutomatico	INT		X		X
fk_Operatore	INT		X		X
fk_TipoTurno	INT		X		X

Vincoli

- Fine deve essere successivo a Inizio.

Nota bene:

La tabella Turni viene redatta mensilmente dal responsabile a cui fa riferimento il parcheggio in cui lavora l'operatore, inserendo per ogni giorno in cui si prevede l'operatore debba lavorare la data e il tipo di turno (attraverso la chiave esterna fk_TipoTurno).

Quando un operatore effettuerà effettivamente un turno, gli orari di effettivo arrivo e uscita saranno inseriti nella tabella.

Se al momento in cui viene effettuato il trigger per l'assegnamento degli stipendi mensili un Turno dovesse avere Inizio e Fine a NULL, è da considerarsi come turno previsto ma non effettuato.

2.27. Soste

`Soste(pk_Sosta, Inizio, Fine, fk_Veicolo*, fk_Box*)`

- pk_Sosta: chiave primaria, intero progressivo;
- Inizio: data e ora di inizio sosta;
- Fine: data e ora di fine sosta;
- fk_Veicolo: chiave esterna, riferisce la chiave primaria di Veicoli;
- fk_Box: chiave esterna, riferisce la chiave primaria di Box;

NOME	TIPO	PK	NN	U	FK
pk_Sosta	INT	X	X	X	
Inizio	TIMESTAMP		X		
Fine	TIMESTAMP				
fk_Veicolo	INT		X		X
fk_Box	INT		X		X

Vincoli

- Fine deve essere successivo a Inizio;

2.28. SosteOrarie

`SosteOrarie(pk_SostaOraria*, TipoPagamento, PrezzoPagato, fk_CodiceSconto*)`

- pk_SostaOraria: chiave primaria, intero progressivo;
- TipoPagamento: stringa che indica il tipo di pagamento;
- PrezzoPagato: numero che rappresenta il prezzo pagato;
- fk_CodiceSconto: chiave esterna, riferisce la chiave primaria di CodiciSconto;

NOME	TIPO	PK	NN	U	FK
pk_SostaOraria	INT	X	X	X	X
TipoPagamento	VARCHAR(45)		X		
PrezzoPagato	FLOAT		X		
fk_CodiceSconto	INT				X

Vincoli

- PrezzoPagato >= 0.

2.29. SosteAbbonamenti

`SosteAbbonamenti(pk_SostaAbbonamento*, fk_Abbonamento*)`

- pk_SostaAbbonamento: chiave primaria, intero progressivo;
- fk_Abbonamento: chiave esterna, riferisce la chiave primaria di Abbonamenti.

NOME	TIPO	PK	NN	U	FK
pk_SostaAbbonamento	INT	X	X	X	X
fk_Abbonamento	INT		X		X

2.30. Notifiche

`Notifiche(pk_Notifica, Data, Descrizione, Tipo, fk_Mittente*, fk_Destinatario*)`

- pk_Notifica: chiave primaria, intero progressivo;
- Data: data in cui viene mandata la notifica;
- Descrizione: stringa che rappresenta la descrizione della notifica;
- Tipo: numero che indica il tipo della notifica;
- fk_Mittente: chiave esterna, riferisce la chiave primaria di Persone
- fk_Destinatario: chiave esterna, riferisce la chiave primaria di Persone.

NOME	TIPO	PK	NN	U	FK
pk_Notifica	INT	X	X	X	
Data	DATE		X		
Descrizione	VARCHAR(500)		X		
Tipo	INT		X		
fk_Mittente	INT		X		X
fk_Destinatario	INT		X		X

2.31. CarburantiSupportati

`CarburantiSupportati(fk_Box*, fk_Carburante*)`

- fk_Box: chiave esterna, riferisce la chiave primaria di Box;
- fk_Carburante: chiave esterna, riferisce la chiave primaria di Carburanti.

NOME	TIPO	PK	NN	U	FK
fk_Box	INT	X	X		X
fk_Carburante	INT	X	X		X

2.32. Carburanti

`Carburanti(pk_Carburante, Nome)`

- pk_Carburante: chiave primaria, intero progressivo;
- Nome: stringa che rappresenta il nome di un carburante.

