Basi di dati

RELAZIONE DEL PROGETTO

VICTOR DUTCA (1122137) & DANIEL ROSSI (1125444)

1. Abstract

1.1. Cosa viene modellato ed in che contesto:

Una “autofficina meccanica” è una realtà lavorativa che si occupa della riparazione e manutenzione di autoveicoli.

Nel nostro esempio prendiamo a riferimento una ipotetica autofficina di medie dimensioni, con un discreto numero di dipendenti e con la necessità di automatizzare diverse fasi che prima risultavano troppo dispendiose in termini di tempo.

Il ciclo di riparazione di un veicolo procede come segue:

- **FASE DI ACCETTAZIONE**
  Proprietario e veicolo vengono inseriti nel database generando quindi un ordine di riparazione legato al veicolo in oggetto, gli viene quindi assegnato un meccanico, il quale esegue un controllo e stila una lista di pezzi necessari alla riparazione del veicolo.

- **FASE CONSENSO PREVENTIVO**
  La lista viene inserita nel DB e viene quindi generato un preventivo dei pezzi, questo viene presentato al cliente che può accettarlo o rifiutarlo.

- **FASE ATTESA PEZZI**
  In caso il preventivo venga accettato si procede all’ordinazione dei pezzi. Ogni pezzo ricevuto viene registrato nel database che procede ad una serie di operazioni tra cui quella di assegnare i pezzi ricevuti ai veicoli.

- **FASE DI LAVORAZIONE**
  Una volta che tutti i pezzi richiesti da un ordine sono arrivati si cominciano le riparazioni.

- **FASE CONCLUSIVA**
  Una volta finite le riparazioni, il meccanico inserisce relativamente all’ordine finito l’ammontare delle ore di lavoro, viene così generato il costo finale dell’intervento tenendo conto delle ore di lavoro e del costo dei pezzi.

1.2. Quali funzionalità offre il sistema:

Le tipiche operazioni saranno quelle di inserire nuovi veicoli ed i loro proprietari, creare ordini di riparazione per i nuovi veicoli arrivati, ordinare e ratificare l’arrivo dei pezzi, quantificare il costo dei pezzi per ciascun ordine ed il costo finale, accettare i preventivi.
Molte operazioni sono state automatizzate per rendere più veloce il processo di riparazione e migliorare l’efficienza dell’autofficina.

2. Analisi dei requisiti

Si vuole realizzare una base di dati per gestire gli ordini di riparazione di un'autofficina.

Gli ordini vengono registrati nello Storico Ordine, il quale tiene traccia di ogni ordine mediante:

- Numero Ordine
- Targa
- Matricola
- Data inizio lavori

Ogni ordine è inserito in base alla fase attuale nelle rispettive istanze: Accettazione, Consenso Preventivo con l'attributo preventivo, Attesa Pezzi con l'attributo preventivo, In Lavorazione con l'attributo preventivo, Concluso con l'attributo preventivo, costo finale e data fine lavori. Queste istanze sono generalizzazioni dello Storico Ordine.

Ogni ordine viene quindi eseguito su un Veicolo di cui ci interessa:

- Targa
- Marca
- Modello
- Versione
- Proprietario

I proprietari dei Veicoli sono Clienti dell'autofficina.

Ogni Cliente e Dipendente dell'autofficina è inserito nell'entità Persone, di ogni Persona ci interessano le principali informazioni anagrafiche e di contatto come:

- Codice Fiscale
- Nome
- Cognome
- Cellulare
- Email

I Dipendenti si dividono in due qualifiche: segretari che si occupano della gestione della parte amministrativa dell'officina e del DB, meccanici che hanno il compito di eseguire le riparazioni sui veicolo a loro assegnati, ogni Dipendente è identificato con una matricola.

L'officina per scegliere i pezzi di ricambio fa riferimento ad un Catalogo dove di ogni pezzo sono d'interesse:

- Codice pezzo
- Versione pezzo
- Categoria
- Nome pezzo
- Marca
- Modello
- Versione
- Prezzo
- Fornitore

Ogni Fornitore inoltre è identificato da un Nome e da un Contatto.

Ad ogni ordine presente nello Storico Ordine corrisponde una linea nella tabella Decisioni, nella seguente verrà dichiarata il risultato della scelta (accettato / non accettato) del cliente riguardo al preventivo proposto dal meccanico per un dato ordine, sarà d'interesse:

- Numero Ordine
- Risultato

Ogni ordine ha una lista di pezzi che verranno inseriti nell'entità Pezzi Necessari, di questi vogliamo sapere i seguenti dati forniti dal catalogo:

- Codice Pezzo
- Nome
- Prezzo

oltre a questi necessitiamo

- Numero Ordine
- Quantità Necessaria
- Quantità Disponibile

Ogni pezzo necessario alla riparazione di un veicolo viene inserito tra gli Ordini Pendenti se il Preventivo riceve risultato positivo.

Di ogni Ordine Pendente è di interesse:

- Codice Pezzo
- Quantità

3. Progettazione concettuale

3.1. Lista delle **classi**

- **Persone**: informazioni anagrafiche e di contatto di Clienti e Dipendenti.
    - Codice Fiscale: string
    - Nome: string
    - Cognome: string
    - Email: string
    - Cellulare: int

  Sono definite le seguenti sottoclassi, con una gerarchia totale e parziale:
    - Cliente
    - Dipendente:
        - Matricola int

  Per la classe Dipendenti sono definite le seguenti sottoclassi, con una gerarchia totale ed esclusiva:

    - Meccanico
    - Segretario

- **Veicolo**: lista di tutti i veicoli accettati dall'autofficina.
    - Targa: string
    - Marca: string
    - Modello: string
    - Versione: string
    - Proprietario: string

- **Storico Ordine**: lista di tutti gli ordini di riparazione.
    - Numero ordine: int
    - Targa: string
    - Matricola: int
    - Data inizio lavori: date

  Sono definite le seguenti sottoclassi, con una gerarchia totale ed esclusiva:
    - Accettazione
    - Consenso Preventivo: Preventivo int
    - Attesa Pezzi: Preventivo int
    - In Lavorazione: Preventivo int
    - Concluso: Preventivo int, Costo finale int, Data fine lavori: date

- **Decisione**: esprime per ogni ordine di riparazione l'esito della decisione del cliente di accettare o rifiutare il preventivo generato per il suo ordine.
  Saranno di interesse:
    - Numero Ordine: int
    - Risultato: string

