Università degli Studi di Salerno

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata

Corso di Laurea in Ingegneria Informatica

Basi di Dati 2020/2021
Canale A-H

Project Work

Bike rental

Gruppo n. **03 – AH**

WP	Cognome e Nome	Matricola	e-mail	Responsabile
1	De Gruttola Andrea	0612704470	a.degruttola@studenti.unisa.it	
2	Grimaldi Salvatore	0612704347	s.grimaldi29@studenti.unisa.it	
3	Di Mauro Enrico Maria	0612704486	e.dimauro5@studenti.unisa.it	X
4	Cuzzocrea Allegra	0612704414	a.cuzzocrea2@studenti.unisa.it	

Anno accademico 2020-2021

Sommario

1. Descrizione della realtà di interesse ... 3

2. Analisi delle specifiche ... 4

2.1. Glossario dei termini ... 4

2.2. Strutturazione dei requisiti in frasi ... 5

2.2.1. Frasi di carattere generale ... 5
2.2.2. Frasi relative a Categoria ... 5
2.2.3. Frasi relative a Modello ... 5
2.2.4. Frasi relative a tipi specifici di Modello ... 5
2.2.5. Frasi relative a Bici ... 5
2.2.6. Frasi relative a Cliente ... 5
2.2.7. Frasi relative a Pagamento ... 5
2.2.8. Frasi relative a Noleggio ... 5
2.2.9. Frasi relative a Punto vendita ... 5

2.3. Identificazione delle operazioni principali ... 6

3. Progettazione Concettuale ... 7

3.1. Schema Concettuale ... 7

3.1.1. Note sullo schema E-R ... 8

3.2. Design Pattern ... 8

3.2.1. Pattern Instance-Of ... 8
3.2.2. Pattern Reificazione-Attributo Addetto ... 8
3.2.3. Pattern Reificazione-Attributo Responsabile ... 9
3.2.4. Pattern Reificazione-Relazione Acquisto ... 9

3.3. Dizionario dei Dati ... 10

3.4. Regole Aziendali ... 12

4. Progettazione Logica ... 14

4.1. Ristrutturazione Schema Concettuale ... 14

4.1.1. Analisi delle Prestazioni ... 14

4.2. Analisi delle ridondanze ... 15

4.2.1. Analisi della ridondanza 1: Totale ... 16

4.3. Eliminazione delle generalizzazioni ... 18

4.3.1. Generalizzazione *Cliente* ... 18
4.3.2. Generalizzazione *Modello* ... 18
4.3.3. Generalizzazione *Addetto* ... 19

4.4. Partizionamento/Accorpamento Entità e Associazioni ... 19

4.4.1. Attributo composto *Residenza (CLIENTE)* ... 19
4.4.2. Attributo multivalore Telefono (ADDETTO) ... 20

4.5. Scelta degli identificatori principali ... 20

4.5.1. Bici ... 20
4.5.2. PuntoVendita ... 20
4.5.3. Responsabile ... 21
4.5.4. Acquisto ... 21
4.5.5. Pagamento ... 21

4.6. Schema ristrutturato finale ... 22

4.7. Schema logico ... 23

4.8. Documentazione dello schema logico ... 24

4.8.1. Vincoli introdotti da ristrutturazione e traduzione 24
4.8.2. Rappresentazione Grafica 25

5. Normalizzazione 26

6. Script Creazione e Popolamento Database 28

7. Query SQL 39

7.1. Query con operatore di aggregazione e join: Studenti poveri 39

7.2. Query nidificata complessa: Punti vendita completi 39

7.3. Query insiemistica: Resoconto spese clienti 39

7.4. Altre query 40
7.4.1. Clienti lusso 40
7.4.2. Noleggi in corso 40

8. Viste 41

8.1. Vista: *Categorie noleggiate* 41
8.1.1. Query con Vista: Categoria privilegiata 41

8.2. Vista: Numero noleggi per cliente 41
8.2.1. Query con vista: Cliente affezionato 41

8.3. Viste: Altri nei PV, Anziani nei PV, Studenti nei PV, ... 42
8.3.1. Query con viste: Noleggi Bologna 43

9. Trigger 44

9.1. Trigger inizializzazione 44
9.1.1. Trigger1: OreRimanentiFirstTime 44
9.1.2. Trigger2: aggiornamentoOreRimanenti 44
9.1.3. Trigger3: checkTotalePagamento 46
9.1.4. Trigger4: PrelievoBici 47
9.1.5. Trigger5: RiconsegnaBici 47

9.2. Trigger per vincoli aziendali 48
9.2.1. Trigger1: CAPpuntoVenditaonlyDigits 48
9.2.2. Trigger2: CAPclienteOnlyDigits 48
9.2.3. Trigger3: MatricolaAddettoOnlyDigits 48
9.2.4. Trigger4: CodicePrepagataOnlyDigits 49
9.2.5. Trigger5: CodiceNoleggioOnlyDigits 49
9.2.6. Trigger6: CartaCreditoOnlyDigits 49
9.2.7. Trigger7: NumTelefonoOnlyDigits 50
9.2.8. Trigger8: ResponsabileAncheAssegnato 50
9.2.9. Trigger9: NOLEGGIOconsistency 50
9.2.10. Trigger10: PRELIEVObeforeRICONSEGNA 51
9.2.11. Trigger11: RICONSEGNAbeforePAGAMENTO 51
9.2.12. Trigger12: dateconsistency 51
9.2.13. Trigger13: pagamentiConCardPrepagata 52
9.2.14. Trigger14: OreRimanentiCardAcquistataMustBeNotNull 52
9.2.15. Trigger15: oreRimanentiInserimentoCard 53
9.2.16. Trigger16: ImpedisciAggiornamentoTariffa 53

1. Descrizione della realtà di interesse

Titolo: **Bike rental**

Bike rental: Una società di bike rental ha più punti vendita nei quali è possibile noleggiare una bici. Si vuole realizzare un sistema per la gestione di tale servizio. Si realizzi il database a supporto di tale sistema con le seguenti specifiche:
•La società gestisce diversi punti vendita nel quale è possibile noleggiare una bici e riconsegnarla (la bici può essere riconsegnata in un punto vendita diverso da quello in cui è stata noleggiata). Per il punto vendita è di interesse conoscere la locazione, il responsabile, le bici attualmente disponibili.
•Esistono diversi modelli di bici: da città, mountain bike, bambino, a pedalata assistita, elettrica. Per tutti i modelli è di interesse la marca, il modello, la grandezza delle ruote, il colore. Per i modelli elettrici, è di interesse conoscere la durata nominale della batteria.
•I clienti che noleggiano una bici devono essere identificati e registrati. In particolare, viene registrato un documento, la residenza del cliente, e le informazioni sulla carta di credito.
•Le bici possono essere noleggiate ad ore oppure per un giorno intero. Le tariffe dipendono dal tipo di bici noleggiata e dal giorno della settimana. Per il noleggio giornaliero, il noleggio viene pagato in anticipo. Per il noleggio ad ore, il cliente decide quante ore pagare. In caso di consegna in ritardo, saranno addebitate le ore aggiuntive. Per ogni noleggio, si vuole conoscere data e ora di inizio noleggio, la bici noleggiata, i dati del cliente, la data di riconsegna, il luogo di inizio noleggio, il luogo di riconsegna.
•Sia per noleggi giornalieri che ad ore, è prevista inoltre una cauzione che verrà restituita al termine del noleggio se la bici è consegnata in buono stato. Se la bici è danneggiata, la cauzione non è restituita. All'atto della consegna, quindi, l'addetto registrerà eventuali danni e inserirà la motivazione per cui la cauzione non è restituita.
•È possibile, inoltre, acquistare delle card prepagate che offrono sconti sul noleggio della bici. Tali card consentono di noleggiare bici per un certo numero di ore o per un certo numero di giorni. Ad ogni noleggio, è necessario decurtare il numero di ore o di noleggi effettuati fino all'esaurimento della card. Esistono tariffe agevolate per studenti e per anziani.

2. Analisi delle specifiche

Workpackage	***Task***	***Responsabile***
WP0	Analisi delle specifiche	Intero Gruppo

2.1. Glossario dei termini

	Termine	**Descrizione**	**Sinonimi**	**Collegamenti**
1	Categoria	Categoria di bici (da città, mountain bike, …)	Modello	Modello, Bici
2	Modello	Modello di una bici (identificato da marca e nome)	Tipo	Categoria, Bici, Pagamento
3	Bici	Istanza di un modello. È il mezzo di trasporto possibile da noleggiare in un punto vendita	-	Categoria, Modello, Cliente, Noleggio, Punto Vendita
4	Cliente	Persona che noleggia una bici	Studente, Anziano	Bici, Noleggio, Pagamento
5	Noleggio	Cessione temporanea della bici, a cui è associato un pagamento. Può essere giornaliero o ad ore	-	Bici, Cliente, Pagamento, Punto Vendita
6	Pagamento	Dipende dalla tariffa. Essa dipende a sua volta da: cliente, modello della bici e giorno della settimana	-	Modello, Cliente, Noleggio
7	Punto Vendita	Luogo fisico in cui si prelevano e riconsegnano le bici	Luogo di inizio noleggio, Luogo di riconsegna	Bici, Noleggio

Tabella 1. Glossario dei Termini

2.2. Strutturazione dei requisiti in frasi

2.2.1. Frasi di carattere generale

Una società di bike rental ha più punti vendita nei quali è possibile noleggiare una bici.
Si vuole realizzare il database a supporto di un sistema per la gestione di tale servizio.

2.2.2. Frasi relative a Categoria

Per la categoria vogliamo rappresentare se è da città, mountain bike, bambino, a pedalata assistita o elettrica.

2.2.3. Frasi relative a Modello

Per il modello vogliamo rappresentare la marca, la categoria, la grandezza delle ruote ed il colore.

2.2.4. Frasi relative a tipi specifici di Modello

Per i modelli elettrici vogliamo rappresentare la durata nominale della batteria.

2.2.5. Frasi relative a Bici

Per le bici vogliamo rappresentare il modello e lo stato.

2.2.6. Frasi relative a Cliente

Per i clienti vogliamo rappresentare un documento, la residenza e le informazioni sulla carta di credito.

2.2.7. Frasi relative a Pagamento

Per il pagamento vogliamo rappresentare il totale, che dipende dalla tariffa. Per quest'ultima, in particolare, sono di interesse: il cliente (esistono tariffe agevolate per studenti e anziani), il modello di bici ed il giorno della settimana.
Nel caso in cui il noleggio sia giornaliero, il pagamento avviene in anticipo.
Nel caso in cui il noleggio sia ad ore, il cliente decide quante ore pagare. In caso di consegna in ritardo, saranno addebitate le ore aggiuntive.

2.2.8. Frasi relative a Noleggio

Per il noleggio vogliamo rappresentare se è giornaliero oppure ad ore, data e ora di inizio noleggio, la bici noleggiata, i dati del cliente, la data di riconsegna, il luogo di inizio noleggio, il luogo di riconsegna. Inoltre, sono di interesse la cauzione e la possibilità di acquistare delle card prepagate che offrono sconti sul noleggio della bici

2.2.9. Frasi relative a Punto vendita

Per il punto vendita vogliamo rappresentare la locazione, il responsabile, gli eventuali addetti e le bici attualmente disponibili.

2.3. Identificazione delle operazioni principali

Operazione 1: *Inserisci un nuovo cliente indicando tutti i suoi dati (in media 50 volte al mese)*

Operazione 2: *Inserisci una nuova bici indicando tutti i suoi dati (in media 20 volte al mese)*

Operazione 3: *Rimuovi una bici (in media 20 volte al mese)*

Operazione 4: *Inserisci un nuovo noleggio (in media 3000 volte al mese)*

Operazione 5: *Per ogni punto vendita, stampa tutte le bici presenti (in media 30 volte al mese)*

Operazione 6: *Stampa il ricavo mensile (in media 1 volta al mese)*

Operazione 7: *Effettua una statistica su tutti i clienti con tutte le informazioni su di essi e sui modelli di bici da loro noleggiate (in media 1 volta al mese)*

3. Progettazione Concettuale

Workpackage	*Task*	*Responsabile*
WP1	Progettazione Concettuale	De Gruttola Andrea

3.1. Schema Concettuale

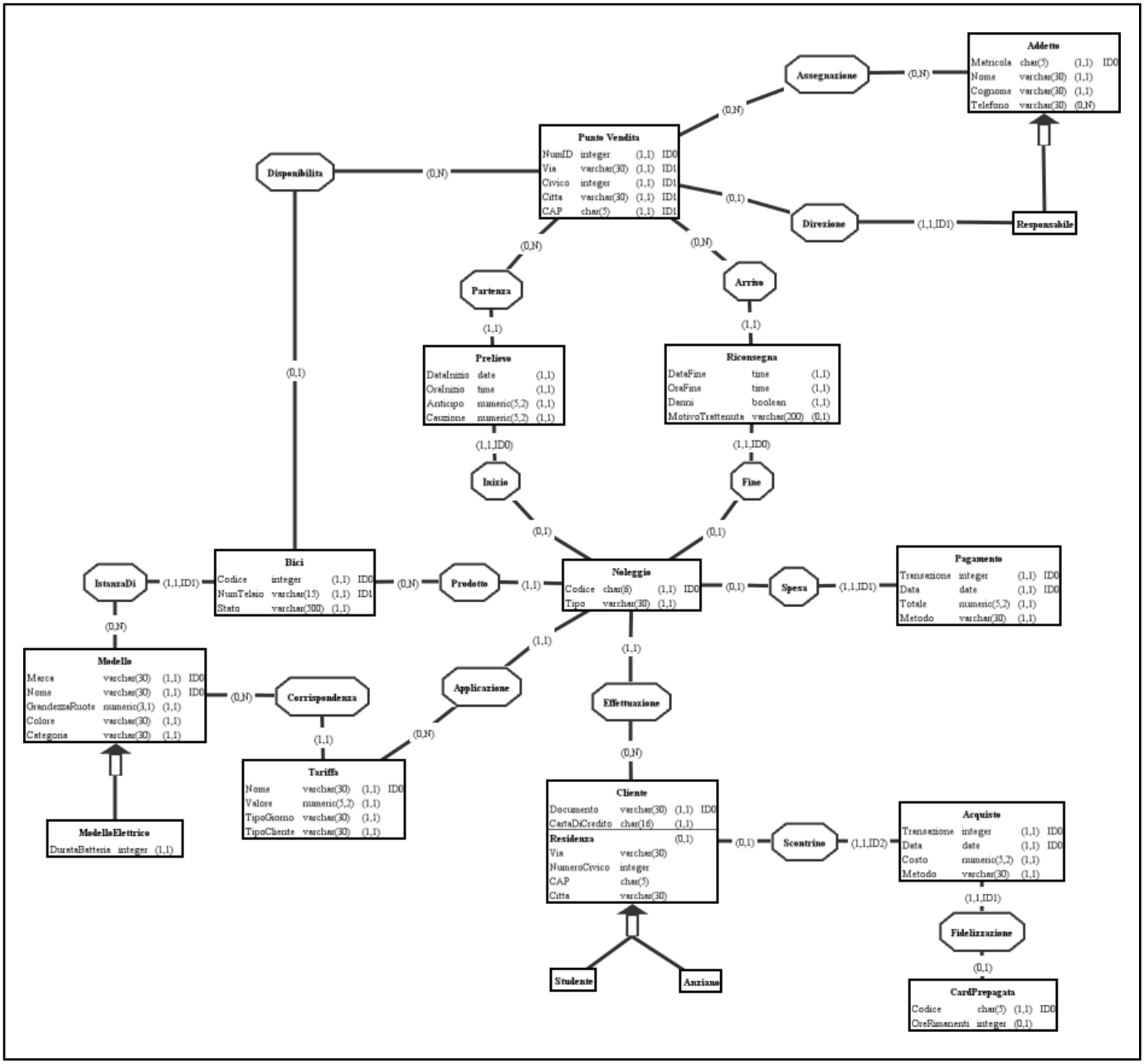


Figura 1. Schema E-R

Note*: Come concordato con il prof D'Aniello, abbiamo scelto di considerare solamente card prepagate *ad ore*, dal momento che quelle giornaliere si basano ugualmente sulle ore.
Assumiamo che la numerazione delle transazioni in Pagamento ed Acquisto ricominci da 0 ogni giorno, ragion per cui l'identificativo di queste due entità è costituito anche dalla data.
Si osservi che la data del Pagamento potrebbe anche non corrispondere alla data di riconsegna, ragion per cui è essenziale la presenza dell'omonimo attributo nell'entità Pagamento. La data di quest'ultimo, in effetti, è una sua caratteristica fondamentale, in quanto contribuisce anche all'individuazione dell'identificatore interno.
Se il noleggio è *ad ore*, anticipo può essere uguale a 0, in quanto è il cliente a decidere quante ore anticipare, indipendentemente dal metodo di pagamento che poi verrà adottato. In particolare, il pagamento potrà essere effettuato con card prepagata solo se non è stato versato un anticipo in soldi.