NOME	TIPO	PK	NN	U	FK
pk_Carburante	INT	X	X	X	
Nome	VARCHAR(45)		X	X	

3. Script per la creazione delle tabelle

3.1. Persone

```
CREATE TABLE Persone(
pk_Persona INT PRIMARY KEY,
CodiceFiscale CHAR(16) UNIQUE NOT NULL,
Nome VARCHAR(45) NOT NULL,
Cognome VARCHAR(45) NOT NULL,
Indirizzo VARCHAR(60) NOT NULL,
Citta VARCHAR(45) NOT NULL,
DataNascita DATE NOT NULL,
LuogoNascita VARCHAR(45) NOT NULL,
Sesso CHAR(1) NOT NULL,
Telefono VARCHAR(45) NOT NULL,
Email VARCHAR(70) NOT NULL,
CONSTRAINT CHK_Persone_Email CHECK (Email LIKE '%@%.%'),
CONSTRAINT CHK_Persone_Sesso CHECK (Sesso IN ('M','F','A'))
);
```

3.2. Utenti

```
CREATE TABLE Utenti(
pk_Utente INT PRIMARY KEY,
UserName VARCHAR(45) NOT NULL UNIQUE,
Password VARCHAR(45) NOT NULL,
Ruolo INT NOT NULL,
Stato INT NOT NULL,
fk_Persona INT NOT NULL,
FOREIGN KEY(fk_Persona) REFERENCES Persone (pk_Persona),
CONSTRAINT CHK_Utenti_Ruolo CHECK (Ruolo IN (1,2,3,4,5)),
CONSTRAINT CHK_Utenti_Stato CHECK (Stato IN (0,1))
);
```

3.3. Dipendenti

```
CREATE TABLE Dipendenti(
pk_Dipendente INT PRIMARY KEY,
Assunzione DATE NOT NULL,
Licenziamento DATE,
CodiceIBAN VARCHAR(30) NOT NULL,
FOREIGN KEY(pk_Dipendente) REFERENCES Persone (pk_Persona),
CONSTRAINT CHK_Dipendenti_Licenziamento CHECK (Licenziamento>=Assunzione)
);
```

3.4. StipendiErogati

```
CREATE TABLE StipendiErogati(
pk_StipendioErogato INT PRIMARY KEY,
Data DATE NOT NULL,
Importo FLOAT NOT NULL,
fk_Dipendente INT NOT NULL,
FOREIGN KEY(fk_Dipendente) REFERENCES Dipendenti (pk_Dipendente),
CONSTRAINT CHK_StipendiErogati_Importo CHECK (Importo>0)
);
```

3.5. SuperUser

```
CREATE TABLE SuperUser(
pk_SuperUser INT PRIMARY KEY,
StipendioMensile FLOAT NOT NULL,
FOREIGN KEY(pk_SuperUser) REFERENCES Dipendenti (pk_Dipendente),
CONSTRAINT CHK_SuperUser_StipendioMensile  CHECK (StipendioMensile >0)
);
```

3.6. Amministratori

```
CREATE TABLE Amministratori(
pk_Amministratore INT PRIMARY KEY,
StipendioMensile FLOAT NOT NULL,
FOREIGN KEY(pk_Amministratore) REFERENCES Dipendenti (pk_Dipendente),
CONSTRAINT CHK_Amministratori_StipendioMensile  CHECK (StipendioMensile >0)
);
```

3.7. Clienti

```
CREATE TABLE Clienti(
pk_Cliente INT PRIMARY KEY,
BlackList CHAR(1) DEFAULT 0 NOT NULL,
Stato CHAR(1) DEFAULT 1 NOT NULL,
FOREIGN KEY(pk_Cliente) REFERENCES Persone (pk_Persona),
CONSTRAINT CHK_Clienti_BlackList CHECK(BlackList IN ('0','1')),
CONSTRAINT CHK_Clienti_Stato CHECK(Stato IN ('0','1'))
);
```