- **Pezzi Necessari**: ogni ordine di riparazione necessita di alcuni pezzi di ricambio.
    - Numero Ordine: int
    - Codice Pezzo: int
    - Pezzo: string
    - Quantità Necessaria: int
    - Quantità Disponibile: int
    - Prezzo: int

- **Catalogo**: lista di tutti i pezzi disponibili in listino da poter ordinare.
    - Codice Pezzo: string
    - Versione Pezzo: int
    - Categoria: string
    - Pezzo: string
    - Marca: string
    - Modello: string
    - Versione: string
    - Prezzo: int
    - Fornitore {Nome,Contatto}: {string,string}

- **Ordine Pendente**: lista dei pezzi ordinati.
    - Codice Pezzo: string
    - Quantità: int

3.2. Lista delle **associazioni**

- *Cliente – Veicolo:* **Possesso**
  - Ogni Cliente può essere proprietario di N veicoli, mentre un veicolo può essere di proprietà di un solo cliente.
  - Molteplicità (N,1)

- *Veicolo – Storico Ordine:* **Inserzione**
  - Un Veicolo può essere oggetto di almeno un ordine di riparazione mentre un ordine di riparazione su essere eseguito su uno ed un solo veicolo.
  - Molteplicità (N,1)

- *Storico Ordine – Pezzi Necessari:* **Richiesta**
  - Ogni ordine di riparazione può non avere bisogno di pezzi di ricambio, mentre ogni pezzo richiesto sarà stato richiesto almeno una volta.
  - Molteplicità (N,M)

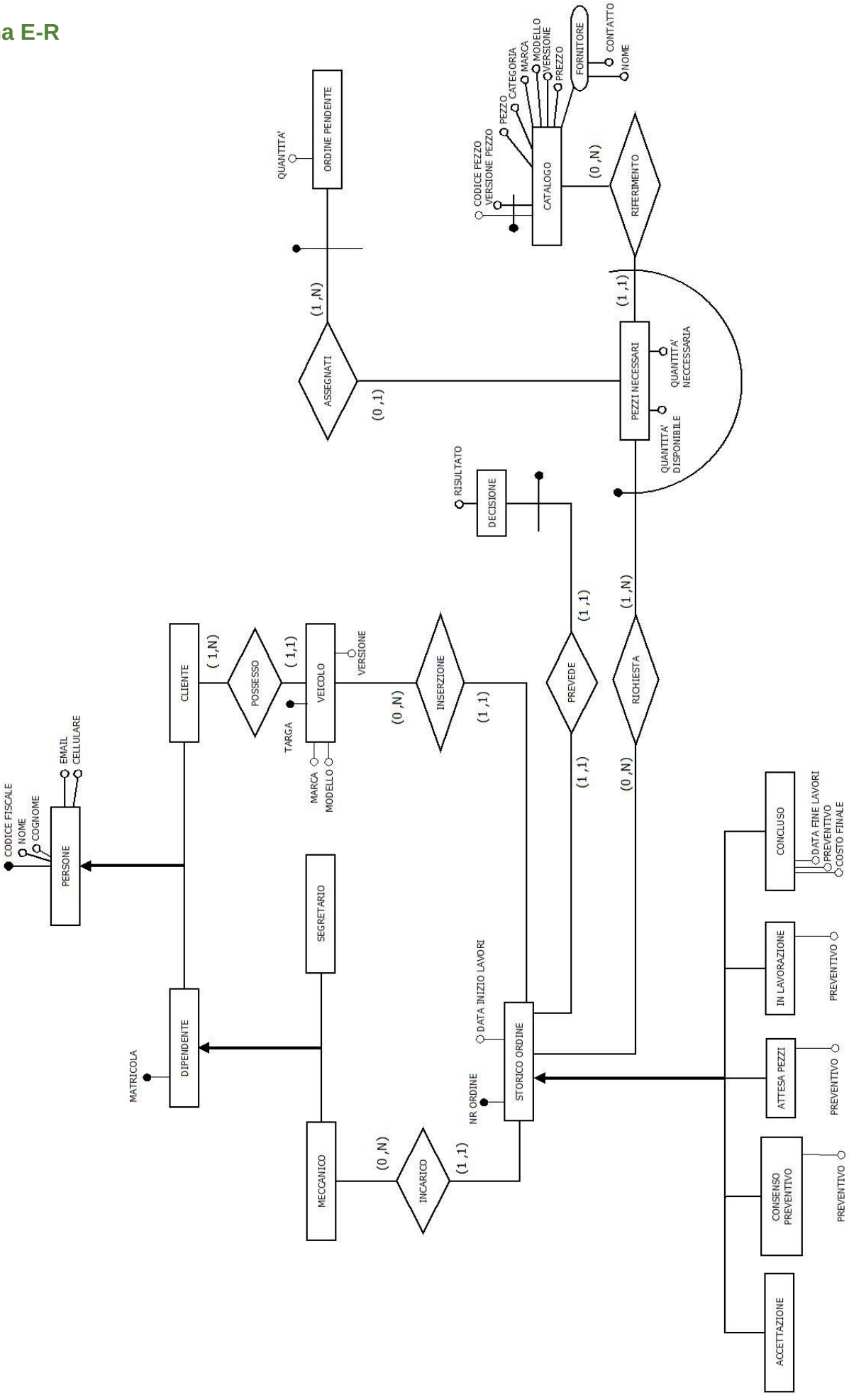
- *Pezzi Necessari – Catalogo:* **Riferimento**
  - Ogni pezzo richiesto ne ha uno e uno solo corrispondente nel Catalogo mentre un Pezzo del Catalogo può anche non essere mai richiesto.
  - Molteplicità (1,N)

- *Pezzi Necessari – Ordine Pendente:* **Assegnati**
  - Ogni pezzo necessario può esserci o non esserci come pezzo ordinato, mentre un pezzo ordinato sarà stato ordinato almeno una volta.
  - Molteplicità (1,N)

- *Meccanico – Storico Ordine:* **Incarico**
  - Ogni meccanico può essere incaricato di eseguire delle riparazioni su un certo numero di ordini di riparazione, mentre un ordine di riparazione può essere oggetto di lavoro di un solo meccanico.
  - Molteplicità (N,1)

