UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

REALIZZAZIONE DI UN SISTEMA INFORMATIVO TRAMITE DIVERSI DATABASE

RELATORE: Prof. Francesco Bombi

LAUREANDO: *Nicola Dalla Benetta*

ANNO ACCADEMICO 2010/2011

Alla mia famiglia.

Indice

Introduzione

Nell'elaborato proposto, si realizza un sistema informativo per la ditta Diesse Informatica s.r.l.; lo scopo del tirocinio, svolto presso la suddetta azienda, consiste nello studio delle caratteristiche specifiche che tale sistema deve possedere, per giungere poi a un'implementazione mediante diverse base di dati opensource presenti in commercio. Per poter capire la necessità di un sistema informativo per l'azienda in causa è utile conoscerne la storia.
La ditta Diesse Informatica s.r.l. nasce nel 1999 a seguito dell'aumento delle richieste di acquisto e riparazione hardware e software nella provincia di Verona. In una prima fase di sviluppo tale azienda vendeva al dettaglio per utenti singoli e interveniva in piccole riparazioni. Successivamente nel 2002 a causa della domanda crescente di tali prodotti e servizi, l'azienda si è ampliata, creando un centro commerciale e un centro per l'assistenza tecnica di notevoli dimensioni. A distanza di 8 anni, il sempre maggior numero di clienti, tra cui numerose aziende, ha reso necessaria la creazione del sistema informativo in questione, per una gestione efficiente del materiale e dei rapporti con le varie entità con cui Diesse Informatica si deve relazionare. (Cfr [2])
Per raggiungere lo scopo prefissato e dati questi presupposti, si sono attraversate varie fasi di studio. Dopo una fase iniziale in cui sono stati raccolti tutti i dati necessari, si è passati quindi alla progettazione concettuale e logica del sistema informativo. Infine si è scelto di implementarlo mediante i seguenti database opensource distinti (se sono state usate interfacce grafiche queste sono specificate):

- MySQL;
- Postgres mediante PgAdmin III;
- HSQLDB mediante OpenOffice-Database;

Infine è stata affrontata una fase di testing delle tre implementazioni, in cui sono stati valutati pro e contro di ciascun database, sulla base di tali valutazioni si è scelta la definitiva realizzazione ovvero quella risultata maggiormente efficente.

Capitolo 1

Raccolta e analisi dei requisiti

Al fine di realizzare i diversi database, è necessario applicare una strategia di progettazione di tipo Top-down, partendo dunque da un'analisi generale, si procede passo dopo passo a raffinamenti e specificazioni successive degli elementi che entrano in gioco man mano. Qui di seguito vengono dunque riportati, in modo schematico, gli elementi fondamentali cui sistema informativo dovrà fare riferimento.

1.1 I Prodotti

La Base di dati (che indicheremo d'ora in poi con DB), deve monitorare i rapporti esistenti tra Diesse Informatica e i prodotti che devono essere tenuti sotto controllo.

Una prima distinzione che si evidenzia è quella tra:

- Prodotti destinati alla vendita: questo primo punto permette di sottolineare quali siano prodotti destinati alla vendita, con eventuali promozioni, sconti e offerte.(Cfr [1])
- Prodotti destinati alla riparazione: questo punto evidenzia invece quei prodotti che saranno utilizzati nella riparazione (soprattutto hardware) di prodotti in possesso di clienti.

Per ogni prodotto destinato alla vendita è necessario considerare alcune informazioni, presenti nella tabella 1.1.

Tabella 1.1: Caratteristiche prodotti destinati alla vendita

Caratteristica	**Breve Descrizione**
Codice di riferimento del prodotto	Ogni prodotto ha un suo codice identificativo
Nome Prodotto	Utile nel rapportarsi con i clienti
Produttore	Necessario per sapere il referente in caso di guasti durante la garanzia
Tipologia del prodotto	Tale campo evidenza l'ambito cui appartiene un prodotto
Descrizione	Utile per sapere le funzionalità di base di un determinato prodotto
Durata garanzia	Evidenzia la durata della garanzia su un certo prodotto
Prezzo	In tale campo sarà presente il prezzo del prodotto
Livello Qualità	Evidenzia con un indice compreso tra 0 e 10 il livello qualitativo del prodotto
Quantità giacenza	In questo campo sarà presente la quantità di risorse presente in magazzino

Per ogni prodotto destinato alla riparazione è invece necessario considerare un altro tipo di informazioni, riportare nella tabella 1.2.