Se, invece, il noleggio è *giornaliero*, esso dovrà essere pagato in anticipo, in contanti o con carta di credito. Si assume, infatti, che nel caso di noleggio giornaliero, un anticipo uguale a 0 corrisponde alla volontà del cliente di pagare con card prepagata (mediante decurtazione di ore) al termine dello stesso, ovvero: se l'anticipo è uguale a 0, il cliente pagherà sicuramente con card prepagata.
Si osservi che si è scelto di non rappresentare Via, Civico, Citta, CAP in PuntoVendita come un attributo composto, poiché si è ravvisata la possibilità di sfruttarli per costituire un ulteriore identificativo dell'omonima entità. È più corretto, infatti, che tutti gli attributi che costituiscono un identificativo, nonostante ne esista un altro che si candida ad essere il principale (ovvero NumID), debbano essere sempre valorizzati. Se avessimo rappresentato tale insieme di attributi mediante un attributo composto, si sarebbe addirittura potuta presentare la possibilità che nessuno di essi fosse valorizzato.
Si è scelto che un cliente possa acquistare una sola card per volta, ovvero un cliente non può avere più card prepagate contemporaneamente.
Si osservi che la cardinalità minima della relazione Assegnazione dal lato di Addetto è 0 in quanto si ammette la possibilità che per un certo periodo un addetto non sia assegnato ad alcun punto vendita.
Si osservi che non deve essere possibile creare un noleggio di una bici che è stata noleggiata in precedenza e non è stata ancora riconsegnata.

3.1.1. Note sullo schema E-R

Abbiamo utilizzato una strategia mista. Siamo partiti dalla realizzazione di uno schema scheletro e successivamente lo abbiamo espanso e raffinato. Abbiamo preferito questo tipo di strategia in quanto essa fornisce flessibilità e facilità di applicazione.

3.2. Design Pattern

3.2.1. Pattern Instance-Of

La scelta dell'utilizzo del pattern instance-of risulta immediata alla luce del significato delle entità Modello e Bici. Le occorrenze della seconda, di fatti, sono istanze di occorrenze della prima.

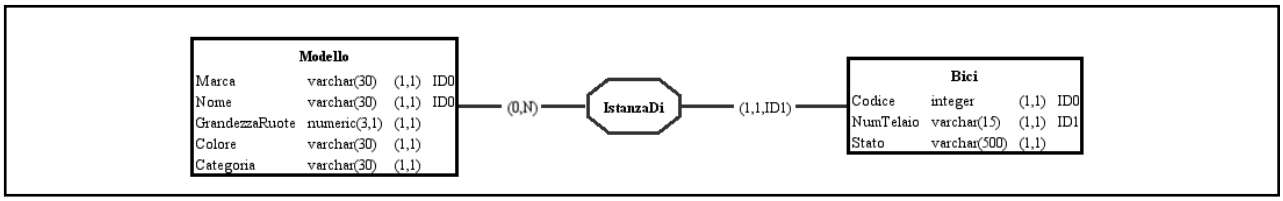


Figura 2. Schema successivo all'applicazione del Pattern INSTANCE-OF.

3.2.2. Pattern Reificazione-Attributo Addetto

Abbiamo ritenuto particolarmente utile effettuare la reificazione dell'attributo multivalore Addetto, originariamente presente nell'entità PuntoVendita. La ragione è che il concetto di Addetto è caratterizzato da proprietà significative, la cui rappresentazione può essere di particolare interesse, quali la matricola, il nome, il cognome ed il numero di telefono. L'Addetto è effettivamente dotato di esistenza autonoma ed è collegato al PuntoVendita attraverso l'associazione Assegnazione.

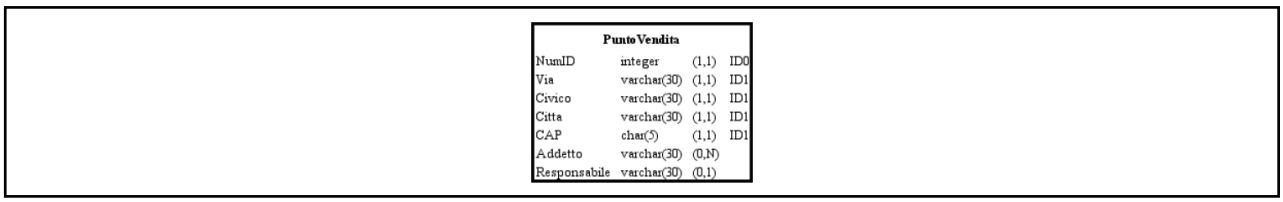


Figura 3. Schema precedente all'applicazione del Pattern REIFICAZIONE-ATTRIBUTO.

Si osservi che Responsabile è opzionale, in quanto si è assunto che al momento della creazione di un punto vendita non debba essere per forza di cose noto il suo Responsabile. Sia per quest'ultimo sia per Addetto si era originariamente scelto di adoperare *varchar(30)* come dominio, in quanto si prevedeva di inserirvi semplicemente il cognome.

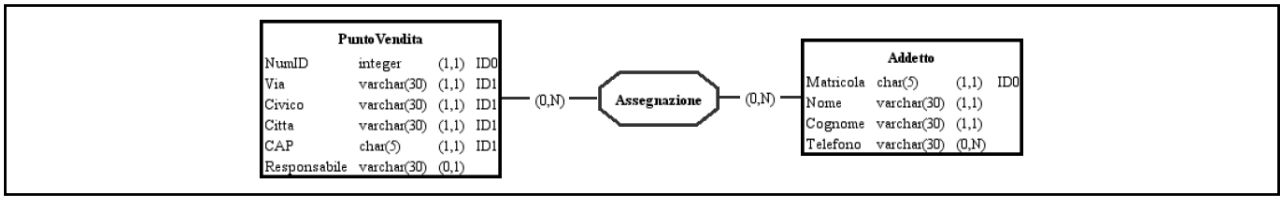


Figura 4. Schema successivo all'applicazione del Pattern REIFICAZIONE-ATTRIBUTO.

3.2.3. Pattern Reificazione-Attributo Responsabile

Partendo dall'assunzione che un Responsabile sia un particolare tipo di Addetto (di cui è stata effettuata la corrispondente specializzazione), abbiamo ritenuto particolarmente utile applicare il pattern di reificazione all'attributo Responsabile, originariamente contenuto nell'entità PuntoVendita. Responsabile è chiaramente collegato al corrispondente punto vendita attraverso l'introduzione dell'associazione Direzione.

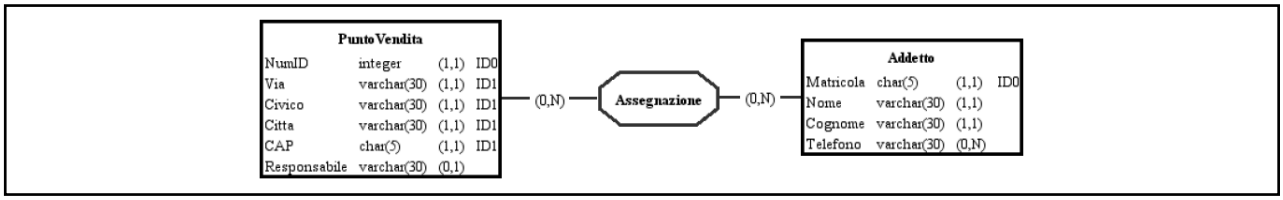


Figura 5. Schema precedente all'applicazione del Pattern REIFICAZIONE-ATTRIBUTO.

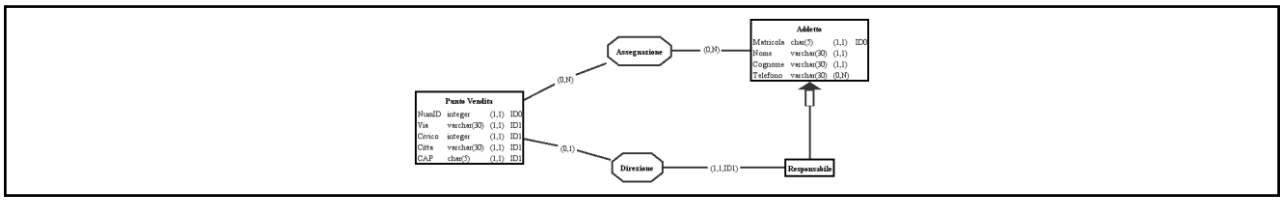


Figura 6. Schema successivo all'applicazione del Pattern REIFICAZIONE-ATTRIBUTO.

3.2.4. Pattern Reificazione-Relazione Acquisto

Data la rilevanza del concetto di acquisto, inizialmente rappresentato da una relazione, abbiamo ritenuto opportuno effettuare la reificazione di quest'ultima. È importante, infatti, tracciare l'acquisto delle card prepagate, tenendo conto del loro costo e della data in cui avviene la transazione.

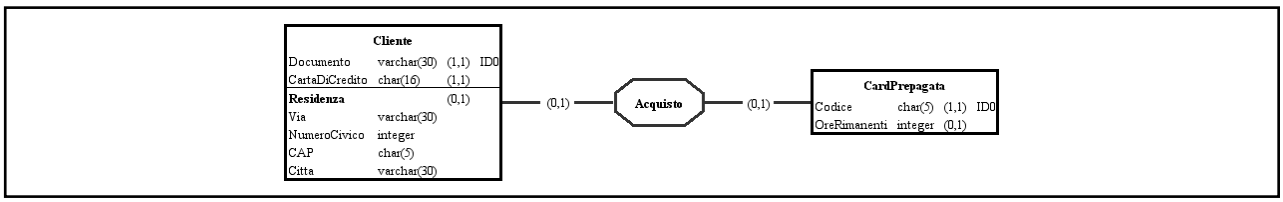


Figura 7. Schema precedente all'applicazione del Pattern REIFICAZIONE-RELAZIONE

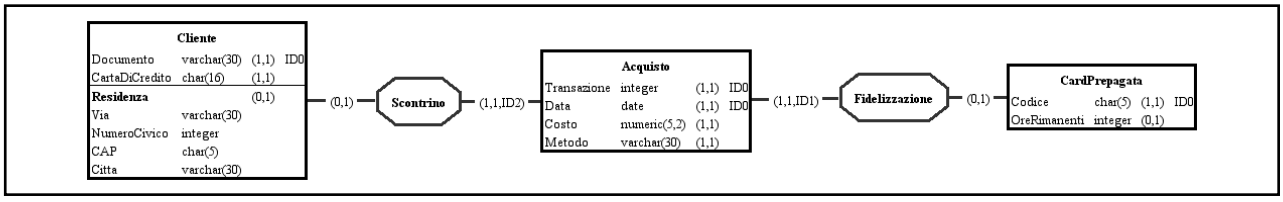


Figura 8. Schema successivo all'applicazione del Pattern REIFICAZIONE-RELAZIONE

3.3. Dizionario dei Dati

Entità	Descrizione	Attributi	Identificatore
PuntoVendita	Luogo fisico in cui si noleggiano e riconsegnano le bici	NumID, Via, Civico, Citta, CAP	NumID; Via, Civico, Citta, CAP
Bici	Istanza di un modello. È il mezzo di trasporto che è possibile noleggiare in un punto vendita	Codice, NumTelaio, Stato	Codice; NumTelaio, Modello
Noleggio	Cessione temporanea di una bici. Può essere giornaliero o ad ore	Codice, Tipo	Codice
Cliente	Persona che noleggia una bici	Documento, CartaDiCredito, Residenza	Documento
Modello	Modello di una bici	Marca, Nome, GrandezzaRuote, Colore, Categoria	Marca, Nome
ModelloElettrico	***Specializzazione di Modello*** Specifico modello di bici dotato di motore elettrico	DurataBatteria	Marca, Nome
Tariffa	Costo orario per il noleggio di una bici	Nome, Valore, TipoGiorno, TipoCliente	Nome
Pagamento	Transazione associata ad un noleggio comprovata dal rilascio di una ricevuta	Transazione, Data, Totale, Metodo	Transazione, Data; Noleggio
CardPrepagata	Carta acquistabile da un cliente	Codice, OreRimanenti	Codice
Studente	***Specializzazione di Cliente*** Cliente che va a scuola o all'università		Documento
Anziano	***Specializzazione di Cliente*** Cliente con età superiore a 65		Documento
Addetto	Persona che lavora in uno o più punti vendita	Matricola, Nome, Cognome, Telefono	Matricola
Responsabile	***Specializzazione di Addetto*** Addetto che dirige un punto vendita		Matricola; PuntoVendita
Acquisto	Transazione che avviene quando un cliente acquista una card prepagata	Transazione, Data, Costo, Metodo	Transazione, Data; CardPrepagata; Cliente
Prelievo	Operazione di prelevamento della bici. Dà il via al noleggio	DataInizio, OraInizio, Anticipo, Cauzione	Noleggio
Riconsegna	Operazione di restituzione della bici. Determina la fine del noleggio	DataFine, OraFine, Danni, MotivoTrattenuta	Noleggio

Tabella 2. Dizionario dei dati – Entità

Relazioni	Descrizione	Entità Coinvolte	Attributi
Disponibilita	Associa le bici disponibili ad un punto vendita	PuntoVendita, Bici	
IstanzaDi	Associa una bici al modello corrispondente	Modello, Bici	
Prodotto	Associa il noleggio alla bici noleggiata	Noleggio, Bici	
Effettuazione	Associa al noleggio il cliente che lo sta effettuando	Noleggio, Cliente	
Scontrino	Associa all'acquisto della card prepagata il cliente che lo sta effettuando	Cliente, Acquisto	
Fidelizzazione	Associa all'acquisto la card prepagata	Acquisto, CardPrepagata	
Assegnazione	Associa l'addetto ai punti vendita a cui è assegnato	PuntoVendita, Addetto	
Direzione	Associa il responsabile al punto vendita che dirige	PuntoVendita, Responsabile	
Applicazione	Associa la tariffa al noleggio a cui è applicato	Tariffa, Noleggio	
Spesa	Associa il pagamento al noleggio a cui è riferito	Pagamento, Noleggio	
Corrispondenza	Associa il modello alle tariffe che vi si riferiscono	Modello, Tariffa	
Inizio	Associa il noleggio alla corrispondente operazione di prelevamento della bici	Noleggio, Prelievo	
Partenza	Associa il prelievo al punto vendita in cui è effettuato	PuntoVendita, Prelievo	
Arrivo	Associa la riconsegna al punto vendita in cui è effettuata	PuntoVendita, Riconsegna	
Fine	Associa il noleggio alla corrispondente operazione di consegna della bici	Noleggio, Riconsegna	