- *Storico Ordine – Decisione:* **Prevede**
  - Ogni ordine di riparazione prevede una decisione, ogni decisione è riferita ad un solo ordine.
  - Molteplicità (1,1)

Schema E-R


PERSONE
CODICE FISCALE
NOME
COGNOME
EMAIL
CELLULARE
CLIENTE
POSSESSO
(1,N)
(1,1)
VEICOLO
TARGA
MARCA
MODELLO
VERSIONE
INSERZIONE
(0,N)
(1,1)
DIPENDENTE
MATRICOLA
SEGRETARIO
MECCANICO
INCARICO
(0,N)
(1,1)
STORICO ORDINE
NR ORDINE
DATA INIZIO LAVORI
PREVEDE
(1,1)
(1,1)
DECISIONE
RISULTATO
RICHIESTA
(1,N)
(0,N)
PEZZI NECESSARI
QUANTITA' NECESSARIA
QUANTITA' DISPONIBILE
ASSEGNATI
(1,N)
(0,1)
ORDINE PENDENTE
QUANTITA'
(1,1)
RIFERIMENTO
(0,N)
CATALOGO
CODICE PEZZO
VERSIONE PEZZO
PEZZO
CATEGORIA
MARCA
MODELLO
VERSIONE
PREZZO
FORNITORE
NOME
CONTATTO
CONCLUSO
DATA FINE LAVORI
PREVENTIVO
COSTO FINALE
IN LAVORAZIONE
PREVENTIVO
ATTESA PEZZI
PREVENTIVO
CONSENSO PREVENTIVO
PREVENTIVO
ACCETTAZIONE

Progettazione logica

4.1. Analisi delle ridondanze

4.1.1. Verifica della necessità di inserire un attributo "*Fornitore*" (ridondante) all'entità Ordine Pendente

- **Operazione di massimo rilievo**:

  Stampare il contenuto della tabella Ordine Pendente, per ogni pezzo riportare anche il nome del fornitore → 40 volte al giorno

  - Con ridondanza:

Concetto	Tipo	Accessi	Tipo
Ordine Pendente	E	1	L

Totale = 1x40 accessi in lettura = 40 accessi totali

  - Senza ridondanza:

Concetto	Tipo	Accessi	Tipo
Ordine Pendente	E	1	S
Assegnati	R	1	S
Pezzi Necessari	E	1	S
Riferimento	R	1	S
Catalogo	E	1	S

Totale = 5x40 accessi in lettura = 200 accessi totali

- **Esito dell'analisi:**
  Poiché 200 >> 40, si decide di inserire la ridondanza.

4.2. Eliminazione delle generalizzazioni

- Generalizzazione totale e parziale “**Persone**”

  La generalizzazione viene in parte eliminata mediante l’accorpamento dell’entità figlia Cliente nell’entità padre, mentre l’entità Dipendente viene legata all’entità padre attraverso una relazione detta “*essere*”. Viene aggiunto all’entità Dipendenti l’attributo Codice Fiscale, così facendo manteniamo l’integrità referenziale tra l’entità Dipendente e l’entità Persona.

- Generalizzazione totale ed esclusiva “**Dipendenti**”

  La generalizzazione viene eliminata mediante l’accorpamento delle entità figlie nell’entità padre, sotto l’assunzione che l’accesso al padre sia contestuale all’accesso alle figlie.
  Viene aggiunto l’attributo “*mansione*”, che potrà assumere i valori “*Meccanico”* oppure “*Segretario”*, per distinguere il ruolo ricoperto da ciascun dipendente dell’officina.

- Generalizzazione totale ed esclusiva “**Storico Ordine**”

  La generalizzazione viene eliminata mediante l’accorpamento delle entità figlie nell’entità padre, sotto l’assunzione che l’accesso al padre sia contestuale all’accesso alle figlie. All’entità padre vengono accorpati gli attributi di tutte e cinque le generalizzazioni, quindi gli attributi “*data fine lavori*”, “*manodopera*” e “*costo finale*” , vengono ora accorpate nello “*Storico Ordine*” e settate a *null*, l’idea di questa generalizzazione era quella di poter avere una lista dello stato di ciascun ordine di riparazione, manterremo l’idea solo implementandola con una funzione.

4.3. Partizionamento/accorpamento di entità e relazioni

- Sotto l’assunzione che l’accesso al Fornitore con relativi sotto attributi sia contestuale all’accesso agli ulteriori attributi dell’entità “*Catalogo*” e data la presenza di una cardinalità (1, N), l’attributo multi valore **Fornitore** viene separato dall’entità “*Catalogo*”, si viene così a creare una nuova entità detta Fornitore, formata dagli attributi “*NomeFornitore*” e “*Contatto*”, dove prima si trovava l’attributo multi valore Fornitore, ora si trova solo il NomeFornitore rinominato Fornitore.

- Abbiamo inserito all’interno dell’entità “*Pezzi Necessari*” l’attributo Richiesto di tipo date, servirà per ordinare in modo prioritario le richieste di uno stesso pezzo da parte di ordini diversi.