1.2 I Fornitori

Il DB deve inoltre monitorare i rapporti tra Diesse Informatica e i fornitori dei prodotti, i rapporti esistenti tra essi sono principalmente di due tipi:

- Ritiro da parte dei fornitori dei prodotti invenduti;
- Ritiro da parte di Diesse dei prodotti ordinati (ovvero la consegna di questi ultimi);

Per ogni fornitore, il DB deve tenere conto delle informazioni presenti nella tabella 1.3.

Tabella 1.2: Caratteristiche prodotti riparazione

Caratteristica	**Breve Descrizione**
Codice di riferimento del prodotto	Ogni prodotto ha un suo codice identificativo
Nome Prodotto	Utile nel rapportarsi con i clienti
Produttore	Utile in caso di problemi con il prodotto
Descrizione	Utile per sapere le funzionalità di base di un determinato prodotto
Durata garanzia	Evidenzia la durata della garanzia
Prezzo	In tale campo sarà presente il prezzo del prodotto
Costo Lavoro	Evidenzia il costo del lavoro svolto
Livello Qualità	Evidenzia con un indice compreso tra 0 e 10 il livello qualitativo del prodotto
Quantità giacenza	In questo campo sarà presente la quantità di risorse presente in magazzino

Tabella 1.3: Caratteristiche Fornitori

Caratteristica	**Breve Descrizione**
Partita IVA	Questo campo evidenzia la partita IVA dell'azienda
Nome dell'azienda Fornitrice	Denominazione della ditta
Cognome, Nome del titolare	Indica Nome e Cognome del titolare
Indirizzo	Recapito fisico della ditta
Contatti	Campo contenente evenuali numeri di telefono, indirizzi mail.

1.3 I Clienti

Un altro rapporto importante che il DB deve gestire, è quello esistente tra Diesse Informatica e i clienti, i clienti devono essere suddivisi in due tipologie in base al rapporto che hanno con Diesse:

- Utenti singoli/privati; tale categoria è suddivisibile a sua volta in altre due sottocategorie:

  - Utenti per la vendita;
  - Utenti per riparazioni;

- Aziende che necessitano di supporto informatico; anche tale categoria è suddivisibile in sottocategorie:

  - Aziende che acquistano prodotti;
  - Aziende che necessitano di riparazione e manutenzione;

Per ogni cliente il DB deve considerare le informazioni riportate in tabella 1.4.

Tabella 1.4: Caratteristiche Clienti

Caratteristica	**Breve Descrizione**
Codice cliente	Un codice sequenziale per identificare il cliente senza invaderne la privacy
Costo dell'operazione	Tale campo evidenzia quanto il cliente deve pagare per il servizio richiesto
Contatto	Un campo contenente un recapito con cui contattare il cliente
Descrizione del lavoro svolto	Una breve descrizione su quanto è stato fatto per il cliente

Nella tabella 1.5 si riporta il listino prezzi per i clienti che richiedono un qualche servizio (se i clienti sono intesi come aziende il prezzo è riportato tra parentesi e sarà minore per incentivare queste a rivolgersi a Diesse).

1.4 Ulteriori servizi informativi

Il DB deve inoltre offrire un servizio di consulenza che possa facilitare le scelte di acquisto alla clientela, è quindi necessario rendere disponibili le seguenti informazioni:

- Classifica dei prodotti più venduti;
- Classifica dei prodotti più economici;
- Classifica dei prodotti di maggior qualità;

Tabella 1.5: Listino dei prezzi

Descrizione	**Prezzo**
Prodotto	Prezzo del prodotto da listino
Intervento di riparazione	25,00 € (20,00 €) + prezzo di eventuali componenti nuove
Intervento di riparazione urgente	50,00 € (40,00 €) + prezzo eventuali componenti nuove
Installazione sistema operativo	20 € (15 €)
Installazione programmi	15 € (10€)
Aggiornamento antivirus,programmi e sistemi operativi	20 € (15 €)

- Elenco delle promozioni del mese;

Tali informazioni risultano essere molto utili, infatti possono aiutare il cliente nella scelta al momento dell'acquisto; ad esempio se volesse un prodotto di alta qualità e contemporaneamente molto venduto, tramite queste classifiche, è possibile aiutarlo ad indirizzarsi verso un certo prodotto. L'elenco delle promozioni del mese corrente permette invece di aiutare il cliente a scegliere un prodotto di qualità in promozione garantendo un buon risparmio.

1.5 Operazioni sul database

Si riportano tabella 1.6 le principali operazioni richieste al DB, la frequenza osservata durante il tirocinio è approssimata rispetto ai valori reali; i valori indicati sono comunque molto fedeli alla realtà effettiva.
É doveroso ribadire come sia importante sapere prima della costruzione fisica del DB quali saranno le grandezze delle operazioni in gioco, in quanto è necessario che il DB stesso possa supportare le differenti operazioni e sopportarne il carico. il DB deve riuscire a velocizzare le operazioni frequenti, anche se a spese di operazioni meno frequenti (Cfr [3]).