Tabella 3. Dizionario dei dati - Relazioni

Workpackage	*Task*	*Responsabile*
WP4	Regole Aziendali	Cuzzocrea Allegra

3.4. Regole Aziendali

Regole di Vincolo
(RV1) La durata della batteria del modello elettrico deve essere maggiore di zero
(RV2) La categoria del modello deve essere in {DaCitta, MountainBike, Bambino, APedalataAssistita, Elettrica}
(RV3) La grandezza delle ruote del modello deve essere maggiore di zero
(RV4) Il codice della bici deve essere maggiore di zero
(RV5) L'identificativo del punto vendita deve essere maggiore di zero
(RV6) Il civico del punto vendita deve essere maggiore di zero
(RV7) Il CAP del punto vendita deve essere costituito da 5 cifre
(RV8) I numeri di telefono di un addetto devono essere costituiti da sole cifre
(RV9) La matricola dell'addetto deve essere costituita da 5 cifre
(RV10) Il responsabile di un punto vendita deve anche essere assegnato a quel punto vendita
(RV11) Una riconsegna senza segnalazione di danni non deve riportare la motivazione della trattenuta, mentre una riconsegna con segnalazione di danni deve riportarla
(RV12) Il codice di un noleggio deve essere costituito da 6 cifre
(RV13) La data di fine di riconsegna di un noleggio deve essere uguale o successiva a quella di inizio del prelievo dello stesso
(RV14) Per noleggi che terminano nella stessa giornata in cui sono iniziati, l'ora di fine di riconsegna deve essere successiva a quella di inizio di prelievo
(RV15) Il tipo del noleggio deve essere in {AdOre, Giornaliero}
(RV16) Il valore della tariffa deve essere maggiore di zero ed indica il costo orario
(RV17) Il tipo di giorno della tariffa deve essere in {Feriale, Festivo}
(RV18) Il tipo di cliente della tariffa deve essere in {Studente, Anziano, Altro}
(RV19) La transazione del pagamento deve essere maggiore di zero
(RV20) La cauzione del prelievo deve essere maggiore di zero
(RV21) Per pagamenti effettuati con card prepagata l'anticipo ed il totale devono essere zero,
(RV22) Per pagamenti non effettuati con card prepagata il totale deve essere diverso da 0
(RV23) Il totale del pagamento deve essere maggiore o uguale dell'anticipo
(RV24) Il totale del pagamento deve essere maggiore o uguale di zero
(RV25) Il metodo del pagamento deve essere in {Contanti, CartaDiCredito, CardPrepagata}
(RV26) Il CAP del cliente deve essere costituito da 5 cifre
(RV27) La carta di credito di un cliente deve essere costituita da 16 cifre
(RV28) Il civico del cliente deve essere maggiore di zero
(RV29) La transazione dell'acquisto deve essere maggiore di zero
(RV30) Il costo dell'acquisto deve essere maggiore di zero
(RV31) Il metodo dell'acquisto deve essere in {Contanti, CartaDiCredito}
(RV32) Il codice della card prepagata deve essere costituito da 5 cifre
(RV33) Le ore rimanenti della card prepagata devono essere maggiori o uguali a zero
(RV34) Per noleggi giornalieri con anticipo uguale a 0, il metodo del pagamento deve essere CardPrepagata
(RV35) Per noleggi giornalieri con anticipo diverso da 0, l'anticipo stesso deve essere uguale alla tariffa moltiplicata per 24
(RV36) OreRimanenti della card prepagata deve essere valorizzato solo in seguito al suo acquisto da parte di un cliente

Tabella 4. Regole di vincolo

Regole di derivazione
(RD1) Il totale del pagamento di un noleggio è uguale a: **Totale = [24 * (DataFine - DataInizio) + (OraFine - OraInizio)] * Valore** dove: • (DataFine – DataInizio) restituisce un intero che rappresenta il numero di giorni che intercorre tra le due date; • (OraFine - OraInizio) restituisce un intero prodotto da un'approssimazione: per eccesso quando i minuti sono maggiori o uguali di 30, per difetto altrimenti. • Per Valore si intende il valore della Tariffa, ossia il prezzo orario. **(RD2)** Le ore rimanenti di una card prepagata, al momento dell'acquisto, si calcolano attraverso la formula: **OreRimanenti = Costo / TariffaCard** dove: • Costo è un attributo dell'entità Acquisto e si riferisce al prezzo della card prepagata • TariffaCard è la tariffa oraria applicata ai noleggi pagati con card prepagata. Il suo valore è uguale a 1.10 **Nota***: L'attributo Costo all'interno di Acquisto non è ridondante poiché è possibile ottenerlo solamente tramite il primo valore di OreRimanenti. Al primo utilizzo della card prepagata, infatti, OreRimanenti viene sovrascritto secondo la regola di derivazione (RD3). **(RD3)** Le ore rimanenti di una card prepagata, a partire dal primo noleggio effettuato con quella card prepagata, si calcolano attraverso la formula: **OreRimanenti = OreRimanenti - [24 * (DataFine - DataInizio) + (OraFine - OraInizio)]** dove: • OreRimanenti al secondo membro si riferiscono alle ore presenti sulla card prepagata prima di effettuare l'i-esimo noleggio; • OreRimanenti al primo membro si riferiscono alle ore presenti sulla card prepagata dopo aver effettuato l'i-esimo noleggio.

Tabella 5. Regole di derivazione

4. Progettazione Logica

Workpackage	***Task***	***Responsabile***
WP2	Progettazione Logica	Grimaldi Salvatore

4.1. Ristrutturazione Schema Concettuale

4.1.1. Analisi delle Prestazioni

4.1.1.1. *Tavola dei volumi*

Concetto	**Tipo**	**Volume**
PuntoVendita	E	5
Bici	E	500
Noleggio	E	18000
Cliente	E	350
Modello	E	10
ModelloElettrico	E	2
Tariffa	E	60
Pagamento	E	18000
CardPrepagata	E	110
Studente	E	105
Anziano	E	70
Addetto	E	15
Responsabile	E	5
Acquisto	E	110
Disponibilita	R	400
IstanzaDi	R	500
Prelievo	E	18000
Riconsegna	E	18000
Prodotto	R	18000
Effettuazione	R	18000
Scontrino	R	110
Fidelizzazione	R	110
Assegnazione	R	25
Direzione	R	5
Applicazione	R	18000
Spesa	R	18000
Corrispondenza	R	60
Inizio	R	18000
Fine	R	18000
Partenza	R	18000
Arrivo	R	18000

Tabella 6. Tavola dei volumi

***Note:** Abbiamo considerato che il regime venga raggiunto dopo 6 mesi.
Abbiamo assunto l'esistenza di 5 punti vendita e 10 clienti abituali per ciascuno di essi. Ogni mese si aggiungono in media altri 10 clienti per punto vendita, pertanto i clienti sono circa 50 + 50 * 6 = 350.
Assumiamo che il 30% dei clienti sia costituito da studenti ed il 20% da anziani.

Assumiamo che in media avvengano 100 noleggi giornalieri, quindi in 6 mesi il numero di noleggi è circa 100 * 30 * 6 = 18000.
Per ogni categoria esistono 2 Modelli differenti per un totale di 10 Modelli.
Le bici sono in totale 500, poiché 50 per ogni modello, ed equamente distribuite nei punti vendita. A regime, tenendo conto dei noleggi giornalieri effettuati, le bici effettivamente disponibili (pronte per essere noleggiate) sono circa 400.
Ipotizziamo che in ogni punto vendita lavorino 3 addetti di cui 1 responsabile, per un totale di 15 addetti di cui 5 responsabili. Si osservi che un addetto è assegnato a più punti vendita, in quanto nel corso della stessa settimana può prestare servizio in più punti vendita, per ipotesi 2.
Le assegnazioni degli addetti, quindi, saranno 25 poiché date dalla somma delle assegnazioni di ciascun addetto non responsabile a 2 punti vendita e quelle di ciascun addetto responsabile ad un punto vendita.
Le tariffe sono 10 * 3 * 2 = 60, dove 10 sono i modelli, 3 sono i tipi di cliente e 2 sono le tipologie di giorno.
Assumiamo che tutti i clienti abituali ed il 20% dei nuovi clienti acquisiti nell'arco di 6 mesi abbiano acquistato una card, quindi le card prepagate sono circa 50 + 20% * 50 * 6 = 110.

4.1.1.2. *Tavola delle operazioni*

Operazione 8: *Stampa la somma dei totali dei pagamenti dei noleggi effettuati per ciascun cliente (in media 1 volta al mese)*
Operazione 9: *Stampa il totale del pagamento di un noleggio (in media 3000 volte al mese)*
Operazione 10: *Creazione di un nuovo pagamento (in media 3000 volte al mese)*

Operazione	Tipo	Frequenza
Operazione 1: Inserimento cliente	I	50 al mese
Operazione 2: Inserimento bici	I	20 al mese
Operazione 3: Rimuovi bici	I	20 al mese
Operazione 4: Inserimento noleggio	I	3000 al mese
Operazione 5: Stampa bici	I	30 al mese
Operazione 6: Stampa ricavo mensile	B	1 al mese
Operazione 7: Statistica info clienti	B	1 al mese
Operazione 8: Stampa somma totale pagamenti noleggi per cliente	I	1 al mese
Operazione 9: Stampa totale pagamento noleggio	I	3000 al mese
Operazione 10: Creazione pagamento	I	3000 al mese

Tabella 7. Tavola delle operazioni

4.2. Analisi delle ridondanze

- Ridondanza 1: Totale (PAGAMENTO). Il totale di un pagamento si ottiene dalla seguente formula: **Totale = [24 * (DataFine - DataInizio) + (OraFine - OraInizio)] * Valore**, dove DataFine, OraFine sono attributi dell'entità RICONSEGNA, DataInizio, OraInizio sono attributi dell'entità PRELIEVO e Valore è attributo dell'entità TARIFFA.
  TIPO: Attributo derivabile da altre entità.

4.2.1. Analisi della ridondanza 1: Totale

- **Operazione 8:** Stampa somma totale pagamenti noleggi per cliente

Con Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
CLIENTE	E	350	L
EFFETTUAZIONE	R	18000	L
NOLEGGIO	E	18000	L
SPESA	R	18000	L
PAGAMENTO	E	18000	L

Senza Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
CLIENTE	E	350	L
EFFETTUAZIONE	R	18000	L
NOLEGGIO	E	18000	L
APPLICAZIONE	R	18000	L
TARIFFA	E	18000	L
INIZIO	R	18000	L
PRELIEVO	E	18000	L
FINE	R	18000	L
RICONSEGNA	E	18000	L

- **Operazione 9:** Stampa totale pagamento noleggio

Con Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
NOLEGGIO	E	1	L
SPESA	R	1	L
PAGAMENTO	E	1	L

Senza Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
NOLEGGIO	E	1	L
APPLICAZIONE	R	1	L
TARIFFA	E	1	L
INIZIO	R	1	L
PRELIEVO	E	1	L
FINE	R	1	L
RICONSEGNA	E	1	L

- **Operazione 10:** Creazione pagamento

Con Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
NOLEGGIO	E	1	L
APPLICAZIONE	R	1	L
TARIFFA	E	1	L
INIZIO	R	1	L
PRELIEVO	E	1	L
FINE	R	1	L
RICONSEGNA	E	1	L
SPESA	R	1	S
PAGAMENTO	E	1	S

Senza Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
PAGAMENTO	E	1	S
SPESA	R	1	S

4.2.1.1. *Valutazione della ridondanza 1*

Considerando la frequenza delle operazioni e moltiplicando per 2 gli accessi in scrittura (poiché tipicamente più onerosi di quelli in lettura) si ottiene quanto segue:

- *In presenza di ridondanza* il costo delle operazioni è di circa **114350** accessi mensili; l'occupazione di memoria aggiuntiva mensile è di circa 72000 byte (4 byte * 18000 noleggi a regime);
- In *assenza di ridondanza* il costo delle operazioni è di **177350** accessi mensili;

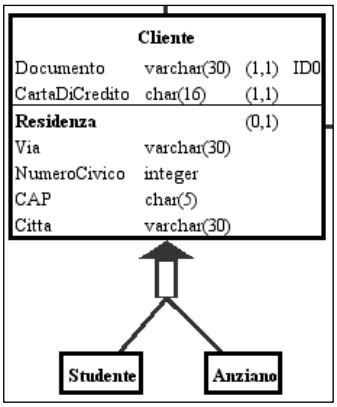
Si decide di mantenere la ridondanza in quanto riduce il numero di accessi e la memoria occupata aggiuntiva non è eccessiva. Il totale, inoltre, è un attributo fondamentale che caratterizza un pagamento, il che giustifica ulteriormente la scelta fatta.

Nota: Si osservi che un'altra ridondanza è rappresentata dall'associazione *Applicazione* tra le entità Tariffa e Noleggio. Essa, infatti, si ottiene valutando il modello di bici noleggiata, il tipo di cliente che effettua il noleggio e il tipo di giorno in cui questo avviene.
Si sceglie, tuttavia, di conservarla al fine di velocizzare una delle operazioni più ricorrenti dell'intera base dati: l'inserimento di un nuovo noleggio.

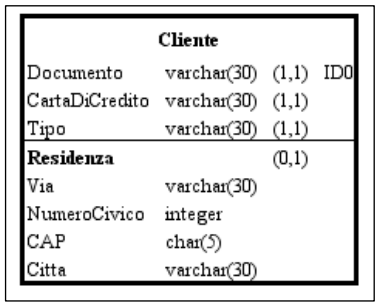
Le due ridondanze individuate sono effettivamente tali poiché si assume che, una volta inserita una tariffa all'interno del database, non sia possibile né cancellarla né modificarla.

4.3. Eliminazione delle generalizzazioni

4.3.1. Generalizzazione *Cliente*



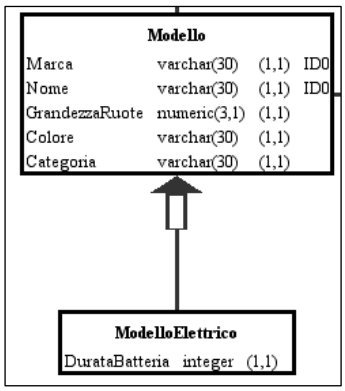
Prima



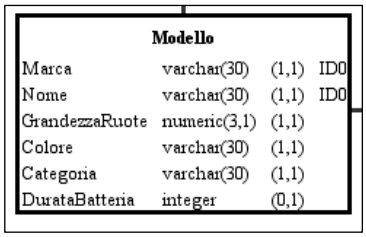
Dopo

La strategia applicata consiste nell'accorpamento delle entità figlie nell'entità padre: la scelta è determinata dal fatto che non sono previste operazioni specifiche che si riferiscono alle specializzazioni Studente ed Anziano e tutte le altre operazioni non fanno molta distinzione tra le occorrenze di Cliente, Studente ed Anziano. Si osservi che in Cliente è necessaria l'introduzione di un nuovo attributo: **Tipo**, fondamentale per distinguere le diverse tipologie di Cliente. Dato che la generalizzazione di partenza non è totale, Tipo potrà assumere i seguenti valori: {Studente, Anziano, Altro}. La scelta della strategia applicata è ulteriormente giustificata dal fatto che le specializzazioni Studente e Anziano non presentano attributi, pertanto l'eliminazione della generalizzazione non comporta l'effetto collaterale di possibili valori nulli.

4.3.2. Generalizzazione *Modello*



Prima

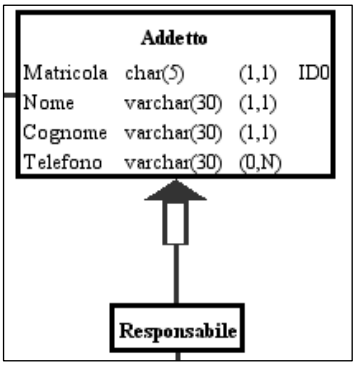


Dopo

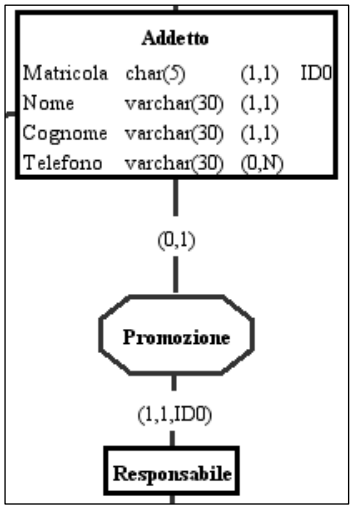
La strategia applicata consiste nell'accorpamento dell'unica entità figlia nell'entità genitore. La scelta è guidata da considerazioni simili a quelle scritte sopra. In particolare, le operazioni previste non fanno distinzione alcuna tra il genitore ed il figlio. Si noti che l'eliminazione della generalizzazione comporta l'introduzione dell'unico attributo di ModelloElettrico (ovvero **DurataBatteria**) in Modello, il che determina anche l'introduzione di vincoli aggiuntivi, secondo cui:

- Un modello la cui categoria è elettrica deve avere DurataBatteria
- Un modello la cui categoria è diversa da elettrica non deve avere DurataBatteria

4.3.3. Generalizzazione *Addetto*



Prima



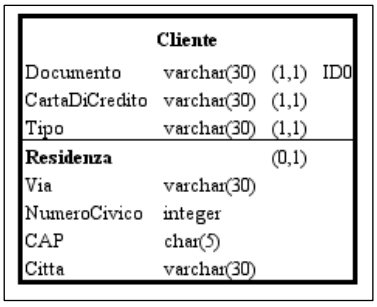
Dopo

La strategia applicata consiste nella sostituzione della generalizzazione con l'associazione **Promozione**. La scelta è determinata dall'esistenza delle associazioni Assegnazione e Direzione, che fanno differenza tra genitore e figlio. È conveniente, pertanto, mantenere entrambi, al fine di accedervi in modo separato. Si osservi che l'entità Responsabile è identificata esternamente da Addetto.