3.8. Aree

```
CREATE TABLE Aree(
pk_Area INT PRIMARY KEY,
NomeArea VARCHAR(45) UNIQUE NOT NULL,
PesoSostenibile FLOAT NOT NULL,
TariffaOraria FLOAT NOT NULL,
LarghezzaMax FLOAT NOT NULL,
LunghezzaMax FLOAT NOT NULL,
AltezzaMax FLOAT NOT NULL,
CONSTRAINT CHK_Aree_PesoSostenibile CHECK (PesoSostenibile>0),
CONSTRAINT CHK_Aree_TariffaOraria CHECK (TariffaOraria>0),
CONSTRAINT CHK_Aree_LarghezzaMax CHECK (LarghezzaMax>0),
CONSTRAINT CHK_Aree_LunghezzaMax CHECK (LunghezzaMax>0),
CONSTRAINT CHK_Aree_AltezzaMax CHECK (AltezzaMax>0)
);
```

3.9. Veicoli

```
CREATE TABLE Veicoli(
pk_Veicolo INT PRIMARY KEY,
Targa VARCHAR(10) NOT NULL UNIQUE,
Larghezza FLOAT NOT NULL,
Lunghezza FLOAT NOT NULL,
Altezza FLOAT NOT NULL,
Peso FLOAT NOT NULL,
TipoCarburante VARCHAR(45) NOT NULL,
Modello VARCHAR(45),
Cancellato CHAR(1) DEFAULT 0 NOT NULL,
fk_Proprietario INT NOT NULL,
fk_Area INT NOT NULL,
FOREIGN KEY(fk_Proprietario) REFERENCES Clienti (pk_Cliente),
FOREIGN KEY(fk_Area) REFERENCES Aree (pk_Area),
CONSTRAINT CHK_Veicoli_Larghezza CHECK (Larghezza>0),
CONSTRAINT CHK_Veicoli_Lunghezza CHECK (Lunghezza>0),
CONSTRAINT CHK_Veicoli_Altezza CHECK (Altezza>0),
CONSTRAINT CHK_Veicoli_Peso CHECK (Peso>0),
CONSTRAINT CHK_Veicoli_Cancellato CHECK(Cancellato IN ('0','1'))
);
```

3.10. ClientiVeicoli

```
CREATE TABLE ClientiVeicoli(
fk_Cliente INT,
fk_Veicolo INT,
FOREIGN KEY(fk_Cliente) REFERENCES Clienti (pk_Cliente),
FOREIGN KEY(fk_Veicolo) REFERENCES Veicoli (pk_Veicolo),
PRIMARY KEY(fk_Cliente, fk_Veicolo)
);
```

3.11. TipiAssicurazione

```
CREATE TABLE TipiAssicurazione(
pk_TipoAssicurazione INT PRIMARY KEY,
DanniCoperti VARCHAR(100) NOT NULL,
MassimaleCoperto INT NOT NULL,
Costo FLOAT NOT NULL,
AgenziaAssicurativaErogatrice VARCHAR(45),
Stipulabile CHAR(1) DEFAULT 1 NOT NULL ,
CONSTRAINT CHK_TipiAssicurazione_MassimaleCoperto CHECK (MassimaleCoperto>0),
CONSTRAINT CHK_TipiAssicurazione_Costo CHECK(Costo>0),
CONSTRAINT CHK_TipiAssicurazione_Stipulabile CHECK (Stipulabile IN ('0','1'))
);
```

3.12. TitoliAbbonamento

```
CREATE TABLE TitoliAbbonamento(
pk_TitoloAbbonamento INT PRIMARY KEY,
Nome CHAR(1) UNIQUE NOT NULL,
OraInizioValidita TIMESTAMP NOT NULL,
OraFineValidita TIMESTAMP NOT NULL
);
```