4.4. Scelta degli identificatori primari

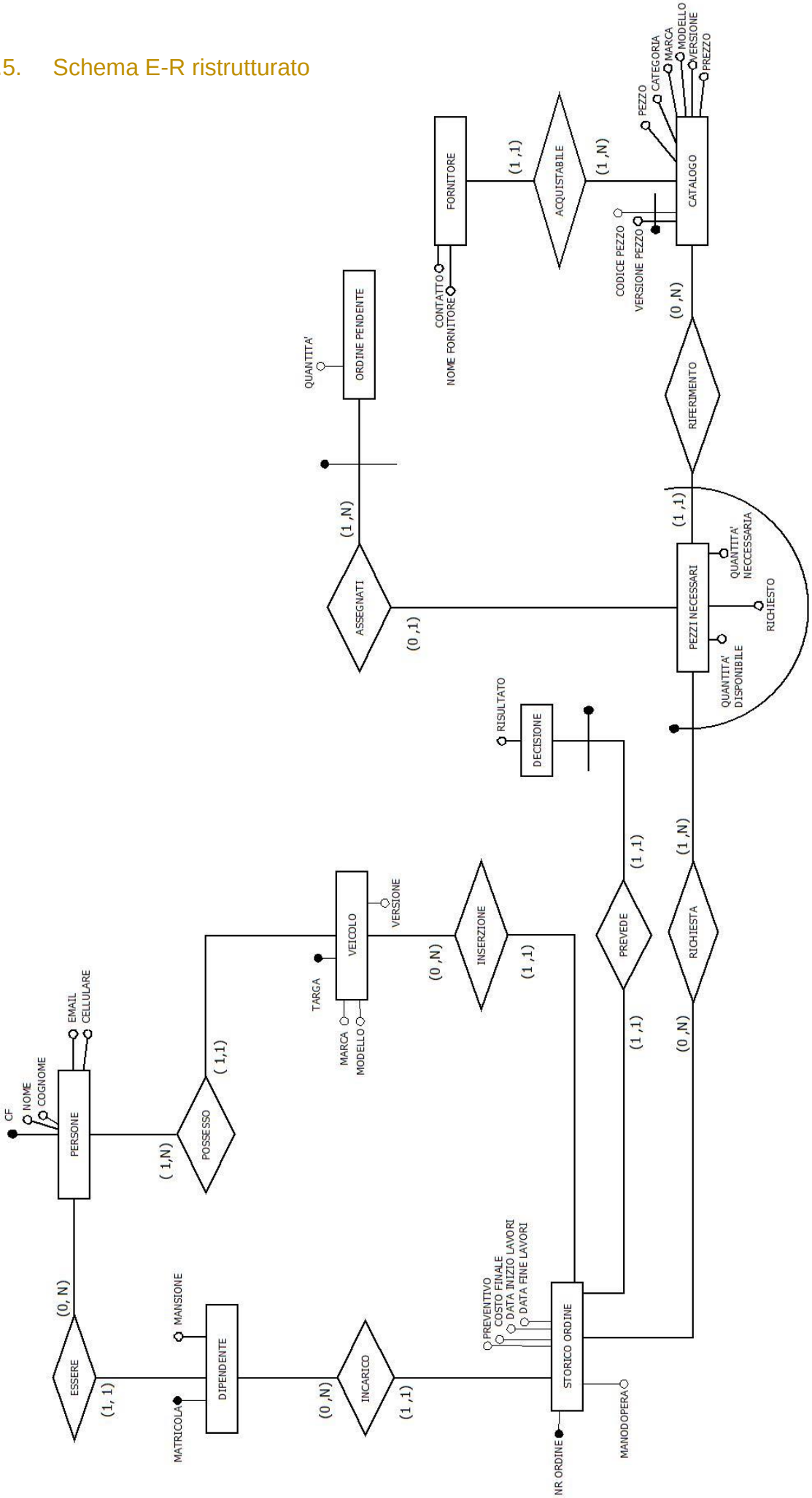
Nota: alcuni identificatori risultano “ridotti” come:

- Numero Ordine = NrOrdine
- Codice Fiscale = CF

Inoltre sia i nomi delle entità che degli attributi formati da più parole sono stati compattati eliminando lo spazio tra le due parole

Concetto	**Tipo**	**Identificatore primario**
Persone	E	*CF*
Dipendente	E	*Matricola*
Veicolo	E	*Targa*
Storico Ordine	E	*NROrdine*
Catalogo	E	*CodicePezzo, VersionePezzo*
Pezzi Necessari	E	*NROrdine, CodicePezzo*
Fornitore	E	*NomeFornitore*
Possesso	R	*CF*
Inserzione	R	*Targa*
Richiesta	R	*NROrdine*
Riferimento	R	*CodicePezzo*
Assegnati	R	*CodicePezzo*
Incarico	R	*Matricola*
Acquistabile	R	*NomeFornitore*
Prevede	R	*NROrdine*
Essere	R	*CF*