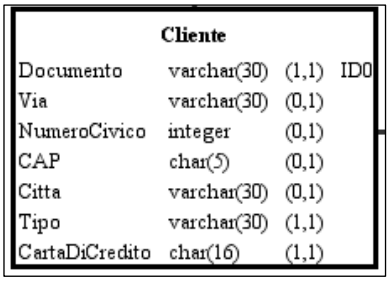
4.4. Partizionamento/Accorpamento Entità e Associazioni

Non abbiamo ritenuto necessario effettuare alcun tipo di partizionamento né di entità né di associazioni. L'opportuna decomposizione dei concetti costituenti la realtà di interesse, infatti, è già stata attuata in fase di progettazione concettuale. Non vi sono, pertanto, ulteriori decomposizioni capaci di garantire un miglioramento dell'efficienza delle operazioni. Si è identificata, tuttavia, l'esistenza di attributi composti e multivalore, che vanno opportunamente rimossi.

4.4.1. Attributo composto *Residenza (CLIENTE)*



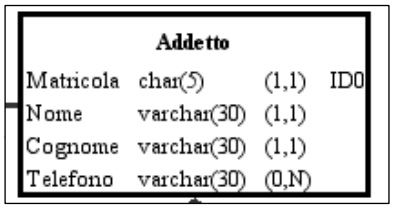
Prima



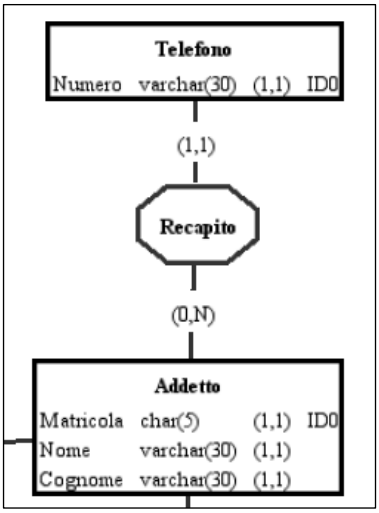
Dopo

La rimozione dell'attributo composto introduce un vincolo aggiuntivo:
- Gli attributi Via, NumeroCivico, CAP, Citta o sono tutti valorizzati o sono tutti non valorizzati.

4.4.2. Attributo multivalore Telefono (ADDETTO)



Prima



Dopo

L'attributo multivalore **Telefono** va rimosso dall'entità Addetto. Al suo posto occorre introdurre un'entità omonima da collegare ad Addetto mediante l'associazione Recapito. Si presti particolare attenzione alle cardinalità della relazione: dal lato di Addetto basta riproporre la cardinalità dell'attributo originario (0, N), dal lato di Telefono si ha (1, 1) in quanto ogni numero di telefono è associato esclusivamente ad uno e un solo Addetto.

4.5. Scelta degli identificatori principali

4.5.1. Bici

L'entità Bici è caratterizzata dalla presenza di due identificatori:
- **Codice**: trattasi di un numero intero (identificatore interno);
- (**NumTelaio, Modello)**: trattasi di un identificatore misto;

La scelta è ricaduta su Codice per chiare ragioni: esso è costituito da un unico attributo, il che garantisce che eventuali strutture ausiliarie create per accedere ai dati, ovvero gli indici, siano di dimensioni ridotte; un identificatore interno è sempre da preferire ad un identificatore misto in vista della fase di traduzione, che altrimenti potrebbe generare chiavi con molti attributi. Si osservi, infine, che è prevedibile che la stragrande maggioranza delle operazioni che intendono accedere alle occorrenze di Bici lo facciano attraverso l'identificatore Codice.

4.5.2. PuntoVendita

L'entità PuntoVendita è caratterizzata dalla presenza di due identificatori:
- **NumID**: trattasi di un numero intero (identificativo interno);
- **(Via, Civico, Citta, CAP)**: trattasi di un identificatore interno costituito da 4 attributi;

La scelta è ricaduta su NumID in quanto è costituito da un numero minore di attributi (solamente uno) ed è pertanto più semplice gestirlo. È prevedibile, inoltre, che la maggior parte delle operazioni che accedono ad occorrenze di PuntoVendita lo facciano sulla base del loro NumID.

4.5.3. Responsabile

L'entità Responsabile è caratterizzata dalla presenza di due identificatori esterni:
- **Addetto**: trattasi di un codice formato sempre da 5 cifre;
- **PuntoVendita**: trattasi di un numero intero;

La scelta è ricaduta su Addetto, in quanto si ritiene più logico identificare un responsabile mediante la sua matricola di addetto piuttosto che mediante il codice del punto vendita di cui è dirigente.

4.5.4. Acquisto

L'entità Acquisto è caratterizzata dalla presenza di tre identificatori:
- **(Transazione, Data)**: trattasi di un identificatore interno costituito da 2 attributi;
- **CardPrepagata**: trattasi di un identificatore esterno;
- **Cliente**: trattasi di un identificatore esterno;

La scelta è ricaduta su (Transazione, Data) in quanto è preferibile utilizzare l'identificatore interno piuttosto che uno di quelli esterni. Inoltre, risulta più naturale identificare un acquisto mediante tali attributi piuttosto che rifacendosi alla card prepagata acquistata.

4.5.5. Pagamento

L'entità Pagamento è caratterizzata dalla presenza di due identificatori:
- **(Transazione, Data)**: trattasi di un identificatore interno costituito da 2 attributi;
- **Noleggio**: trattasi di un identificatore esterno;

La scelta è ricaduta su (Transazione, Data) in quanto è preferibile utilizzare l'identificatore interno piuttosto che quello esterno. Inoltre, risulta più naturale identificare un pagamento mediante tali attributi piuttosto che rifacendosi al noleggio pagato.

4.6. Schema ristrutturato finale

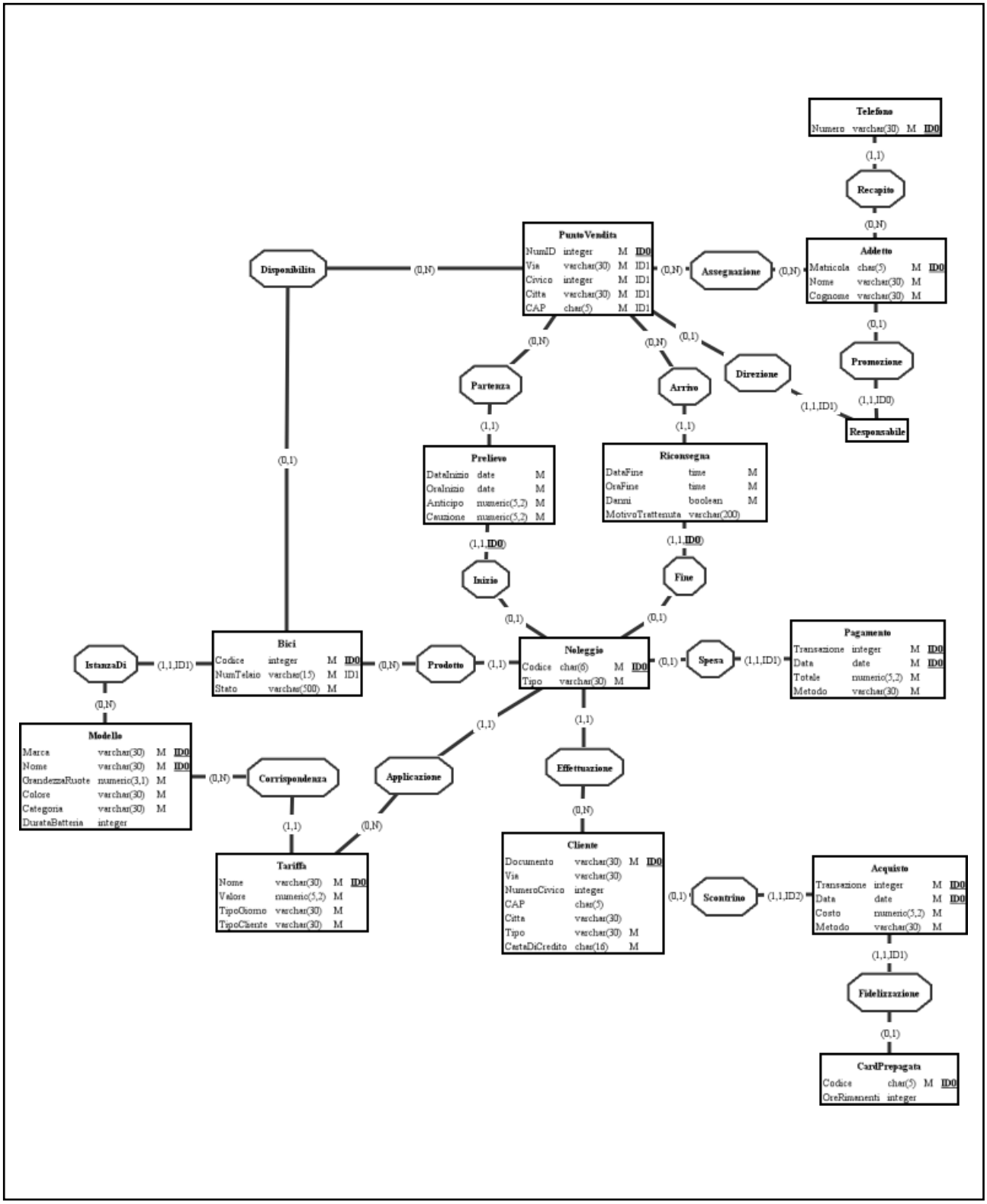


Figura 9. Schema ER Ristrutturato

4.7. Schema logico

BICI (Codice, NumTelaio, Stato, MarcaModello, NomeModello, PuntoVendita*)
[NumTelaio, MarcaModello, NomeModello] = AK (Alternative Key)
MarcaModello è in integrità referenziale con Marca in Modello
NomeModello è in integrità referenziale con Nome in Modello
PuntoVendita è in integrità referenziale con NumID in PuntoVendita

MODELLO (Marca, Nome, GrandezzaRuote, Colore, Categoria, DurataBatteria*)

TARIFFA (Nome, Valore, TipoGiorno, TipoCliente, MarcaModello, NomeModello)
MarcaModello è in integrità referenziale con Marca in Modello
NomeModello è in integrità referenziale con Nome in Modello

PUNTOVENDITA (NumID, Via, Civico, Citta, CAP)
[Via, Civico, Citta, CAP] = AK

ADDETTO (Matricola, Nome, Cognome)

TELEFONO (Numero, Addetto)
Addetto è in integrità referenziale con Matricola in Addetto

ASSEGNAZIONE (PuntoVendita, Addetto)
PuntoVendita è in integrità referenziale con NumID in PuntoVendita
Addetto è in integrità referenziale con Matricola in Addetto

RESPONSABILE (Addetto, PuntoVendita)
[PuntoVendita] = AK
Addetto è in integrità referenziale con Matricola in Addetto
PuntoVendita è in integrità referenziale con NumID in PuntoVendita

CLIENTE (Documento, Via, NumeroCivico, CAP, Citta, Tipo, CartaDiCredito)

ACQUISTO (Transazione, Data, Costo, Metodo, Cliente, CardPrepagata)
[CardPrepagata] = AK
[Cliente]=AK
Cliente è in integrità referenziale con Documento in Cliente
CardPrepagata è in integrità referenziale con Codice in CardPrepagata

CARDPREPAGATA (Codice, OreRimanenti*)

NOLEGGIO (Codice, Tipo, Bici, Tariffa, Cliente)
Bici è in integrità referenziale con Codice in Bici
Tariffa è in integrità referenziale con Nome in Tariffa
Cliente è in integrità referenziale con Documento in Cliente

PRELIEVO (Noleggio, DataInizio, OraInizio, Anticipo, Cauzione, PuntoVendita)
Noleggio è in integrità referenziale con Codice in Noleggio
PuntoVendita è in integrità referenziale con NumID in PuntoVendita

RICONSEGNA (Noleggio, DataFine, OraFine, Danni, MotivoTrattenuta*, PuntoVendita)
Noleggio è in integrità referenziale con Codice in Noleggio
PuntoVendita è in integrità referenziale con NumID in PuntoVendita

PAGAMENTO (Transazione, Data, Totale, Metodo, Noleggio)
[Noleggio] = AK
Noleggio è in integrità referenziale con Codice in Noleggio

4.8. Documentazione dello schema logico

4.8.1. Vincoli introdotti da ristrutturazione e traduzione

- Un modello la cui categoria è elettrica deve avere DurataBatteria;
- Un modello la cui categoria è diversa da elettrica *non* deve avere DurataBatteria;
- Un telefono deve essere assegnato ad un addetto (come si evince anche dallo schema);
- Via, NumeroCivico, CAP, Citta in *Cliente* devono essere tutti contemporaneamente valorizzati oppure tutti contemporaneamente non valorizzati;
- In *Cliente* l'attributo Tipo deve essere in {Studente, Anziano, Altro}

4.8.2. Rappresentazione Grafica

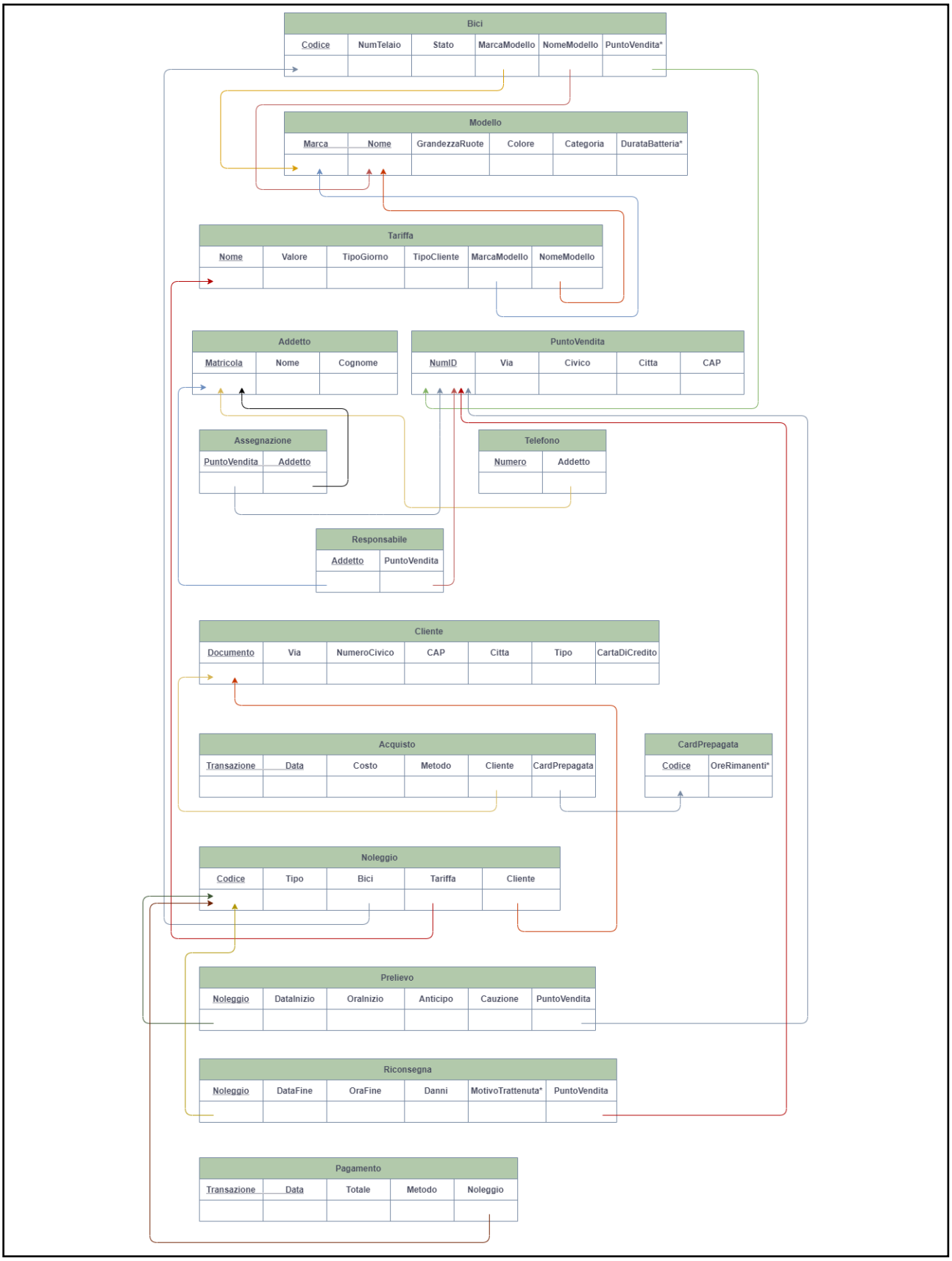


Figura 10. Grafico dei vincoli di integrità referenziale

5. Normalizzazione

Workpackage	***Task***	***Responsabile***
WP3	Normalizzazione	Di Mauro Enrico Maria

BICI (Codice, NumTelaio, Stato, MarcaModello, NomeModello, PuntoVendita*)
Sicuramente in 2NF perché la chiave è composta da un solo attributo.
Si individuano le seguenti dipendenze funzionali critiche:
NomeModello, MarcaModello, NumTelaio → Codice
NomeModello, MarcaModello, NumTelaio → Stato
NomeModello, MarcaModello, NumTelaio → PuntoVendita
Scegliamo comunque di non decomporre la tabella, dal momento che (NomeModello, MarcaModello, NumTelaio) costituisce una chiave alternativa, pertanto non vi saranno sicuramente né ridondanze né anomalie di inserimento/aggiornamento/cancellazione problematiche.