3.13. TipiAbbonamento

```
CREATE TABLE TipiAbbonamento(
pk_TipoAbbonamento INT PRIMARY KEY,
DurataValidita INT NOT NULL,
Costo FLOAT NOT NULL,
PosteggioGarantito CHAR(1) DEFAULT 0 NOT NULL,
Sottoscrivibile CHAR(1) DEFAULT 1 NOT NULL,
fk_Area INT NOT NULL,
fk_TitoloAbbonamento INT NOT NULL,
FOREIGN KEY(fk_Area) REFERENCES Aree (pk_Area),
FOREIGN KEY(fk_TitoloAbbonamento) REFERENCES TitoliAbbonamento
(pk_TitoloAbbonamento),
CONSTRAINT CHK_TipiAbbonamento_Costo CHECK (Costo>0),
CONSTRAINT CHK_TipiAbbonamento_PosteggioGarantito CHECK (PosteggioGarantito
IN ('0','1')),
CONSTRAINT CHK_TipiAbbonamento_Sottoscrivibile CHECK (Sottoscrivibile  IN
('0','1'))
);
```

3.14. Sedi

```
CREATE TABLE Sedi(
pk_Sede INT PRIMARY KEY,
Indirizzo VARCHAR(60) NOT NULL,
Citta VARCHAR(45) NOT NULL,
Telefono VARCHAR(45) NOT NULL,
Stato CHAR(1) DEFAULT 1 NOT NULL,
CONSTRAINT CHK_Sedi_Stato CHECK (Stato IN ('0','1'))
);
```

3.15. Responsabili

```
CREATE TABLE Responsabili(
pk_Responsabile INT PRIMARY KEY,
StipendioMensile FLOAT NOT NULL,
fk_Sede INT NOT NULL,
FOREIGN KEY(pk_Responsabile) REFERENCES Dipendenti (pk_Dipendente),
FOREIGN KEY(fk_Sede) REFERENCES Sedi (pk_Sede),
CONSTRAINT CHK_Responsabili_StipendioMensile  CHECK (StipendioMensile >0)
);
```

3.16. ParcheggiAutomatici

```
CREATE TABLE ParcheggiAutomatici(
pk_ParcheggioAutomatico INT PRIMARY KEY,
Citta VARCHAR(45) NOT NULL,
Indirizzo VARCHAR(60) NOT NULL,
AssicurazioneParcheggio VARCHAR(45),
Telefono VARCHAR(45) NOT NULL,
Zona INT NOT NULL,
MaggiorazioneZona FLOAT NOT NULL,
Stato CHAR(1) DEFAULT 1 NOT NULL,
fk_Sede INT NOT NULL,
FOREIGN KEY(fk_Sede) REFERENCES Sedi (pk_Sede),
UNIQUE(Citta,Indirizzo),
CONSTRAINT CHK_ParcheggiAutomatici_Stato CHECK (Stato IN ('0','1'))
);
```

3.17. CodiciSconto

```
CREATE TABLE CodiciSconto(
pk_CodiceSconto INT PRIMARY KEY,
Codice VARCHAR(15) NOT NULL UNIQUE,
Sconto FLOAT NOT NULL,
DataScadenza DATE NOT NULL,
fk_ParcheggioAutomatico INT NOT NULL,
FOREIGN KEY(fk_ParcheggioAutomatico) REFERENCES
ParcheggiAutomatici(pk_ParcheggioAutomatico),
CONSTRAINT CHK_CodiciSconto_Sconto CHECK (Sconto>0)
);
```

3.18. Abbonamenti

```
CREATE TABLE Abbonamenti(
pk_Abbonamento INT PRIMARY KEY,
DataInizioValidita DATE NOT NULL,
DataFineValidita DATE NOT NULL,
PrezzoPagato FLOAT NOT NULL,
TipoPagamento VARCHAR(45) NOT NULL,
fk_Veicolo INT NOT NULL,
fk_TipoAssicurazione INT,
fk_TipoAbbonamento INT NOT NULL,
fk_ParcheggioAutomatico INT NOT NULL,
fk_CodiceSconto INT,
FOREIGN KEY(fk_Veicolo) REFERENCES Veicoli (pk_Veicolo),
FOREIGN KEY(fk_TipoAssicurazione) REFERENCES TipiAssicurazione
(pk_TipoAssicurazione) ON DELETE SET NULL,
FOREIGN KEY(fk_TipoAbbonamento) REFERENCES TipiAbbonamento
(pk_TipoAbbonamento),
FOREIGN KEY(fk_ParcheggioAutomatico) REFERENCES ParcheggiAutomatici
(pk_ParcheggioAutomatico),
FOREIGN KEY(fk_CodiceSconto) REFERENCES CodiciSconto(pk_CodiceSconto),
CONSTRAINT CHK_Abbonamenti_DataFineValidita CHECK (DataFineValidita >
DataInizioValidita),
CONSTRAINT CHK_Abbonamenti_PrezzoPagato CHECK (PrezzoPagato>=0)
);
```