4.5. Schema E-R ristrutturato



4.6. Traduzione verso il modello relazionale

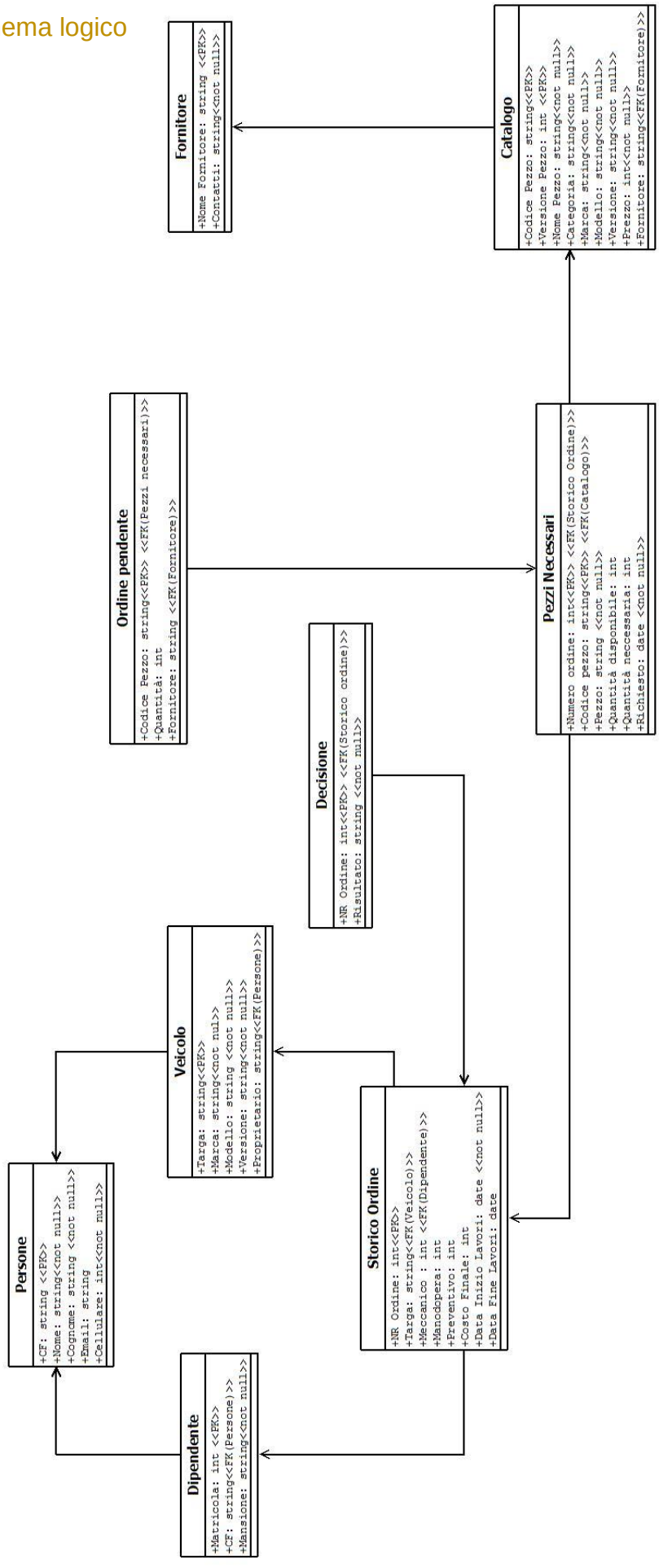
4.6.1. Traduzione delle entità

- **Persona** (CF, Nome, Cognome, Email, Cellulare)
- **Veicolo** (Targa, Marca, Modello, Versione, Proprietario)
  - FK Proprietario REFERENCES Persona(CF)
- **Dipendente** (Matricola, CF, Mansione)
  - FK CF REFERENCES Persona(CF)
- **StoricoOrdine** (NR_Ordine, Targa, Meccanico, DataInizioLavori, DataFineLavori, Manodopera, Preventivo, CostoFinale)
  - FK Targa REFERENCES Veicolo(Targa)
  - FK Meccanico REFERENCES Dipendente(Matricola)
- **PezziNecessari** (NrOrdine, CodicePezzo, Pezzo, QuantitàNecessaria, QuantitàDisponibile, Prezzo, Richiesto)
  - FK CodicePezzo REFERENCES Catalogo(CodicePezzo)
  - FK NROrdine REFERENCES StoricoOrdine(NROrdine)
- **Catalogo** (CodicePezzo, VersionePezzo, Categoria, Pezzo, Marca, Modello, Versione, Prezzo, Fornitore)
  - FK Fornitore REFERENCES Fornitore(NomeFornitore)
- **OrdinePendente** (CodicePezzo, Quantità, Fornitore)
  - FK CodicePezzo REFERENCES Catalogo(CodicePezzo)
  - FK Fornitore REFERENCES Fornitore(NomeFornitore)
- **Fornitore** (NomeFornitore, Contatto)

4.6.2. Traduzione delle relazioni

- **Possesso** (CF)
  - FK CF REFERENCES Persona(CF)
- **Inserzione** (Targa)
  - FK Targa REFERENCES Veicolo (Targa)
- **Richiesta** (NrOrdine)
  - FK NrOrdine REFERENCES StoricoOrdine(NrOrdine)
- **Riferimento** (CodicePezzo)
  - FK CodicePezzo REFERENCES Catalogo(CodicePezzo)
- **Assegnati** (CodicePezzo)
  - FK CodicePezzo REFERENCES Catalogo(CodicePezzo)
- **Incarico** (Matricola)
  - FK Matricola REFERENCES Dipendente(Matricola)
  - FK Settore REFERENCES Settore(Codice)
- **Acquistabile** (NomeFornitore)
  - FK NomeFornitore REFERENCES Fornitore(NomeFornitore)
- **Prevede** (NROrdine)
  - FK NROrdine REFERENCES StoricoOrdine(NROrdine)
- **Essere** (CF)
  - FK CF REFERENCES Persone(CF)

4.7. Schema logico



5. Query, procedure, trigger e funzioni

1. Query

I. La Matricola, il Nome, le OreTotali di lavoro e il numero di OrdiniTotali per ciascun Dipendente con la mansione di meccanico.

```
SELECT I.Matricola, P.Nome, SUM(S.Manodopera) AS OreTotali, count(NROrdine) AS OrdiniTotali
FROM Dipendente I join StoricoOrdine S ON (I.Matricola = S.Meccanico), Persone P
WHERE P.CF = I.CF AND I.Mansione = 'Meccanico'
GROUP BY I.Matricola;
```

Matricola	Nome	OreTotali	OrdiniTotali
1	ALESSANDRO	25	8
2	LUIGI	17	7
3	GIULIA	18	7
4	GIUSEPPINA	16	5

II. Abbiamo appena ricevuto una quantità di pezzi pari a 6 del pezzo 629 e li abbiamo registrati nella tabella ordine pendente, richiamiamo la procedura Assegnazioni per sapere a quali ordini sono stati assegnati.
CALL Assegnazioni(6,629);

```
DROP PROCEDURE IF EXISTS Assegnazioni;
DELIMITER $$
CREATE PROCEDURE Assegnazioni (IN Arrivi int,IN Pezzo nstring (4))
Begin
DECLARE PRIOR int (2);
SET PRIOR = (select limite(Pezzo,Arrivi));
SELECT NROrdine, QuantitaDisponibile AS Quantita FROM PezziNecessari
WHERE CodicePezzo = Pezzo AND QuantitaDisponibile <> 0
ORDER BY Richiesti DESC
LIMIT PRIOR;
END ;$$  DELIMITER ;
```

NROrdine	Quantita
19	2
14	1
9	2
4	1

III. Mostrare per ciascun ordine di riparazione lo stato in cui si trova, gli stati in cui può trovarsi un ordine possono essere: accettazione, preventivo non accettato, attesa pezzi, in lavorazione, concluso.

```
SELECT NROrdine, StatoOrdine (NROrdine) AS StatoOrdine from StoricoOrdine;
```

NROrdine	StatoOrdine
1	CONCLUSO
2	CONCLUSO
3	CONCLUSO
4	CONCLUSO
5	CONCLUSO
6	CONCLUSO
7	CONCLUSO
8	CONCLUSO
9	CONCLUSO
10	CONCLUSO
11	CONCLUSO
12	CONCLUSO
13	CONCLUSO
14	CONCLUSO
15	CONCLUSO
16	CONCLUSO
17	CONCLUSO
18	CONCLUSO
19	CONCLUSO
20	CONCLUSO
21	CONCLUSO
22	CONCLUSO
23	IN LAVORAZIONE
24	IN ATTESA PEZZI
25	PREVENTIVO NON ACCETTATO
26	PREVENTIVO NON ACCETTATO
27	ACCETTAZIONE

IV. Per ogni fornitore restituire il nome del fornitore, il numero di pezzi venduti e il guadagno, il tutto ordinato secondo il guadagno maggiore.

```
SELECT C.Fornitore, COUNT(*) AS NRpezziVenduti, SUM(C.Prezzo) AS Guadagno
FROM PezziNecessari P JOIN (SELECT * FROM Catalogo GROUP BY CodicePezzo) AS C
WHERE P.CodicePezzo=C.CodicePezzo
GROUP BY Fornitore
ORDER BY Guadagno DESC;
```