MODELLO (Marca, Nome, GrandezzaRuote, Colore, Categoria, DurataBatteria*)
Sicuramente in 2NF perché ciascun attributo non primo dipende in maniera piena dalla chiave.
Inoltre, possiamo affermare che la tabella è in BCNF poiché non esistono dipendenze funzionali del tipo Y → Z tali che Y sia diversa da una superchiave.

TARIFFA (Nome, Valore, TipoGiorno, TipoCliente, MarcaModello, NomeModello)
Sicuramente in 2NF perché la chiave è composta da un solo attributo.
Si individua la seguente DF:
Valore, TipoGiorno, TipoCliente, MarcaModello, NomeModello → Nome
Pertanto, la definizione della 3NF è rispettata mentre non lo è quella della BCNF.

PUNTOVENDITA (NumID, Via, NumeroCivico, Citta, CAP)
Sicuramente in 2NF perché la chiave è composta da un solo attributo.
Si individuano le seguenti DF:
Citta, Via, NumeroCivico → CAP
Pertanto, la tabella non rispetta la definizione di 3NF, tuttavia, scegliamo di non decomporla poiché, dato il basso numero di occorrenze previste a regime, è altamente improbabile che si verifichino ridondanze e/o anomalie problematiche.
Per ulteriori motivazioni si guardi l'analisi della tabella Cliente.

ADDETTO (Matricola, Nome, Cognome)
È immediato osservare che la tabella si trova in BCNF, dato che tutte le dipendenze funzionali non banali ne rispettano la definizione.

TELEFONO (Numero, Addetto)
È immediato osservare che la tabella si trova in BCNF, dato che tutte le dipendenze funzionali non banali ne rispettano la definizione.

ASSEGNAZIONE (PuntoVendita, Addetto)
La tabella è sicuramente in BCNF, dato che gli unici attributi costituiscono la chiave primaria.

RESPONSABILE (Addetto, PuntoVendita)
La tabella è sicuramente in BCNF, dato che l'unica DF è tra un attributo primo e uno non primo: Addetto → PuntoVendita.

CLIENTE (Documento, Via, NumeroCivico, CAP, Citta, Tipo, CartaDiCredito)
Sicuramente in 2NF perché la chiave è composta da un solo attributo.
Si individua la seguente DF:
Citta, Via, NumeroCivico → CAP
Pertanto, la tabella non rispetta la definizione di 3NF. Si sceglie, tuttavia, di non decomporla poiché ciò comporterebbe l'introduzione di una tabella i cui attributi sono esattamente i quattro presenti nella DF. Tale decisione, infatti, determinerebbe lo spreco di ulteriore memoria per la memorizzazione di informazioni già contenute nella tabella Cliente. Ciò evidenzia, infine, che non è possibile ricavare il CAP in modo semplice.

ACQUISTO (Transazione, Data, Costo, Metodo, Cliente, CardPrepagata)
Sicuramente in 2NF perché ciascun attributo non primo dipende in maniera piena dalla chiave.
Dato che non si evidenziano ulteriori dipendenze oltre a quelle che mettono in relazione ciascun attributo non primo all'intera chiave primaria, possiamo concludere che la tabella è in BCNF.

CARDPREPAGATA (Codice, OreRimanenti*)
La tabella è sicuramente in BCNF, dato che l'unica DF è tra un attributo primo e uno non primo: Codice → OreRimanenti.

NOLEGGIO (Codice, Tipo, Bici, Tariffa, Cliente)
È immediato osservare che la tabella si trova in BCNF, dato che tutte le dipendenze funzionali non banali ne rispettano la definizione.

PRELIEVO (Noleggio, DataInizio, OraInizio, Anticipo, Cauzione, PuntoVendita)
È immediato osservare che la tabella si trova in BCNF, dato che tutte le dipendenze funzionali non banali ne rispettano la definizione.

RICONSEGNA (Noleggio, DataFine, OraFine, Danni, MotivoTrattenuta*, PuntoVendita)
È immediato osservare che la tabella si trova in BCNF, dato che tutte le dipendenze funzionali non banali ne rispettano la definizione.

PAGAMENTO (Transazione, Data, Totale, Metodo, Noleggio)
Sicuramente 2NF perché ciascun attributo non primo dipende in maniera piena dalla chiave
Noleggio →Metodo
Noleggio → Totale
Noleggio → Transazione
Noleggio → Data
Scegliamo comunque di non decomporre la tabella, dal momento che Noleggio costituisce una chiave alternativa, pertanto non vi saranno sicuramente né ridondanze né anomalie di inserimento/aggiornamento/cancellazione problematiche.

6. Script Creazione e Popolamento Database

Workpackage	*Task*	*Responsabile*
WP2	SQL: Script creazione e popolamento	Grimaldi Salvatore