3.19. Operatori

```
CREATE TABLE Operatori(
pk_Operatore INT PRIMARY KEY,
fk_ParcheggioAutomatico INT NOT NULL,
FOREIGN KEY(pk_Operatore) REFERENCES Dipendenti (pk_Dipendente),
FOREIGN KEY(fk_ParcheggioAutomatico) REFERENCES ParcheggiAutomatici
(pk_ParcheggioAutomatico)
);
```

3.20. Sanzioni

```
CREATE TABLE Sanzioni(
pk_Sanzione INT PRIMARY KEY,
MotivoSanzione VARCHAR(100) NOT NULL,
Rilevamento DATE NOT NULL,
Costo FLOAT NOT NULL,
StatoPagamento CHAR(1) NOT NULL,
DataEmissione DATE,
DataScadenza DATE,
fk_Operatore INT NOT NULL,
fk_Veicolo INT NOT NULL,
FOREIGN KEY(fk_Operatore) REFERENCES Operatori (pk_Operatore),
FOREIGN KEY(fk_Veicolo) REFERENCES Veicoli (pk_Veicolo),
CONSTRAINT CHK_Sanzioni_StatoPagamento CHECK (StatoPagamento IN ('0','1')),
CONSTRAINT CHK_Sanzioni_DataEmissione CHECK (DataEmissione >= Rilevamento),
CONSTRAINT CHK_Sanzioni_DataScadenza CHECK (DataScadenza >= DataEmissione),
CONSTRAINT CHK_Sanzioni_Costo CHECK (Costo>0)
);
```

3.21. Colonne

```
CREATE TABLE Colonne(
pk_Colonna INT PRIMARY KEY,
NumeroPianiSopraelevati INT NOT NULL,
NumeroPianiSottoterra INT NOT NULL,
Stato VARCHAR(45),
fk_ParcheggioAutomatico INT NOT NULL,
FOREIGN KEY(fk_ParcheggioAutomatico) REFERENCES ParcheggiAutomatici
(pk_ParcheggioAutomatico),
CONSTRAINT CHK_Colonne_NumeroPianiSopraelevati CHECK
(NumeroPianiSopraelevati>=0),
CONSTRAINT CHK_Colonne_NumeroPianiSottoterra CHECK (NumeroPianiSottoterra>=0)
);
```

3.22. Box

```
CREATE TABLE Box(
pk_Box INT PRIMARY KEY,
Piano INT NOT NULL,
Stato CHAR(1) DEFAULT 1 NOT NULL,
fk_Area INT NOT NULL,
fk_Colonna INT NOT NULL,
FOREIGN KEY(fk_Area) REFERENCES Aree (pk_Area),
FOREIGN KEY(fk_Colonna) REFERENCES Colonne (pk_Colonna),
CONSTRAINT CHK_Box_Stato CHECK (Stato IN ('0','1'))
);
```

3.23. Carburanti

```
CREATE TABLE Carburanti(
pk_Carburante INT PRIMARY KEY,
Nome VARCHAR(45) NOT NULL UNIQUE
);
```

3.24. CarburantiSupportati

```
CREATE TABLE CarburantiSupportati(
fk_Box INT NOT NULL,
fk_Carburante INT NOT NULL,
FOREIGN KEY(fk_Box) REFERENCES Box(pk_Box),
FOREIGN KEY(fk_Carburante) REFERENCES Carburanti(pk_Carburante),
PRIMARY KEY(fk_Box, fk_Carburante)
);
```