Fornitore	NRpezziVenduti	Guadagno
ARBO s.r.l	23	875
RAEM s.r.l	16	683
AUTOCAR LAGUNA	14	582
ROSI STAR	15	549

V. Stampare per ciascuna categoria quanti tipi di pezzi sono stati utilizzati, quanti non sono stati utilizzati, e la quantità totale.
Per tipo di pezzo si intende ad esempio il “Tergicristalli”, ci sono diversi Tergicristalli compatibili con i diversi veicoli nel catalogo, in questa query invece il Tergicristalli viene inteso come singolo pezzo.

```
select A.Categoria, A.Utilizzati, A.NonUtilizzati, B.NRpezzi AS Totale
from (select A.Categoria, B.NRpezzi AS Utilizzati,  A.NRpezzi NonUtilizzati
from
(SELECT B.Categoria, NRpezzi
FROM (select C.Categoria, count(*) as NRpezzi
from (select * from Catalogo group by Pezzo) C RIGHT JOIN (select * from Catalogo group by Pezzo) P
ON(C.CodicePezzo = P.CodicePezzo)
where C.Pezzo NOT IN (
select Pezzo
from PezziNecessari)
GROUP BY C.Categoria) AS A RIGHT JOIN (select * from Catalogo group by Categoria) AS B
ON (A.Categoria = B.Categoria)) AS A JOIN (
SELECT A.Categoria, B.NRpezzi
FROM (select * from Catalogo group by Categoria) as A LEFT JOIN
(select C.Categoria, count(*) as NRpezzi
FROM (SELECT * FROM Catalogo GROUP BY Pezzo) C
JOIN (SELECT * FROM PezziNecessari GROUP BY Pezzo) P
ON (C.Pezzo=P.Pezzo)
GROUP BY C.Categoria) AS B ON (A.Categoria = B.Categoria)) AS B ON (A.Categoria = B.Categoria)) AS A
JOIN (select a.Categoria, count(*) as NRpezzi
from (select Categoria, Pezzo from Catalogo group by Pezzo) AS A
group by A.Categoria
) AS B ON (A.Categoria = B.Categoria)
ORDER BY A.Categoria;
```

Categoria	Utilizzati	NonUtilizzati	Totale
Equipaggiamento interno	2	1	3
Filtro	NULL	2	2
Frizione / Parti di montaggio	NULL	8	8
Guarnizioni	1	1	2
Impianto alimentazione carburante	NULL	1	1
Impianto di accensione	NULL	4	4
Impianto elettrico	1	15	16
Impianto gas scarico	NULL	10	10
Motore	5	34	39
Preparazione Carburante	NULL	1	1
Raffreddamento	2	4	6
Riscaldamento / aerazion	1	NULL	1
Sistema chiusura	NULL	2	2
Sistema frenante	5	11	16
Sistemi per il comfort	NULL	1	1
Sospensione /ammortizzazione	4	9	13
Sterzo	NULL	6	6
Tergicristalli	1	NULL	1
Trasmissione a cinghia	NULL	3	3
Trasmissione finale	NULL	5	5

VI. Stampa il guadagno totale dell'officina, risultante dalla somma del numero di ore lavorate per ciascun ordine moltiplicato per 25€ all'ora, più il 10% sui pezzi, inserito di default nel preventivo.

```
SELECT Saldo() AS Saldo;
```

Saldo
2239

2. Funzioni

I. Utilizzata nella query nr. 3, restituisce lo stato in cui si trova un ordine di riparazione.

```
DROP FUNCTION IF EXISTS StatoOrdine;
DELIMITER $$
CREATE FUNCTION StatoOrdine( Ordine int (5) )
RETURNS STRING(30)
BEGIN
DECLARE Stato STRING(30);
IF (SELECT COUNT(*) FROM PezziNecessari WHERE NROrdine=Ordine) = 0
then
SET Stato='ACCETTAZIONE';
RETURN Stato;
ELSEIF (select Consenso from Preventivo where NROrdine=Ordine) = 'Negativo'
then
SET Stato='PREVENTIVO NON ACCETTATO';
RETURN Stato;
ELSEIF (select SUM(QuantitaNecessaria) from PezziNecessari where NROrdine=Ordine)
<>
(select SUM(QuantitaDisponibile) from PezziNecessari where NROrdine=Ordine)
then
SET Stato='IN ATTESA PEZZI';
RETURN Stato;
ELSEIF (select CostoFInale from StoricoOrdine where NROrdine=Ordine) = 0
then
SET Stato='IN LAVORAZIONE';
RETURN Stato;
ELSE
SET Stato='CONCLUSO';
RETURN Stato;
END IF; END ;$$
DELIMITER :
```

II. Invocata dalla query 6, restituisce il saldo totale dell'officina, ossia la somma del numero di ore lavorate per ciascun ordine moltiplicato per 25€ all'ora, più il 10% sui pezzi, inserito di default nel preventivo.

```
DROP FUNCTION IF EXISTS Saldo;
DELIMITER $$
CREATE FUNCTION Saldo()
RETURNS int (30)
BEGIN
DECLARE SituazioneEconomica int(30);
DECLARE PercentualePezzi int(10);
DECLARE OreMensili int(10);
SET PercentualePezzi = ((select SUM(Preventivo) from StoricoOrdine WHERE Manodopera <> 0)/100)*10 ;
SET OreMensili = (select SUM(Manodopera) from StoricoOrdine)*25 ;
SET SituazioneEconomica = PercentualePezzi + OreMensili;
RETURN SituazioneEconomica;
END ;$$
DELIMITER ;
```

III. Quando si ricevono un certo numero di pezzi, questi vanno registrati nella tabella OrdinePendente, quando si richiama l'update sulla tabella bisogna inserire la quantità rimanente, ossia la differenza tra i pezzi ordinati e quelli ricevuti, quello che fa questa funzione è evitare che l'utente una volta ricevuti i pezzi debba andare a vedere quanti pezzi con quel CodicePezzo erano stati ordinati e poi fare la differenza, questa funzione automatizza questa parte evitando perdite di tempo e possibili errori.

```
DROP FUNCTION IF EXISTS Ricevuti;
DELIMITER $$
CREATE FUNCTION Ricevuti(Arrivi int (2), Pezzo nstring (4))
RETURNS int (2)
BEGIN
DECLARE Quantita_Rimanente int (2);
DECLARE Quantita_Precedente int (2);
SET Quantita_Precedente = (select Quantita from OrdinePendente where CodicePezzo = Pezzo);
SET Quantita_Rimanente = Quantita_Precedente - Arrivi;
if Quantita_Rimanente < 0
then
        return Quantita_Precedente;
elseif Quantita_Rimanente > 0
then
        return Quantita_Rimanente;
else
        return 0;
end if; END ;$$
DELIMITER ;
```