--SCRIPT DI CREAZIONE

```
drop table if exists modello cascade;
drop table if exists puntovendita cascade;
drop table if exists addetto cascade;
drop table if exists assegnazione cascade;
drop table if exists telefono cascade;
drop table if exists responsabile cascade;
drop table if exists cliente cascade;
drop table if exists cardprepagata cascade;
drop table if exists acquisto cascade;
drop table if exists bici cascade;
drop table if exists tariffa cascade;
drop table if exists noleggio cascade;
drop table if exists prelievo cascade;
drop table if exists riconsegna cascade;
drop table if exists pagamento cascade;

create table modello(
   marca varchar(30) not null,
   nome varchar(30) not null,
   grandezzaruote numeric(3,1) not null check(grandezzaruote > 0),
         colore varchar(30) not null,
   categoria varchar(30) not null,
   duratabatteria integer null,
   primary key(marca, nome),
   check (categoria='DaCitta' OR categoria = 'MountainBike' OR categoria = 'Bambino' OR categoria =
'APedalataAssistita' OR categoria = 'Elettrica'),
   check ((categoria='Elettrica' AND duratabatteria is not null AND duratabatteria>0) OR(categoria != 'Elettrica' AND
duratabatteria is null))
);

create table puntovendita(
   numid integer primary key,
         via varchar(30) not null,
   civico integer not null,
   citta varchar(30) not null,
   cap char(5) not null,
   unique(via, civico, citta, cap),
   check(numid > 0),
   check(civico > 0),
   check(char_length(cap)=5)
);

create table addetto(
   matricola char(5) primary key check((char_length(matricola)=5)),
   nome varchar(30) not null,
   cognome varchar(30) not null
);

create table assegnazione(
   puntovendita integer not null references puntovendita(numid)
```

```
    on delete cascade on update cascade,
        addetto char(5) not null references addetto(matricola)
    on delete cascade on update cascade,
  primary key(puntovendita, addetto)
);

create table telefono(
  numero varchar(30) primary key,
  addetto char(5) not null references addetto(matricola)
    on delete cascade on update cascade
);

create table responsabile(
  addetto char(5) not null references addetto(matricola)
      on delete cascade on update cascade,
    puntovendita integer not null references puntovendita(numid)
      on delete cascade on update cascade,
    primary key(addetto),
  unique(puntovendita)
);

create table cliente(
  documento varchar(30) primary key,
  via varchar(30) null,
  numerocivico integer null check(numerocivico > 0),
  cap char(5) null check((char_length(cap)=5)),
  citta varchar(30) null,
  tipo varchar(30) not null check(tipo='Studente' OR tipo='Anziano' OR tipo='Altro'),
  cartadicredito char(16) not null check((char_length(cartadicredito)=16)),
  check((via is not null AND numerocivico is not null AND cap is not null AND citta is not null)OR
          (via is null AND numerocivico is null AND cap is null AND citta is null))
);

create table cardprepagata(
  codice char(5) primary key check((char_length(codice)=5)),
  orerimanenti integer null check((orerimanenti is null) OR (orerimanenti is not null AND orerimanenti >= 0))
);

create table acquisto(
  transazione integer not null check(transazione > 0),
  data date not null,
      primary key (transazione, data),
  costo numeric(5,2) not null check(costo > 0),
  metodo varchar(30) not null,
  check (metodo = 'Contanti' OR metodo = 'CartaDiCredito'),
      cliente varchar(30) unique not null references cliente(documento)
    on delete restrict on update cascade,
  cardprepagata char(5) not null references cardprepagata(codice)
    on delete cascade on update cascade,
      unique(cardprepagata)
);

create table bici(
  codice integer primary key check(codice > 0),
  numtelaio varchar(15) not null,
  stato varchar(500) not null,
  marcamodello varchar(30) not null,
```

```
    nomemodello varchar(30) not null ,
    puntovendita integer null references puntovendita(numid)
      on update cascade on delete set null,
            foreign key(marcamodello, nomemodello) references modello(marca, nome)
              on update cascade on delete restrict,
            unique(numtelaio, marcamodello, nomemodello)
);

create table tariffa(
    nome varchar(30) primary key,
    valore numeric(5,2) not null check(valore > 0),
    tipogiorno varchar(30) not null check (tipogiorno='Feriale' OR tipogiorno='Festivo'),
    tipocliente varchar(30) not null check(tipocliente='Studente' OR tipocliente='Anziano' OR tipocliente='Altro'),
    marcamodello varchar(30) not null,
    nomemodello varchar(30) not null,
    foreign key(marcamodello, nomemodello) references modello(marca, nome)
              on update cascade on delete cascade
);

create table noleggio(
    codice char(6) primary key check((char_length(codice)=6)),
    tipo varchar(30) not null check(tipo='AdOre' OR tipo='Giornaliero'),
    bici integer not null references bici(codice)
      on delete restrict on update cascade,
    tariffa varchar(30) not null references tariffa(nome)
              on delete restrict on update cascade,
            cliente varchar(30) not null references cliente(documento)
              on delete restrict on update cascade
);

create table prelievo(
    noleggio char(6) primary key references noleggio(codice)
               on delete cascade on update cascade,
    datainizio date not null,
    orainizio time not null,
    anticipo numeric(5,2) not null check(anticipo>=0),
    cauzione numeric(5,2) not null check(cauzione>0),
            puntovendita integer not null references puntovendita(numid)
               on delete restrict on update cascade
);

create table riconsegna(
    noleggio char(6) primary key references noleggio(codice)
               on delete cascade on update cascade,
            datafine date not null,
    orafine time not null,
            danni boolean not null,
    motivotrattenuta varchar(200) null,
            puntovendita integer not null references puntovendita(numid)
               on delete restrict on update cascade,
            check((danni is true AND motivotrattenuta is not null)OR(danni is false AND motivotrattenuta is null))
);


create table pagamento(
    transazione integer not null check(transazione > 0),
    data date not null,
```

```
        primary key (transazione, data),
        totale numeric(5,2) not null check(totale >= 0),
   metodo varchar(30) not null check(metodo='Contanti' OR metodo='CartaDiCredito' OR metodo='CardPrepagata'),
   noleggio char(6) not null unique references noleggio(codice)
         on delete cascade on update cascade
);


--SCRIPT DI POPOLAMENTO

insert into modello(marca, nome, grandezzaruote, colore, categoria, duratabatteria)
values('Trek', 'District 4', 28, 'grigio', 'DaCitta', null),
    ('Trek', 'Session 8', 29, 'nero', 'MountainBike', null),
    ('Focus', 'Paralane', 29, 'argento', 'DaCitta', null),
        ('Focus', 'Jam', 29, 'grigio', 'MountainBike', null),
        ('Specialized', 'MiniP3', 15, 'blu', 'Bambino', null),
        ('Specialized', 'Kenevo', 29, 'bianco', 'APedalataAssistita', null),
        ('KTM', 'CityFun Mini', 16.5, 'bianco', 'Bambino', null),
        ('KTM', 'Macina', 29, 'arancione', 'APedalataAssistita', null),
        ('KTM', 'Race 271', 29, 'marrone', 'Elettrica', 700),
        ('Canyon', 'Grail Plus', 29, 'bianco', 'Elettrica', 650);

insert into puntovendita(numid, via, civico, citta, cap)
values(1, 'Parigi', 55, 'Bologna', '40121'),
    (2, 'Roma', 121, 'Imola', '40026'),
        (3, 'Tosarelli', 64, 'Bologna', '40126'),
        (4, 'Delle Feste', 32, 'Marzabotto', '77777'),
        (5, 'San Rocco', 23, 'Sasso Marconi', '40033');

insert into addetto(matricola, nome, cognome)
values('00001', 'Salvatore', 'Grimaldi'),
    ('00002', 'Andrea', 'De Gruttola'),
        ('00003', 'Enrico Maria', 'Di Mauro'),
        ('00004', 'Allegra', 'Cuzzocrea'),
        ('00005', 'Guglielmo', 'Barone'),
        ('00006', 'Francesco', 'La Manna'),
        ('00007', 'Luigi', 'Grasso'),
        ('00008', 'Francesco', 'Silano'),
        ('00009', 'Giovanni', 'Cassiodoro'),
        ('00010', 'Manuele', 'Memoli'),
        ('00011', 'Jacopo', 'Memoli'),
        ('00012', 'Raffaella', 'Meninno'),
        ('00013', 'Cristian', 'Polidoro'),
        ('00014', 'Giovanna', 'Saggese'),
        ('00015', 'Marzio', 'Delli Priscoli');

insert into assegnazione(puntovendita, addetto)
values(1, '00001'),
    (2, '00002'),
        (3, '00003'),
        (4, '00004'),
        (5, '00005'),
        (1, '00006'),
        (2, '00006'),
        (3, '00007'),
        (4, '00007'),
        (5, '00008'),
```

```
            (1, '00008'),
            (2, '00009'),
            (3, '00009'),
            (4, '00010'),
            (5, '00010'),
            (1, '00011'),
            (2, '00011'),
            (3, '00012'),
            (4, '00012'),
            (5, '00013'),
            (1, '00013'),
            (2, '00014'),
            (3, '00014'),
            (4, '00015'),
            (5, '00015');

insert into telefono(numero, addetto)
values('0512193111', '00001'),
    ('0512193134', '00001'),
            ('0512193137', '00002'),
            ('0512193140', '00002'),
            ('0512193143', '00003'),
            ('0512193146', '00003'),
            ('0512193147', '00004'),
            ('0512193149', '00004'),
            ('0512193152', '00005'),
            ('0512193155', '00005'),
            ('0512193158', '00006'),
            ('0512193165', '00006'),
            ('0512193167', '00007'),
            ('0512193168', '00007'),
            ('0512193170', '00008'),
            ('0512193171', '00008'),
            ('0512193173', '00009'),
            ('0512193177', '00009'),
            ('0512193179', '00010'),
            ('0512193180', '00011'),
            ('0512193188', '00012'),
            ('0512193187', '00013'),
            ('0512193190', '00014'),
            ('0512191109', '00015');

insert into responsabile(addetto, puntovendita)
values('00001', 1),
    ('00002', 2),
            ('00003', 3),
            ('00004', 4),
            ('00005', 5);

insert into cardprepagata(codice, orerimanenti)
values('00001', null),
            ('00002', null),
            ('00003', null),
            ('00004', null),
            ('00005', null),
            ('00006', null),
            ('00007', null),
```

```
        ('00008', null),
        ('00009', null),
        ('00010', null),
        ('00011', null),
        ('00012', null),
        ('00013', null),
        ('00014', null),
        ('00015', null);

insert into cliente(documento, via, numerocivico, cap, citta, tipo, cartadicredito)
values('CA36495GP', 'Montepruno', 53, '20021', 'Milano', 'Studente', '0897321452345235'),
    ('TR8765FR', 'San Gemelli', 32, '40128', 'Bologna', 'Anziano', '0891221442345315'),
        ('GF1234YH', 'Europa', 17, '40128', 'Bologna', 'Altro', '1111321452345235'),
        ('DF6848GR', 'Dei due principati', 341, '40139','Bologna', 'Altro', '2222421452345235'),
    ('GF6758FS', 'Santissima Teresa', 53, '40139', 'Bologna', 'Altro', '0897321388845235'),
        ('RF543112', 'Delle tre sorelle', 45, '00161', 'Roma', 'Studente', '0897312228845235'),
        ('RF7895RD', 'Rimembranza', 65, '40139', 'Bologna', 'Altro', '0897321388841999'),
        ('BB7677BB', 'Della Buona Sorte', 1, '40141', 'Bologna', 'Altro', '0897326666665235'),
        ('PP6532DD', 'San Pio', 11, '40141', 'Bologna', 'Anziano', '0897327666665235'),
        ('YU3121LL', 'Dei Malavoglia', 7, '40132', 'Bologna', 'Altro', '0897326666665299'),
        ('KJ5312KK', 'Nazionale', 543, '40132', 'Bologna', 'Studente', '0812876666665235'),
        ('RF1234HH', 'Delle Puglie', 98, '40132', 'Bologna', 'Altro', '0897226668376235'),
        ('TG5999CX', 'San Rocco', 43, '00156', 'Roma', 'Anziano', '0897326666665111'),
        ('MN5314FF', 'Delle Battaglie', 9, '40139', 'Bologna', 'Altro', '1234326666665235'),
        ('GT4567AA', 'San Leucio', 4, '40139', 'Bologna', 'Altro', '0897321196665235');

insert into bici(codice, numtelaio, stato, marcamodello, nomemodello, puntovendita)
values(1, 'F61465', 'ok', 'Canyon', 'Grail Plus', 2),
    (19, 'F61466', 'luce posteriore rotta', 'Canyon', 'Grail Plus', 3),
        (2, 'G61474', 'ok', 'KTM', 'Race 271', 1),
        (3, 'G61475', 'ok', 'KTM', 'Macina', 2),
        (18, 'G614WW', 'ok', 'KTM', 'Macina', 1),
        (4, 'G61477', 'ok', 'KTM', 'CityFun Mini', 3),
        (17, 'G61111', 'ok', 'KTM', 'CityFun Mini', 2),
        (5, 'D61477', 'cavalletto mancante', 'Specialized', 'Kenevo', 2),
        (16, 'D61499', 'ok', 'Specialized', 'Kenevo', 3),
        (6, 'D61489', 'ok', 'Specialized', 'MiniP3', 2),
        (20, 'D61490', 'ok', 'Specialized', 'MiniP3', 1),
        (7, 'E61408', 'ok', 'Focus', 'Jam', 5),
        (8, 'E63301', 'luce difettosa', 'Focus', 'Paralane', 5),
        (9, 'J63355', 'ok', 'Trek', 'Session 8', 4),
        (15, 'J63366', 'vernice mancante in qualche punto', 'Trek', 'Session 8', 5),
        (10, 'J63367', 'ok', 'Trek', 'District 4', 5),
        (11, 'J39999', 'ok', 'Trek', 'District 4', 4),
        (12, 'E68888', 'ok', 'Focus', 'Jam', 3),
        (13, 'E67777', 'cavalletto mancante', 'Focus', 'Paralane', 4),
        (14, 'E67788', 'cavalletto rotto', 'Focus', 'Paralane', 1),
        (21, 'RD3456', 'ok', 'Trek', 'District 4', 1),
        (22, 'FR567W', 'ok', 'Trek', 'Session 8', 1),
        (23, '456578', 'sverniciatura', 'Focus', 'Paralane', 1),
        (24, 'FRGG76', 'ok', 'Focus', 'Jam', 1),
        (25, 'GTY789', 'ok', 'Specialized', 'MiniP3', 1),
        (26, '897654', 'ok', 'Specialized', 'Kenevo', 1),
        (27, '0097GHT', 'ok', 'KTM', 'CityFun Mini', 1),
        (28, '897HG', 'ok', 'KTM', 'Macina', 1),
        (29, 'KIHGBK', 'ok', 'KTM', 'Race 271', 1),
        (30, 'IIIkHJ','ok','Canyon', 'Grail Plus', 1);
```

```
insert into tariffa(nome, valore, tipogiorno, tipocliente, marcamodello, nomemodello)
values('District4_feriale_studente', 1, 'Feriale', 'Studente', 'Trek', 'District 4'),
    ('District4_feriale_anziano', 1, 'Feriale', 'Anziano', 'Trek', 'District 4'),
        ('District4_feriale_altro', 1.20, 'Feriale', 'Altro', 'Trek', 'District 4'),
        ('District4_festivo_studente', 1.20, 'Festivo', 'Studente', 'Trek', 'District 4'),
        ('District4_festivo_anziano', 1.20, 'Festivo', 'Anziano', 'Trek', 'District 4'),
        ('District4_festivo_altro', 1.40, 'Festivo', 'Altro', 'Trek', 'District 4'),
        ('Session8_feriale_studente', 1.10, 'Feriale', 'Studente', 'Trek', 'Session 8'),
    ('Session8_feriale_anziano', 1.15, 'Feriale', 'Anziano', 'Trek', 'Session 8'),
        ('Session8_feriale_altro', 1.25, 'Feriale', 'Altro', 'Trek', 'Session 8'),
        ('Session8_festivo_studente', 1.30, 'Festivo', 'Studente', 'Trek', 'Session 8'),
        ('Session8_festivo_anziano', 1.30, 'Festivo', 'Anziano', 'Trek', 'Session 8'),
        ('Session8_festivo_altro', 1.40, 'Festivo', 'Altro', 'Trek', 'Session 8'),
        ('Paralane_feriale_studente', 0.9, 'Feriale', 'Studente', 'Focus', 'Paralane'),
    ('Paralane_feriale_anziano', 0.9, 'Feriale', 'Anziano', 'Focus', 'Paralane'),
        ('Paralane_feriale_altro', 1, 'Feriale', 'Altro', 'Focus', 'Paralane'),
        ('Paralane_festivo_studente', 1.10, 'Festivo', 'Studente', 'Focus', 'Paralane'),
        ('Paralane_festivo_anziano', 1.10, 'Festivo', 'Anziano', 'Focus', 'Paralane'),
        ('Paralane_festivo_altro', 1.30, 'Festivo', 'Altro', 'Focus', 'Paralane'),
        ('Jam_feriale_studente', 2, 'Feriale', 'Studente', 'Focus', 'Jam'),
    ('Jam_feriale_anziano', 2.10, 'Feriale', 'Anziano', 'Focus', 'Jam'),
        ('Jam_feriale_altro', 2.30, 'Feriale', 'Altro', 'Focus', 'Jam'),
        ('Jam_festivo_studente', 2.50, 'Festivo', 'Studente', 'Focus', 'Jam'),
        ('Jam_festivo_anziano', 2.45, 'Festivo', 'Anziano', 'Focus', 'Jam'),
        ('Jam_festivo_altro', 2.60, 'Festivo', 'Altro', 'Focus', 'Jam'),
        ('MiniP3_feriale_studente', 0.90, 'Feriale', 'Studente', 'Specialized', 'MiniP3'),
    ('MiniP3_feriale_anziano', 0.90, 'Feriale', 'Anziano', 'Specialized', 'MiniP3'),
        ('MiniP3_feriale_altro', 1, 'Feriale', 'Altro', 'Specialized', 'MiniP3'),
        ('MiniP3_festivo_studente', 0.9, 'Festivo', 'Studente', 'Specialized', 'MiniP3'),
        ('MiniP3_festivo_anziano', 0.9, 'Festivo', 'Anziano', 'Specialized', 'MiniP3'),
        ('MiniP3_festivo_altro', 1.2, 'Festivo', 'Altro', 'Specialized', 'MiniP3'),
        ('Kenevo_feriale_studente', 1.55, 'Feriale', 'Studente', 'Specialized', 'Kenevo'),
    ('Kenevo_feriale_anziano', 1.6, 'Feriale', 'Anziano', 'Specialized', 'Kenevo'),
        ('Kenevo_feriale_altro', 1.8, 'Feriale', 'Altro', 'Specialized', 'Kenevo'),
        ('Kenevo_festivo_studente', 2, 'Festivo', 'Studente', 'Specialized', 'Kenevo'),
        ('Kenevo_festivo_anziano', 2, 'Festivo', 'Anziano', 'Specialized', 'Kenevo'),
        ('Kenevo_festivo_altro', 2, 'Festivo', 'Altro', 'Specialized', 'Kenevo'),
        ('CityFunMini_feriale_studente', 0.8, 'Feriale', 'Studente', 'KTM', 'CityFun Mini'),
    ('CityFunMini_feriale_anziano', 0.8, 'Feriale', 'Anziano', 'KTM', 'CityFun Mini'),
        ('CityFunMini_feriale_altro', 1.5, 'Feriale', 'Altro', 'KTM', 'CityFun Mini'),
        ('CityFunMini_festivo_studente', 1, 'Festivo', 'Studente', 'KTM', 'CityFun Mini'),
        ('CityFunMini_festivo_anziano', 1, 'Festivo', 'Anziano', 'KTM', 'CityFun Mini'),
        ('CityFunMini_festivo_altro', 1.1, 'Festivo', 'Altro', 'KTM', 'CityFun Mini'),
        ('Macina_feriale_studente', 1.2, 'Feriale', 'Studente', 'KTM', 'Macina'),
    ('Macina_feriale_anziano', 1.2, 'Feriale', 'Anziano', 'KTM', 'Macina'),
        ('Macina_feriale_altro', 1.4, 'Feriale', 'Altro', 'KTM', 'Macina'),
        ('Macina_festivo_studente', 1.45, 'Festivo', 'Studente', 'KTM', 'Macina'),
        ('Macina_festivo_anziano', 1.45, 'Festivo', 'Anziano', 'KTM', 'Macina'),
        ('Macina_festivo_altro', 1.55, 'Festivo', 'Altro', 'KTM', 'Macina'),
        ('Race271_feriale_studente', 2.5, 'Feriale', 'Studente', 'KTM', 'Race 271'),
    ('Race271_feriale_anziano', 2.5, 'Feriale', 'Anziano', 'KTM', 'Race 271'),
        ('Race271_feriale_altro', 2.8, 'Feriale', 'Altro', 'KTM', 'Race 271'),
        ('Race271_festivo_studente', 2.7, 'Festivo', 'Studente', 'KTM', 'Race 271'),
        ('Race271_festivo_anziano', 2.7, 'Festivo', 'Anziano', 'KTM', 'Race 271'),
        ('Race271_festivo_altro', 2.8, 'Festivo', 'Altro', 'KTM', 'Race 271'),
        ('GrailPlus_feriale_studente', 2.10, 'Feriale', 'Studente', 'Canyon', 'Grail Plus'),
```

```
    ('GrailPlus_feriale_anziano', 2.10, 'Feriale', 'Anziano', 'Canyon', 'Grail Plus'),
            ('GrailPlus_feriale_altro', 2.30, 'Feriale', 'Altro', 'Canyon', 'Grail Plus'),
            ('GrailPlus_festivo_studente', 2.60, 'Festivo', 'Studente', 'Canyon', 'Grail Plus'),
            ('GrailPlus_festivo_anziano', 2.60, 'Festivo', 'Anziano', 'Canyon', 'Grail Plus'),
            ('GrailPlus_festivo_altro', 2.65, 'Festivo', 'Altro', 'Canyon', 'Grail Plus');

insert into acquisto(transazione, data, costo, metodo, cliente, cardprepagata)
values (1, '2021-05-02', 13.20, 'Contanti', 'TR8765FR', '00001'),
            (2, '2021-05-02', 13.20, 'CartaDiCredito','GF1234YH', '00002'),
            (3, '2021-05-02', 26.40, 'Contanti','DF6848GR', '00003'),
            (1, '2021-05-03', 11.00, 'CartaDiCredito','GF6758FS', '00004'),
            (2, '2021-05-03', 26.40, 'Contanti','RF7895RD', '00005');

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000001', 'AdOre', 19, 'GrailPlus_festivo_altro', 'GF1234YH');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000001', '2021-05-02', '08:00:00', 0, 20, 3);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000001', '2021-05-02', '10:00:00', false, null, 3);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000002', 'AdOre', 2, 'Race271_feriale_altro', 'GF6758FS');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000002', '2021-05-03', '10:00:00', 0, 20, 1);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000002', '2021-05-03', '14:00:00', false, null, 1);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000003', 'AdOre', 18, 'Macina_feriale_altro', 'RF7895RD');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000003', '2021-05-03', '17:00:00', 0, 20, 1);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000003', '2021-05-03', '21:00:00', false, null, 1);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000004', 'AdOre', 4, 'CityFunMini_feriale_altro', 'RF7895RD');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000004', '2021-05-04', '09:30:00', 0, 20, 3);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000004', '2021-05-04', '13:30:00', false, null, 3);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000005', 'Giornaliero', 13, 'Paralane_feriale_studente', 'CA36495GP'),
            ('000006', 'AdOre', 17, 'CityFunMini_feriale_altro', 'RF7895RD');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000005', '2021-05-05', '11:00:00', 21.60, 20, 4),
            ('000006', '2021-05-05', '14:00:00', 0, 20, 2);
```

```
insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000006', '2021-05-05', '18:00:00', false, null, 2);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000007', 'AdOre', 15, 'Session8_feriale_altro', 'DF6848GR');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000007', '2021-05-06', '09:00:00', 6.25, 20, 2);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000005', '2021-05-06', '11:00:00', false, null, 4);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000008', 'AdOre', 8, 'Paralane_feriale_anziano', 'TR8765FR');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000008', '2021-05-06', '15:45:00', 0, 20, 5);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000007', '2021-05-06', '18:30:00', false, null, 2),
    ('000008', '2021-05-06', '20:45:00', false, null, 5);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000009', 'AdOre', 10, 'District4_feriale_studente', 'RF543112'),
        ('000010', 'Giornaliero', 20, 'MiniP3_feriale_altro', 'BB7677BB');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000009', '2021-05-07', '09:15:00', 0, 20, 5),
        ('000010', '2021-05-07', '10:00:00', 24, 20, 1);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000009', '2021-05-07', '17:15:00', false, null, 5);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000011', 'AdOre', 19, 'GrailPlus_festivo_altro', 'YU3121LL'),
        ('000012', 'AdOre', 11, 'District4_feriale_anziano', 'PP6532DD');


insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000011', '2021-05-08', '08:00:00', 0, 20, 3),
    ('000012', '2021-05-08', '10:00:00', 0, 20, 4);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000010', '2021-05-08', '10:00:00', true, 'luce difettosa', 1),
    ('000011', '2021-05-08', '13:00:00', false, null, 3);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000013', 'AdOre', 4, 'CityFunMini_feriale_altro', 'DF6848GR');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000013', '2021-05-08', '13:30:00', 0, 20, 3);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000013', '2021-05-08', '19:30:00', false, null, 3),
        ('000012', '2021-05-08', '20:00:00', false, null, 4);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
```

```
values ('000014', 'Giornaliero', 5, 'Kenevo_festivo_studente', 'KJ5312KK'),
        ('000015', 'AdOre', 10, 'District4_festivo_altro', 'RF1234HH'),
        ('000016', 'AdOre', 13, 'Paralane_festivo_anziano', 'TG5999CX'),
        ('000017', 'AdOre', 18, 'Macina_festivo_altro', 'MN5314FF');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000014', '2021-05-09', '07:00:00', 48, 20, 2),
        ('000015', '2021-05-09', '08:00:00', 0, 20, 5),
        ('000016', '2021-05-09', '09:15:00', 5.50, 20, 4),
        ('000017', '2021-05-09', '14:30:00', 0, 20, 1);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000016', '2021-05-09', '18:15:00', true, 'freno mancante', 4),
        ('000015', '2021-05-09', '19:00:00', false, null, 5),
        ('000017', '2021-05-09', '20:30:00', false, null, 1);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000014', '2021-05-10', '07:00:00', false, null, 2);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000018', 'AdOre', 1, 'Kenevo_feriale_studente', 'KJ5312KK'),
        ('000019', 'AdOre', 16, 'Kenevo_feriale_altro', 'GT4567AA'),
        ('000020', 'AdOre', 8, 'Paralane_feriale_anziano', 'TR8765FR');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000018', '2021-05-10', '08:00:00', 24, 20, 2),
        ('000019', '2021-05-10', '11:45:00', 18, 20, 3),
        ('000020', '2021-05-10', '13:00:00', 0, 20, 5);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000020', '2021-05-10', '19:00:00', false, null, 5),
        ('000019', '2021-05-10', '22:45:00', false, null, 3),
        ('000018', '2021-05-10', '23:30:00', false, null, 2);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000021', 'AdOre', 2, 'Race271_feriale_studente', 'RF543112'),
        ('000022', 'AdOre', 19, 'GrailPlus_feriale_studente', 'KJ5312KK'),
        ('000023', 'AdOre', 4, 'CityFunMini_feriale_anziano', 'TR8765FR'),
        ('000024', 'AdOre', 18, 'Macina_feriale_anziano', 'PP6532DD');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000021', '2021-05-11', '07:15:00', 0, 20, 1),
        ('000022', '2021-05-11', '07:30:00', 0, 20, 3),
        ('000023', '2021-05-11', '07:45:00', 0, 20, 3),
        ('000024', '2021-05-11', '08:00:00', 0, 20, 1);

insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
values ('000021', '2021-05-11', '09:15:00', false, null, 1),
        ('000022', '2021-05-11', '09:30:00', false, null, 3),
        ('000023', '2021-05-11', '09:45:00', false, null, 3),
        ('000024', '2021-05-11', '10:00:00', false, null, 1);

insert into noleggio(codice, tipo, bici, tariffa, cliente)
values ('000025', 'AdOre', 4, 'CityFunMini_feriale_anziano', 'TR8765FR');

insert into prelievo(noleggio, datainizio, orainizio, anticipo, cauzione, puntovendita)
values ('000025', '2021-05-12', '07:45:00', 0, 20, 3);
```

```
--insert into riconsegna(noleggio, datafine, orafine, danni, motivotrattenuta, puntovendita)
--values ('000025', '2021-05-12', '09:45:00', false, null, 3);

insert into pagamento(transazione, data, totale, metodo, noleggio)
values (1, '2021-05-02', 0, 'CardPrepagata', '000001'),
    (1, '2021-05-03', 0, 'CardPrepagata', '000002'),
            (2, '2021-05-03', 0, 'CardPrepagata', '000003'),
            (1, '2021-05-04', 0, 'CardPrepagata', '000004'),
            (1, '2021-05-05', 0, 'CardPrepagata', '000006'),
            (1, '2021-05-06', 21.60, 'Contanti', '000005'),
            (3, '2021-05-06', 12.50, 'Contanti', '000007'),
            (2, '2021-05-06', 4.50, 'Contanti', '000008'),
            (1, '2021-05-07', 8, 'Contanti', '000009'),
            (1, '2021-05-08', 24, 'CartaDiCredito', '000010'),
            (4, '2021-05-08', 13.25, 'CartaDiCredito', '000011'),
            (3, '2021-05-08', 10, 'Contanti', '000012'),
            (2, '2021-05-08', 9, 'Contanti', '000013'),
            (1, '2021-05-09', 15.40, 'Contanti', '000015'),
            (3, '2021-05-09', 9.90, 'Contanti', '000016'),
            (2, '2021-05-09', 9.30, 'CartaDiCredito', '000017'),
            (1, '2021-05-10', 48, 'Contanti', '000014'),
            (2, '2021-05-10', 5.40, 'Contanti', '000020'),
            (3, '2021-05-10', 19.80, 'Contanti', '000019'),
            (4, '2021-05-10', 24.80, 'Contanti', '000018'),
            (1, '2021-05-11', 5, 'Contanti', '000021'),
            (2, '2021-05-11', 4.20, 'CartaDiCredito', '000022'),
            (3, '2021-05-11', 1.60, 'CartaDiCredito', '000023'),
            (4, '2021-05-11', 2.40, 'Contanti', '000024');
            --(1, '2021-05-12', 1.60, 'CartaDiCredito', '000025');
```