3.25. TipiTurno

```
CREATE TABLE TipiTurno(
pk_TipoTurno INT PRIMARY KEY,
Nome VARCHAR(45) NOT NULL UNIQUE,
OraInizio TIMESTAMP NOT NULL,
OraFine TIMESTAMP NOT NULL,
RetribuzioneOraria FLOAT NOT NULL,
CONSTRAINT CHK_TipiTurno_RetribuzioneOraria CHECK (RetribuzioneOraria>0)
);
```

3.26. Turni

```
CREATE TABLE Turni(
pk_Turno INT PRIMARY KEY,
Data DATE NOT NULL,
Inizio TIMESTAMP,
Fine TIMESTAMP,
fk_ParcheggioAutomatico INT NOT NULL,
fk_Operatore INT NOT NULL,
fk_TipoTurno INT NOT NULL,
FOREIGN KEY(fk_ParcheggioAutomatico) REFERENCES ParcheggiAutomatici
(pk_ParcheggioAutomatico),
FOREIGN KEY(fk_Operatore) REFERENCES Operatori (pk_Operatore),
FOREIGN KEY(fk_TipoTurno) REFERENCES TipiTurno (pk_TipoTurno),
CONSTRAINT CHK_Turni_Fine CHECK (Fine>Inizio)
);
```

3.27. Soste

```
CREATE TABLE Soste(
pk_Sosta INT PRIMARY KEY,
Inizio TIMESTAMP NOT NULL,
Fine TIMESTAMP,
fk_Veicolo INT NOT NULL,
fk_Box INT NOT NULL,
FOREIGN KEY(fk_Veicolo) REFERENCES Veicoli (pk_Veicolo),
FOREIGN KEY(fk_Box) REFERENCES Box (pk_Box),
CONSTRAINT CHK_Soste_Fine CHECK (Fine>Inizio)
);
```

3.28. SosteOrarie

```
CREATE TABLE SosteOrarie(
pk_SostaOraria INT PRIMARY KEY,
TipoPagamento VARCHAR(45) NOT NULL,
PrezzoPagato FLOAT NOT NULL,
fk_CodiceSconto INT,
FOREIGN KEY(fk_CodiceSconto) REFERENCES CodiciSconto (pk_CodiceSconto),
FOREIGN KEY(pk_SostaOraria) REFERENCES Soste (pk_Sosta),
CONSTRAINT CHK_SosteOrarie_PrezzoPagato CHECK (PrezzoPagato>=0)
);
```

3.29. SosteAbbonamenti

```
CREATE TABLE SosteAbbonamenti(
pk_SostaAbbonamento INT PRIMARY KEY,
fk_Abbonamento INT NOT NULL,
FOREIGN KEY(pk_SostaAbbonamento) REFERENCES Soste (pk_Sosta),
FOREIGN KEY(fk_Abbonamento) REFERENCES Abbonamenti (pk_Abbonamento)
);
```

3.30. Giorni

```
CREATE TABLE Giorni(
pk_Giorno INT PRIMARY KEY,
Giorno VARCHAR(9) NOT NULL UNIQUE,
CONSTRAINT CHK_Giorni_Giorno CHECK (Giorno IN
('Lunedi','Martedi','Mercoledi','Giovedi','Venerdi','Sabato','Domenica',
'Festivo'))
);
```

3.31. GiorniValidi

```
CREATE TABLE GiorniValidi(
fk_Giorno INT,
fk_TitoloAbbonamento INT,
FOREIGN KEY(fk_Giorno) REFERENCES Giorni (pk_Giorno),
FOREIGN KEY(fk_TitoloAbbonamento) REFERENCES TitoliAbbonamento
(pk_TitoloAbbonamento),
PRIMARY KEY(fk_Giorno, fk_TitoloAbbonamento)
);
```

3.32. Notifiche

```
CREATE TABLE Notifiche(
pk_Notifica INT PRIMARY KEY,
Data DATE NOT NULL,
Descrizione VARCHAR(500) NOT NULL,
Tipo INT NOT NULL,
fk_Mittente INT NOT NULL,
fk_Destinatario INT NOT NULL,
FOREIGN KEY(fk_Mittente) REFERENCES Persone (pk_Persona),
FOREIGN KEY(fk_Destinatario) REFERENCES Persone (pk_Persona)
);
```

4. Rappresentazione grafica dello schema logico