IV. Questa funzione viene utilizzata quando si deve creare un nuovo ordine di riparazione, restituisce il numero identificativo di ordine + 1, così si può avere il prossimo numero di ordine senza doverselo andare a vedere per poi potenzialmente dimenticare e commettere quindi qualche errore.

```
DROP FUNCTION IF EXISTS prossimo;
DELIMITER $$
CREATE FUNCTION prossimo()
RETURNS int (2)
BEGIN
DECLARE prossimo int;
SET prossimo = (select Max(NROrdine) from StoricoOrdine);
if prossimo IS NULL
then set prossimo = 1;
else set prossimo = prossimo +1;
end if;
return prossimo;
END ;$$
DELIMITER ;
```

V. Questa funzione è utilizzata dalla procedura della query 2, restituisce un valore che rappresenta di quanto va limitata la lista affinché la quantità ricevuta sia compatibile con la somma dei pezzi richiesti dagli ordini ordinati secondo l'attributo richiesto.

```
DROP FUNCTION IF EXISTS limite;
DELIMITER $$
CREATE FUNCTION limite(Pezzo nstring (4),Arrivi int (2))
RETURNS int (2)
BEGIN
        declare PRIOR int;
        set PRIOR = (select count(*) from PezziNecessari where CodicePezzo=Pezzo AND
QuantitaDisponibile);
        WHILE(
                select SUM(A.QuantitaDisponibile)
                from (
                        select CodicePezzo,QuantitaDisponibile
                        from PezziNecessari
                        where CodicePezzo=Pezzo AND QuantitaDisponibile <> 0
                        ORDER BY Richiesti DESC
                        LIMIT PRIOR
                        ) A
                ) <> Arrivi AND PRIOR >0
        DO
        set PRIOR=PRIOR-1;
        end while;
RETURN PRIOR; END ;$$
 DELIMITER ;
```

VI. Stiamo creando un nuovo ordine di riparazione, dobbiamo assegnare un meccanico, questa funzione assegna il primo meccanico disponibile, nel caso siano tutti occupati, quello con il numero di ore di lavoro minore.

```
DROP FUNCTION IF EXISTS Meccanico;
DELIMITER $$
CREATE FUNCTION Meccanico()
RETURNS int (2)
BEGIN
DECLARE candidato int (2);
set candidato = (select Matricola from Dipendente
        where Mansione = 'Meccanico' and Matricola NOT IN
        (SELECT Meccanico from StoricoOrdine where Manodopera = 0) LIMIT 1);
        if candidato IS NULL
        then set candidato = (select A.Meccanico from
(select Meccanico, SUM(Manodopera) AS OreTotali from StoricoOrdine group by Meccanico) as A
        where A.OreTotali = (select MIN(A.OreTotali) from
(select SUM(Manodopera) AS OreTotali from StoricoOrdine group by Meccanico) AS A LIMIT 1);
        end if;
return candidato; END ;$$
 DELIMITER ;
```

3. Trigger

I. Ogni volta che si aggiunge un pezzo nella tabella PezziNecessari, questo trigger aggiornerà il preventivo relativo all'ordine richiedente del pezzo, sommando al preventivo vecchio il preventivo stesso più il prezzo del nuovo pezzo per la quantità richiesta.

```
delimiter $$
create trigger CalcoloPreventivo
after insert on PezziNecessari
        for each row
begin
        update StoricoOrdine
        set Preventivo=Preventivo+new.Prezzo*new.QuantitaNecessaria
        where NROrdine=new.NROrdine;
end; $$
delimiter ;
```

II. Ogni qual volta noi creiamo un nuovo ordine di riparazione viene creata una nuova riga nella tabella decisione che ci permette di indicare la scelta del cliente riguardo il preventivo.

```
delimiter $$
create trigger AccettazionePreventivo
after insert on StoricoOrdine
        for each row
begin
        INSERT INTO Preventivo values (new.NROrdine,'Negativo');
end; $$
delimiter ;
```

III. Ogni qual volta verrà inserito un veicolo all'interno della tabella Veicolo, si procederà all'eliminazione di tutte le righe della tabella OrdinePendente con quantità pari a 0.

```
delimiter $$
create trigger OrdineCompleto
after insert on Veicolo
        for each row
begin
        delete from OrdinePendente
                where Quantita=0;
end; $$
delimiter ;
```

IV. Quando si inserisce un pezzo richiesto da un ordine di riparazione nella tabella PezziNecessari, si deve verificare se il veicolo su cui verrà eseguita la riparazione è compatibile con il pezzo richiesto, se così non fossi si richiede di generare un errore.

```
delimiter $$
create trigger CompatibilitàPezzo
before insert on PezziNecessari
        for each row
begin
if (select A.Marca
        from(
                        select V.Marca,V.Modello,V.Versione
                        from Veicolo V join StoricoOrdine S ON (V.Targa=S.Targa)
                        where S.NROrdine=New.NROrdine
                ) A join (
                        select C.Marca,C.Modello,C.Versione
                        from Catalogo C
                        where NEW.CodicePezzo=C.CodicePezzo
                        ) B
        where A.Marca=B.Marca AND A.Modello=B.Modello AND A.Versione=B.Versione) IS NULL
then SIGNAL SQLSTATE '45000' SET MYSQL_ERRNO=30001, MESSAGE_TEXT='PEZZO INCOMPATIBILE';
end if;
end; $$
delimiter ;
```

V. Ogni qual volta si inserisce un nuovo veicolo si deve verificare che siano disponibili nel catalogo pezzi sostitutivi per quel veicolo, nel caso non fosse si genera un errore.

```
delimiter $$
create trigger CompatibilitàVeicolo
before insert on Veicolo
        for each row
begin
if (select C.Marca
    from Catalogo C
    where NEW.Marca=C.Marca AND NEW.Modello=C.Modello AND NEW.Versione=C.Versione
    group by C.Marca) IS NULL
then
SIGNAL SQLSTATE '45000' SET MYSQL_ERRNO=30001, MESSAGE_TEXT='VEICOLO NON DISPONIBILE';
end if;
end; $$
delimiter ;
```