7. Query SQL

Workpackage	*Task*	*Responsabile*
WP3	SQL: Query	Di Mauro Enrico Maria

7.1. Query con operatore di aggregazione e join: Studenti poveri

Stampa il documento dei clienti studenti che hanno almeno una volta pagato un noleggio meno di 10 e il numero di volte che lo hanno fatto.

```
select documento, count(*)
from noleggio, pagamento, cliente
where noleggio.codice=pagamento.noleggio and
    pagamento.totale<10 and
    noleggio.cliente=cliente.documento and
    cliente.tipo='Studente'
group by(documento);
```

7.2. Query nidificata complessa: Punti vendita completi

Stampa tutti i punti vendita in cui sono disponibili tutti i modelli di bici.

```
select *
from puntovendita PV
where not exists (select M.nome, M.marca
                  from modello M
                  except
                  select B.nomemodello, B.marcamodello
                  from bici B
                  where B.puntovendita=PV.numid);
```

7.3. Query insiemistica: Resoconto spese clienti

Stampa resoconto spese per ogni cliente, indicando il tipo di spesa.

```
select cliente, totale, transazione, data, 'noleggio' as tipo
from noleggio, pagamento
where noleggio.codice=pagamento.noleggio
union all
select documento as cliente, costo as totale, transazione, data, 'card' as tipo
from cliente, acquisto
where cliente.documento=acquisto.cliente
order by cliente;
```

7.4. Altre query

7.4.1. Clienti lusso

Stampa i clienti che hanno noleggiato il modello di bici più costoso con il relativo modello e tariffa.

```
select cliente.*, tariffa.nomemodello, tariffa.marcamodello, tariffa.valore
from cliente, noleggio, tariffa
where cliente.documento=noleggio.cliente and
      noleggio.tariffa=tariffa.nome and
      tariffa.valore=(select max(tariffa.valore)
                                  from tariffa, noleggio
                                  where tariffa.nome=noleggio.tariffa);
```

7.4.2. Noleggi in corso

Stampa tutti i noleggi che non sono stati riconsegnati.

- *Versione con operatore insiemistico*

```
select *
from noleggio
except
select noleggio.*
from noleggio, riconsegna
where noleggio.codice=riconsegna.noleggio;
```

- *Versione con query nidificata*

```
select *
from noleggio
where noleggio.codice not in (select riconsegna.noleggio
                                  from riconsegna);
```

- *Versione con query nidificata complessa*

```
select *
from noleggio
where not exists (select riconsegna.noleggio
                                  from riconsegna
                                  where noleggio.codice=riconsegna.noleggio);
```

8. Viste

Workpackage	*Task*	*Responsabile*
WP4	Viste	Cuzzocrea Allegra

8.1. Vista: *Categorie noleggiate*

La vista produce una tabella composta dalle categorie di bici noleggiate e per ognuna il numero di volte in cui sono state noleggiate.

```
create view numpercategoria as
select categoria, count(*) as n_volte
from noleggio, bici, modello
where noleggio.bici=bici.codice and
    bici.nomemodello=modello.nome and
          bici.marcamodello=modello.marca
group by categoria;
```

8.1.1. Query con Vista: Categoria privilegiata

Stampa la categoria di bici noleggiata più volte.

```
select distinct categoria, n_volte
from numpercategoria
where n_volte=(select max(n_volte)
                        from numpercategoria)
order by categoria;
```

8.2. Vista: Numero noleggi per cliente

La vista produce una tabella composta dai clienti che hanno effettuato almeno un noleggio ed il numero di noleggi che hanno effettuato.

```
create view numnoleggi as
select cliente, count(*) as num
from noleggio
group by cliente;
```

8.2.1. Query con vista: Cliente affezionato

Stampa il cliente che ha effettuato più noleggi

```
select cliente.*, num
from cliente, numnoleggi
where numnoleggi.cliente=cliente.documento and
          cliente.documento in (select cliente
                                                  from numnoleggi
                                                  where num=(select max(num)
                                                                        from numnoleggi));
```

8.3. Viste: Altri nei PV, Anziani nei PV, Studenti nei PV, Pagamenti Studenti e Anziani

- *La vista produce una tabella composta dai punti vendita in cui è stato effettuato almeno un noleggio da un cliente di tipo Altro e per ogni punto vendita il numero di noleggi effettuati da questo tipo di cliente.*

```
create view numaltroperpv as
select puntovendita, count(*) as numaltro
from riconsegna, noleggio, cliente
where riconsegna.noleggio=noleggio.codice and
        noleggio.cliente=cliente.documento and
        cliente.tipo like 'Altro'
group by puntovendita;
```

- *La vista produce una tabella composta dai punti vendita in cui è stato effettuato almeno un noleggio da un cliente di tipo Anziano e per ogni punto vendita il numero di noleggi effettuati da questo tipo di cliente.*

```
create view numanzianoperpv as
select puntovendita, count(*) as numanziano
from riconsegna, noleggio, cliente
where riconsegna.noleggio=noleggio.codice and
        noleggio.cliente=cliente.documento and
        cliente.tipo like 'Anziano'
group by puntovendita;
```

- *La vista produce una tabella composta dai punti vendita in cui è stato effettuato almeno un noleggio da un cliente di tipo Studente e per ogni punto vendita il numero di noleggi effettuati da questo tipo di cliente.*

```
create view numstudenteperpv as
select puntovendita, count(*) as numstudente
from riconsegna, noleggio, cliente
where riconsegna.noleggio=noleggio.codice and
        noleggio.cliente=cliente.documento and
        cliente.tipo like 'Studente'
group by puntovendita;
```

- *La vista produce una tabella composta dai noleggi effettuati da clienti studenti ed anziani ed il totale di ogni noleggio.*

```
create view pagamenti_st_an as
select noleggio.codice as noleggio, totale
from pagamento, noleggio, cliente
where pagamento.noleggio=noleggio.codice and
      noleggio.cliente=cliente.documento and
      cliente.tipo not like 'Altro';
```

8.3.1. Query con viste: Noleggi Bologna

Stampa il punto vendita situato a Bologna in cui ogni tipo di cliente ha effettuato un noleggio ed è stato effettuato il maggior numero di noleggi da clienti di tipo Altro. Stampare, inoltre, anche il guadagno ricavato da studenti e anziani.

```
select PV.*, sum(pagamenti_st_an.totale)
from puntovendita PV, riconsegna, noleggio, cliente, pagamenti_st_an
where PV.citta like 'Bologna' and
        PV.numid=riconsegna.puntovendita and
        riconsegna.noleggio=noleggio.codice and
        noleggio.cliente=cliente.documento and
        pagamenti_st_an.noleggio=noleggio.codice and
        PV.numid in (select numaltroperpv.puntovendita
                    from numaltroperpv, numanzianoperpv, numstudenteperpv
                    where numaltroperpv.puntovendita=numanzianoperpv.puntovendita and
                          numanzianoperpv.puntovendita=numstudenteperpv.puntovendita) and
        PV.numid in (select puntovendita
                    from numaltroperpv
                    where numaltro=(select max(numaltro)
                                    from numaltroperpv, puntovendita
                                    where numaltroperpv.puntovendita=puntovendita.numid and
                                          puntovendita.citta='Bologna'))
group by PV.numid, PV.via, PV.civico, PV.citta, PV.cap;
```

9. Trigger

9.1. Trigger inizializzazione

Workpackage	*Task*	*Responsabile*
WP1	Trigger inizializzazione/popolamento database	De Gruttola Andrea

9.1.1. Trigger1: OreRimanentiFirstTime

Il seguente trigger scatta all' inserimento di una tupla della relazione Acquisto ed esegue il calcolo delle ore rimanenti, inizializzando il rispettivo attributo nella relazione CardPrepagata.

```
create or replace function ProcOreRimanentiFirstTime() returns trigger as $$
        declare
        tariffaCard numeric(5,2) = 1.10;
        begin
        update cardprepagata set
            orerimanenti = new.costo/tariffaCard
                        where codice = new.cardprepagata;
        return new;
        end $$ language plpgsql;
create trigger OreRimanentiFirstTime
    after insert on acquisto
        for each row execute procedure ProcOreRimanentiFirstTime();
```

9.1.2. Trigger2: aggiornamentoOreRimanenti

Il seguente trigger scatta all' inserimento di una tupla della relazione Pagamento ed aggiorna l'attributo orerimanenti nella relazione CardPrepagata.
Nel caso in cui vi fossero problemi lancia un'eccezione.

```
create or replace function ProcaggiornamentoOreRimanenti() returns trigger as $$
    declare
        vecchieOrerimanenti integer;
        datafineApp date;
        datainizioApp date;
        orainizioApp time;
        orafineApp time;
        difData integer;
        difOre integer;
        app integer;
        sixty numeric(5,2)=60.0;
        app1 numeric(5,2);
        codiceApp char(5);
        begin
        select cardprepagata.orerimanenti, cardprepagata.codice into vecchieOrerimanenti, codiceApp
            from pagamento, noleggio, cliente, acquisto, cardprepagata
            where new.metodo='CardPrepagata' and
                                new.noleggio=noleggio.codice and
                                noleggio.cliente=cliente.documento and
                                noleggio.cliente=acquisto.cliente and
                                acquisto.cardprepagata=cardprepagata.codice ;
        select datainizio, orainizio, datafine, orafine into datainizioapp, orainizioApp, datafineApp, orafineApp
            from pagamento, prelievo, riconsegna
                        where new.noleggio=prelievo.noleggio and prelievo.noleggio=riconsegna.noleggio;
```

```
        if(vecchieOrerimanenti = 0) then
            raise exception 'Sulla cardprepagata non vi sono ore residue!!!';
        end if;
        --OreRimanenti = OreRimanenti - [24 * (DataFine - DataInizio) + (OraFine - OraInizio)]
        difData = DATEDIFF('day', datainizioApp, datafineApp);
        app = TIMEDIFF('minute', orainizioApp::time, orafineApp::time);
        if(app=0) then
           difOre=0;
        elsif(app>0 and app<30) then
           difOre=1;
        else
           app1=app/sixty;
                  if((app1-(app/60))>=0.5) then
                     difOre=app/60+1;
                  else
                     difOre=app/60;
                  end if;
        end if;
        if((vecchieOrerimanenti - ((24 * difData) + difOre))<0) then
                raise exception 'Le ore rimanenti non sono sufficienti per effettuare il pagamento';
        end if;
        update cardprepagata set
             orerimanenti = vecchieOrerimanenti - ((24 * difData) + difOre)
                          where cardprepagata.codice=codiceApp and
cardprepagata.orerimanenti=vecchieOrerimanenti;
        return new;
        end $$ language plpgsql;
create trigger aggiornamentoOreRimanenti
    after insert on pagamento
        for each row execute procedure ProcaggiornamentoOreRimanenti();
```

Note*: Le funzioni DATEDIFF e TIMEDIFF sono state adoperate per semplificare la gestione dei vincoli di derivazione e sono le seguenti:

```
CREATE OR REPLACE FUNCTION DATEDIFF (units VARCHAR(30), start_t TIMESTAMP, end_t TIMESTAMP)
   RETURNS INT AS $$
  DECLARE
   diff_interval INTERVAL;
   diff INT = 0;
   years_diff INT = 0;
  BEGIN
   IF units IN ('yy', 'yyyy', 'year', 'mm', 'm', 'month') THEN
    years_diff = DATE_PART('year', end_t) - DATE_PART('year', start_t);
    IF units IN ('yy', 'yyyy', 'year') THEN
     -- SQL Server does not count full years passed (only difference between year parts)
     RETURN years_diff;
    ELSE
     -- If end month is less than start month it will subtracted
     RETURN years_diff * 12 + (DATE_PART('month', end_t) - DATE_PART('month', start_t));
    END IF;
   END IF;
   -- Minus operator returns interval 'DDD days HH:MI:SS'
   diff_interval = end_t - start_t;
   diff = diff + DATE_PART('day', diff_interval);
   IF units IN ('wk', 'ww', 'week') THEN
    diff = diff/7;
    RETURN diff;
   END IF;
   IF units IN ('dd', 'd', 'day') THEN
    RETURN diff;
   END IF;
   diff = diff * 24 + DATE_PART('hour', diff_interval);
```

```
  IF units IN ('hh', 'hour') THEN
    RETURN diff;
  END IF;
  diff = diff * 60 + DATE_PART('minute', diff_interval);
  IF units IN ('mi', 'n', 'minute') THEN
    RETURN diff;
  END IF;
  diff = diff * 60 + DATE_PART('second', diff_interval);
  RETURN diff;
END;
$$ LANGUAGE plpgsql;



CREATE OR REPLACE FUNCTION TIMEDIFF (units VARCHAR(30), start_t TIME, end_t TIME)
  RETURNS INT AS $$
DECLARE
  diff_interval INTERVAL;
  diff INT = 0;
BEGIN
  -- Minus operator for TIME returns interval 'HH:MI:SS'
  diff_interval = end_t - start_t;
  diff = DATE_PART('hour', diff_interval);
  IF units IN ('hh', 'hour') THEN
    RETURN diff;
  END IF;
  diff = diff * 60 + DATE_PART('minute', diff_interval);
  IF units IN ('mi', 'n', 'minute') THEN
    RETURN diff;
  END IF;
  diff = diff * 60 + DATE_PART('second', diff_interval);
  RETURN diff;
END;
$$ LANGUAGE plpgsql;
```