VI. Questo, insieme al prossimo trigger, è di fondamentale importanza al fine di automatizzare la gestione dei pezzi ordinati e ricevuti.
Questo trigger si attiva quando il cliente accetta il preventivo, quindi se nella tabella Decisione l'attributo Risultato che prima si trovava nello stato 'Negativo' viene settato a 'Positivo' significa che la lista di pezzi richiesti che si trova nella tabella PezziNecessari deve essere ordinata.
Ogni singolo pezzo da ordinare deve essere quindi inserito nella tabella OrdinePendente, avendo cura però di controllare se nella tabella stessa non sia già presente quel pezzo, nel caso non sia presente lo si inserisce impostando la quantità uguale alla QuantitàNecessaria di quel pezzo, se invece quel pezzo risulta già presente, la quantità deve essere aggiornata sommando alla quantità stessa la QuantitàNecessaria di quel pezzo.

```
delimiter $$
create trigger Ordinazioni
after update on Decisione
for each row
begin
if new.Risultato = 'Positivo'
then if (select A.CodicePezzo from(
select P.CodicePezzo, P.QuantitaNecessaria, C.Fornitore
from PezziNecessari P join (
select CodicePezzo,Fornitore
from Catalogo
group by CodicePezzo) C
where P.CodicePezzo=C.CodicePezzo AND P.NROrdine=new.NROrdine) A
where A.CodicePezzo IN (select CodicePezzo from OrdinePendente)
LIMIT 1  ) IS NOT NULL
then DROP TEMPORARY TABLE IF EXISTS Nuovi_Pezzi;
CREATE TEMPORARY TABLE Nuovi_Pezzi (CodicePezzo nstring (4),QuantitaNecessaria int (2));
INSERT INTO Nuovi_Pezzi (CodicePezzo,QuantitaNecessaria)
select A.CodicePezzo,A.QuantitaNecessaria
from( select P.CodicePezzo, P.QuantitaNecessaria, C.Fornitore
from PezziNecessari P join (
select CodicePezzo,Fornitore
from Catalogo
group by CodicePezzo) C
where P.CodicePezzo=C.CodicePezzo AND P.NROrdine=new.NROrdine) A
where A.CodicePezzo IN (select CodicePezzo from OrdinePendente);
UPDATE OrdinePendente INNER JOIN Nuovi_Pezzi ON OrdinePendente.CodicePezzo=Nuovi_Pezzi.CodicePezzo
SET OrdinePendente.Quantita = OrdinePendente.Quantita + Nuovi_Pezzi.QuantitaNecessaria;
end if;
```

```
if  ( select A.CodicePezzo from(
select P.CodicePezzo, P.QuantitaNecessaria, C.Fornitore
from PezziNecessari P join (
select CodicePezzo,Fornitore
from Catalogo
group by CodicePezzo) C
where P.CodicePezzo=C.CodicePezzo AND P.NROrdine=new.NROrdine) A
where A.CodicePezzo NOT IN (select CodicePezzo from OrdinePendente)
LIMIT 1  ) IS NOT NULL
then INSERT INTO OrdinePendente (CodicePezzo,Quantita,Fornitore)
select A.CodicePezzo,A.QuantitaNecessaria,A.Fornitore
from( select P.CodicePezzo, P.QuantitaNecessaria, C.Fornitore
from PezziNecessari P join (
select CodicePezzo,Fornitore
from Catalogo
group by CodicePezzo ) C
where P.CodicePezzo=C.CodicePezzo AND P.NROrdine=new.NROrdine ) A
where A.CodicePezzo NOT IN (select CodicePezzo from OrdinePendente );
end if; end if; end; $$
delimiter ;
```

VII. Questo trigger before update on StoricoOrdine, si attiva quando viene inserita una manodopera (numero di ore di lavoro) diversa da 0.
Se questo è il caso si setta l'attributo DataFineLavori alla data corrente, poi imposta l'attributo CostoFinale uguale alla somma di Manodopera*25(€/h costo di un'ora di lavoro del meccanico) e il 110% del preventivo, il 10% in più è l'interesse dell'officina sui pezzi.

```
delimiter $$
create trigger CostoFinale
before update on StoricoOrdine
        for each row
begin
        if new.Manodopera <> 0
        then
        set new.CostoFInale = new.Manodopera*25+ new.Preventivo*1.1;
        set new.Data_fine_lavori=CURDATE();
        end if;
end; $$
delimiter ;
```

VIII. Questo trigger rispetto al trigger 6 si occupa di assegnare i pezzi arrivati dai fornitori agli ordini con priorità maggiore (data di ordinazione più vecchia), nel caso in cui la quantità arrivata non sia compatibile con la somma dei pezzi richiesti dagli ordini non assegna i pezzi, poiché tale quantità potrebbe essere o maggiore del necessario oppure minore rispetto la richiesta fatta.

```
delimiter $$
create trigger AssegnazioneNuoviPezzi
before update on OrdinePendente
        for each row
begin
        declare PRIOR int;
        declare ARRIVATI int;
        if new.Quantita < old.Quantita
        then
        set PRIOR = (select count(*) from PezziNecessari where CodicePezzo=new.CodicePezzo);
        set ARRIVATI = old.Quantita-new.Quantita;
        WHILE(
                select SUM(A.QuantitaNecessaria)
                from (
                        select CodicePezzo,QuantitaNecessaria
                        from PezziNecessari
                        where CodicePezzo=new.CodicePezzo AND QuantitaDisponibile=0
                        ORDER BY Richiesti
                        LIMIT PRIOR
                        ) A
                 ) <> ARRIVATI AND PRIOR >0
        DO
        set PRIOR=PRIOR-1;
        end while;
        if PRIOR > 0
        then
                update PezziNecessari
                SET QuantitaDisponibile=QuantitaNecessaria
                where CodicePezzo=new.CodicePezzo AND QuantitaDisponibile=0
                ORDER BY Richiesti
                LIMIT PRIOR ;
        else
                SET new.Quantita=old.Quantita;
        end if;
end if;
end; $$
delimiter ;
```