9.1.3. Trigger3: checkTotalePagamento

Il seguente trigger scatta all' inserimento o alla modifica di una tupla della relazione Pagamento e verifica che l'attributo noleggio in Pagamento sia coerente con il noleggio effettuato.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcCheckTotalePagamento() returns trigger as $$
    declare
            datafineApp date;
            datainizioApp date;
            orainizioApp time;
            orafineApp time;
            difData integer;
            difOre integer;
            totaleCorretto numeric(5,2);
            valoreApp numeric(5,2);
            app integer;
            app1 numeric(5,2);
            sixty numeric(5,2)=60.0;
            begin
            select datainizio, orainizio, datafine, orafine into datainizioapp, orainizioApp, datafineApp, orafineApp
                from pagamento, prelievo, riconsegna
                                where new.noleggio=prelievo.noleggio and prelievo.noleggio=riconsegna.noleggio;
            select valore into valoreApp from pagamento, noleggio, tariffa
                    where new.noleggio=noleggio.codice and noleggio.tariffa=tariffa.nome;
            difData = DATEDIFF('day', datainizioApp, datafineApp);
            app = TIMEDIFF('minute', orainizioApp::time, orafineApp::time);
            if(app=0) then
```

```
            difOre=0;
        elsif(app>0 and app<30) then
            difOre=1;
        else
            app1=app/sixty;
                if((app1-(app/60))>=0.5) then
                    difOre=app/60+1;
                else
                    difOre=app/60;
                end if;
        end if;
        totaleCorretto = (24*difData + difOre)*valoreApp;
        if(new.metodo!='CardPrepagata' and new.totale != totaleCorretto) then
            raise notice 'Il totale NON è corretto!!! Dovrebbe essere: %; noleggio: %', totaleCorretto, new.noleggio;
            raise exception 'Il totale NON è corretto!!!';
        end if;
        return new;
        end $$ language plpgsql;
create trigger checkTotalePagamento
    after insert or update on pagamento
        for each row execute procedure ProcCheckTotalePagamento();
```

9.1.4. Trigger4: PrelievoBici

Il seguente trigger scatta all' inserimento di una tupla della relazione Prelievo e imposta a null l'attributo puntovendita della relazione Bici.

```
create or replace function ProcPrelievoBici() returns trigger as $$
    declare
        biciApp integer;
        begin
        select bici.codice into biciApp from noleggio, prelievo, bici
            where new.noleggio=noleggio.codice and noleggio.bici=bici.codice;
        update bici set
        puntovendita = null
        where bici.codice=biciApp;
        return new;
        end $$ language plpgsql;
create trigger PrelievoBici
    after insert on prelievo
        for each row execute procedure ProcPrelievoBici();
```

9.1.5. Trigger5: RiconsegnaBici

Il seguente trigger scatta all' inserimento di una tupla della relazione Riconsegna ed assegna alla bici il punto vendita in cui essa è stata riconsegnata.

```
create or replace function ProcRiconsegnaBici() returns trigger as $$
    declare
        biciApp integer;
        begin
        select bici.codice into biciApp from noleggio, riconsegna, bici
            where new.noleggio=noleggio.codice and noleggio.bici=bici.codice;
        update bici set
        puntovendita = new.puntovendita
        where bici.codice=biciApp;
```

```
        return new;
        end $$ language plpgsql;
create trigger RiconsegnaBici
    after insert on riconsegna
        for each row execute procedure ProcRiconsegnaBici();
```

9.2. Trigger per vincoli aziendali

Workpackage	***Task***	***Responsabile***
WP4	Trigger per vincoli aziendali	Cuzzocrea Allegra

9.2.1. Trigger1: CAPpuntoVenditaonlyDigits

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione PuntoVendita e verifica che il cap sia costituito da sole cifre.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcCAPpuntoVenditaOnlyDigits() returns trigger as $$
    begin
                if(new.cap !~'[0-9]{5}') then
                    raise exception 'cap in PuntoVendita deve essere di sole cifre';
                end if;
                return new;
                end $$ language plpgsql;
create trigger CAPpuntoVenditaonlyDigits
    before insert or update of cap on puntovendita
        for each row execute procedure ProcCAPpuntoVenditaOnlyDigits();
```

9.2.2. Trigger2: CAPclienteOnlyDigits

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione Cliente e verifica che il cap sia costituito da sole cifre.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcCAPclienteOnlyDigits() returns trigger as $$
    begin
                if(new.cap !~'[0-9]{5}') then
                    raise exception 'cap in Cliente deve essere di sole cifre';
                end if;
                return new;
                end $$ language plpgsql;
create trigger CAPclienteOnlyDigits
    before insert or update of cap on cliente
        for each row execute procedure ProcCAPclienteOnlyDigits();
```

9.2.3. Trigger3: MatricolaAddettoOnlyDigits

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione Addetto e verifica che la sua matricola sia costituita da sole cifre.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcMatricolaAddettoOnlyDigits() returns trigger as $$
    begin
```

```
                if(new.matricola !~'[0-9]{5}') then
                    raise exception 'matricola in Addetto deve essere di sole cifre';
                end if;
                return new;
                end $$ language plpgsql;
create trigger MatricolaAddettoOnlyDigits
    before insert or update of matricola on addetto
        for each row execute procedure ProcMatricolaAddettoOnlyDigits();
```

9.2.4. Trigger4: CodicePrepagataOnlyDigits

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione CardPrepagata e verifica che il codice sia costituito da sole cifre.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcCodicePrepagataOnlyDigits() returns trigger as $$
    begin
                if(new.codice !~'[0-9]{5}') then
                    raise exception 'codice in CardPrepagata deve essere di sole cifre';
                end if;
                return new;
                end $$ language plpgsql;
create trigger CodicePrepagataOnlyDigits
    before insert or update of codice on cardprepagata
        for each row execute procedure ProcCodicePrepagataOnlyDigits();
```

9.2.5. Trigger5: CodiceNoleggioOnlyDigits

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione Noleggio e verifica che il codice sia costituito da sole cifre.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcCodiceNoleggioOnlyDigits() returns trigger as $$
    begin
                if(new.codice !~'[0-9]{6}') then
                    raise exception 'codice in Noleggio deve essere di sole cifre';
                end if;
                return new;
                end $$ language plpgsql;
create trigger CodiceNoleggioOnlyDigits
    before insert or update of codice on noleggio
        for each row execute procedure ProcCodiceNoleggioOnlyDigits();
```

9.2.6. Trigger6: CartaCreditoOnlyDigits

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione Cliente e verifica che la cartadicredito sia costituita da sole cifre.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcCartaCreditoOnlyDigits() returns trigger as $$
    begin
                if(new.cartadicredito !~'[0-9]{16}') then
                    raise exception 'cartadicredito in Cliente deve essere di sole cifre';
                end if;
                return new;
```

```
        end $$ language plpgsql;
create trigger CartaCreditoOnlyDigits
    before insert or update of cartadicredito on cliente
        for each row execute procedure ProcCartaCreditoOnlyDigits();
```

9.2.7. Trigger7: NumTelefonoOnlyDigits

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione Telefono e verifica che il numero sia costituito da sole cifre.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcNumTelefonoOnlyDigits() returns trigger as $$
    begin
        if(new.numero !~'^\d{1,30}$') then
            raise exception 'numero in Telefono deve essere di sole cifre';
        end if;
        return new;
        end $$ language plpgsql;
create trigger NumTelefonoOnlyDigits
    before insert or update of numero on telefono
        for each row execute procedure ProcNumTelefonoOnlyDigits();
```

9.2.8. Trigger8: ResponsabileAncheAssegnato

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione Responsabile e verifica che il responsabile sia anche assegnato al punto vendita che dirige.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcResponsabileAncheAssegnato() returns trigger as $$
    begin
        if(not exists(select * from assegnazione where addetto=new.addetto and
puntovendita=new.puntovendita)) then
            raise exception 'Il responsabile deve anche essere assegnato al punto vendita che dirige';
        end if;
        return new;
        end $$ language plpgsql;
create trigger ResponsabileAncheAssegnato
    before insert or update of addetto on responsabile
        for each row execute procedure ProcResponsabileAncheAssegnato();
```

9.2.9. Trigger9: NOLEGGIOconsistency

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione Noleggio e verifica che la bici da noleggia sia disponibile (riconsegnata).
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcNOLEGGIOconsistency() returns trigger as $$
    begin
        if(exists(select noleggio.codice from noleggio where noleggio.bici=new.bici except
                    select riconsegna.noleggio from riconsegna, noleggio where
riconsegna.noleggio=noleggio.codice)) then
            raise exception 'Prima di creare il noleggio è necessario riconsegnare la bici';
        end if;
        return new;
        end $$ language plpgsql;
```

```
create trigger NOLEGGIOconsistency
    before insert or update on noleggio
        for each row execute procedure ProcNOLEGGIOconsistency();
```

9.2.10. Trigger10: PRELIEVObeforeRICONSEGNA

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione Riconsegna e verifica che esista il prelievo al noleggio associato.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcPRELIEVObeforeRICONSEGNA() returns trigger as $$
    begin
        if(not exists(select * from prelievo where prelievo.noleggio=new.noleggio)) then
           raise exception 'Prima di creare la riconsegna è necessario creare il prelievo';
        end if;
        return new;
        end $$ language plpgsql;
create trigger PRELIEVObeforeRICONSEGNA
    before insert or update on riconsegna
        for each row execute procedure ProcPRELIEVObeforeRICONSEGNA();
```

9.2.11. Trigger11: RICONSEGNAbeforePAGAMENTO

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione Pagamento e verifica che esista la riconsegna al noleggio associato.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcRICONSEGNAbeforePAGAMENTO() returns trigger as $$
    begin
        if(not exists(select * from riconsegna where riconsegna.noleggio=new.noleggio)) then
           raise exception 'Prima di creare il pagamento è necessario creare la riconsegna';
        end if;
        return new;
        end $$ language plpgsql;
create trigger RICONSEGNAbeforePAGAMENTO
    before insert or update on pagamento
        for each row execute procedure ProcRICONSEGNAbeforePAGAMENTO();
```

9.2.12. Trigger12: dateconsistency

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione Riconsegna e verifica che la data di fine del noleggio sia posteriore o al più uguale a quella di inizio.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcDateconsistency() returns trigger as $$
        declare
          app date;
                att time;
        begin
        select datainizio,orainizio into app,att from prelievo where prelievo.noleggio=new.noleggio;
        if(app > new.datafine) then
           raise exception 'La data di fine del noleggio deve essere posteriore o al più uguale a quella di inizio';
        end if;
        if((app = new.datafine) and (att>=new.orafine)) then
```

```
            raise exception 'Per noleggi che terminano nella stessa giornata in cui sono iniziati, l'ora di fine di riconsegna deve essere successiva a quella di inizio di prelievo';
        end if;
        return new;
        end $$ language plpgsql;
create trigger dateconsistency
    before insert or update on riconsegna
        for each row execute procedure ProcDateconsistency();
```

9.2.13. Trigger13: pagamentiConCardPrepagata

Il seguente trigger scatta all'inserimento o alla modifica di una tupla della relazione Pagamento e verifica che molteplici aspetti circa i suoi attributi siano corretti.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcpagamentiConCardPrepagata() returns trigger as $$
    declare
                anticipoApp numeric(5,2);
                tipoApp varchar(30);
                valoreApp numeric(5,2);
        begin
        select anticipo into anticipoApp from prelievo where prelievo.noleggio = new.noleggio;
        select tipo into tipoApp from noleggio where noleggio.codice = new.noleggio;
        select valore into valoreApp from tariffa,noleggio where tariffa.nome = noleggio.tariffa and noleggio.codice=new.noleggio;
        if((new.metodo='CardPrepagata' and anticipoApp != 0) or (new.metodo='CardPrepagata' and new.totale != 0)) then
          raise exception 'Per pagamenti effettuati con card prepagata l'anticipo ed il totale devono essere zero';
        end if;
        if(new.metodo !='CardPrepagata' and new.totale = 0) then
          raise exception 'Per pagamenti effettuati NON con card prepagata il totale deve essere maggiore di 0';
        end if;
        if(new.metodo !='CardPrepagata' and anticipoApp=0 and tipoApp='Giornaliero') then
          raise exception 'Un noleggio giornaliero NON pagato con card prepagata NON può avere anticipo=0';
        end if;
        if(tipoApp='Giornaliero' and anticipoApp != 0 and anticipoApp != valoreApp*24) then
          raise exception 'Per noleggi giornalieri con anticipo diverso da 0, l'anticipo stesso deve essere uguale alla tariffa moltiplicata per 24';
        end if;
        if(new.totale < anticipoApp) then
          raise exception 'Il totale del pagamento deve essere maggiore o uguale dell'anticipo';
        end if;
        return new;
        end $$ language plpgsql;
create trigger pagamentiConCardPrepagata
    before insert or update on pagamento
        for each row execute procedure ProcpagamentiConCardPrepagata();
```

9.2.14. Trigger14: OreRimanentiCardAcquistataMustBeNotNull

Il seguente trigger scatta alla modifica di una tupla della relazione CardPrepagata e verifica che una card prepagata acquistata non abbia un numero di ore rimanenti non definito.
Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcOreRimanentiCardAcquistataMustBeNotNull() returns trigger as $$
    begin
```

```
        if(exists(select from acquisto where acquisto.cardprepagata=new.codice) and new.orerimanenti is null)
then
          raise exception 'Una card prepagata che è stata acquistata NON può avere un numero di ore rimanenti
non definito';
        end if;
        return new;
        end $$ language plpgsql;
create trigger OreRimanentiCardAcquistataMustBeNotNull
    before update of orerimanenti on cardprepagata
        for each row execute procedure ProcOreRimanentiCardAcquistataMustBeNotNull();
```

9.2.15. Trigger15: oreRimanentiInserimentoCard

Il seguente trigger scatta all' inserimento o alla modifica di una tupla della relazione CardPrepagata e verifica che una card prepagata non acquistata abbia un numero di ore rimanenti non definito. Nel caso in cui così non fosse lancia un'eccezione.

```
create or replace function ProcOreRimanentiInserimentoCard() returns trigger as $$
    begin
        if(not exists(select * from cardprepagata, acquisto where new.codice=acquisto.cardprepagata) and
new.orerimanenti is not null) then
          raise exception 'Una card prepagata che NON è stata acquistata deve avere ore rimanenti uguali a null';
        end if;
        return new;
        end $$ language plpgsql;
create trigger oreRimanentiInserimentoCard
    before insert or update on cardprepagata
        for each row execute procedure ProcOreRimanentiInserimentoCard();
```

9.2.16. Trigger16: ImpedisciAggiornamentoTariffa

Il seguente trigger scatta alla modifica di una tupla della relazione Tariffa e la impedisce tranne che sull'attributo nome, lanciando un'eccezione.

```
create or replace function ProcImpedisciAggiornamentoTariffa() returns trigger as $$
  begin
  if(exists (select * from noleggio,tariffa where noleggio.tariffa=old.nome))then
    if(new.valore!=old.valore OR new.tipogiorno!=old.tipogiorno OR new.tipocliente!=old.tipocliente OR
      new.marcamodello!=old.marcamodello OR new.nomemodello!=old.nomemodello)then
                raise exception 'Impossibile modificare una tariffa a cui corrisponde almeno un noleggio';
        end if;
      end if;
      return new;
      end $$ language plpgsql;
create trigger ImpedisciAggiornamentoTariffa
    before update on tariffa
        for each row execute procedure ProcImpedisciAggiornamentoTariffa();
```