Tabella 1.6: Tavola delle Operazioni

Tipo Operazione	**Frequenza**
Inserimento nuovo fornitore	1 volte a bimestre
Inserimento nuovo prodotto hardware	12 volte a settimana
Inserimento nuovo prodotto software	15 volte a settimana
Inserimento nuovo codice cliente	60 volte a settimana
Inserimento nuova riparazione utente	30 volte a settimana
Inserimento nuova riparazione azienda	27 volte a settimana
Visualizzazione elenco fornitori	2 volte a settimana
Visualizzazione elenco codici clienti	30 volte a settimana
Visualizzazione elenco prodotti venduti	5 volte a settimana
Visualizzazione elenco prodotti presenti in magazzino	2 volte a settimana
Visualizzazione elenco prodotti e loro descrizione	20 volte a settimana
Visualizzazione elenco ordini fornitori	6 volte a settimana
Visualizzazione elenco prodotti da restituire ai fornitori	3 volte a settimana
Aggiornamento prezzi dei prodotti	6 volte a settimana
Aggiornamento materiale presente in magazzino	20 volte a settimana
Aggiornamento dei prodotti	5 volte a settimana
Aggiornamento aziende visitate per riparazioni	30 volte a settimana
Aggiornamento prodotti venduti	25 volte a settimana
Visualizzazione classifica prodotti più venduti	1 volte a settimana
Visualizzazione classifica prodotti più economici	1 volte a settimana
Visualizzazione classifica prodotti di maggior qualità	1 volte a settimana
Visualizzazione promozioni del mese	5 volte al mese

1.6 Stima del carico

Si riporta ora una stima approssimata, ma attendibile, delle dimensioni dei dati della realtà che si è analizzato, nell'elenco sotto riportato, sono evidenziati le principali entità considerate e rispettiva mole.
Come nel caso del paragrafo 1.5 è necessario avere ben chiara la quantità di dati che il sistema informativo deve supportare, al fine di rendere maggiormente ottimizzato il database; risulta altresì importante ipotizzare come tali dati varieranno (presumibilmente) nei prossimi anni in modo tale che il DB possa essere in grado di gestire l'aumento (o la diminuzione) dei dati.

- Situazione attuale:
  - Quantità clienti con cui Diesse ha rapporto mensilmente: 150
  - Quantità dei fornitori: 30
  - Quantità prodotti presenti in magazzino: 2000
  - Numero clienti medio giornaliero: 20
- Variazioni previste:
  - Nuovi clienti: +3%;
  - Nuovi fornitori: +4%;
  - Fornitori revocati: -3%;
  - Giacenza in magazzino: circa costante;
  - Numero clienti medio mensile: +1%;

Capitolo 2

Progettazione concettuale

In questo capitolo, si affronta la creazione di un modello concettuale basato sullo schema ER seguendo l'iter classico (Cfr. [10]).

2.1 Schema scheletro

Partendo dallo schema scheletro grezzo, si procede, per raffinamenti successivi, all'espansione di tale schema, giungendo quindi allo schema ER definitivo.

Nel capitolo sono state evidenziate le entità principali del nostro database ovvero:

- Prodotti;
- Fornitore;
- Cliente;

Le relazioni che intercorrono tra tali entità, in una prima fase di progettazione, sono le seguenti:

- Consegna: evidenzia i rapporti tra Fornitori e Prodotti;
- Vendita: evidenzia i rapporti tra Prodotti e Clienti;

Il primo schema è mostrato in figura 2.1.

Si noti che tale schema può essere raffinato immediatamente, infatti le due relazioni *Vendita* e *Consegna* possono essere scisse in sotto-relazioni che meglio ne

Figura 2.1: Primo schema concettuale

specificano il rapporto; la relazione *Consegna* può essere divisa nelle due seguenti relazioni:

- Consegna: evidenzia il fatto che un fornitore consegna alla ditta certi prodotti;
- Ritiro: evidenzia il rapporto per cui un fornitore ritira dalla ditta prodotti invenduti, rotti o da sostituire.

La relazione *Vendita* deve essere suddivisa invece nelle seguenti relazioni:

- Vendita: evidenzia il rapporto tra le entità **Prodotti** e **Clienti** nel quale c'è un acquisto da parte del cliente di un qualche prodotto reperibile presso la ditta;
- Riparazione: evidenzia il rapporto tra le stesse entità prese in considerazione nel punto sopra, in cui il cliente chiede la riparazione di un qualche prodotto di proprietà;

In figura 2.2 è evidenziata questa fase di raffinamento.

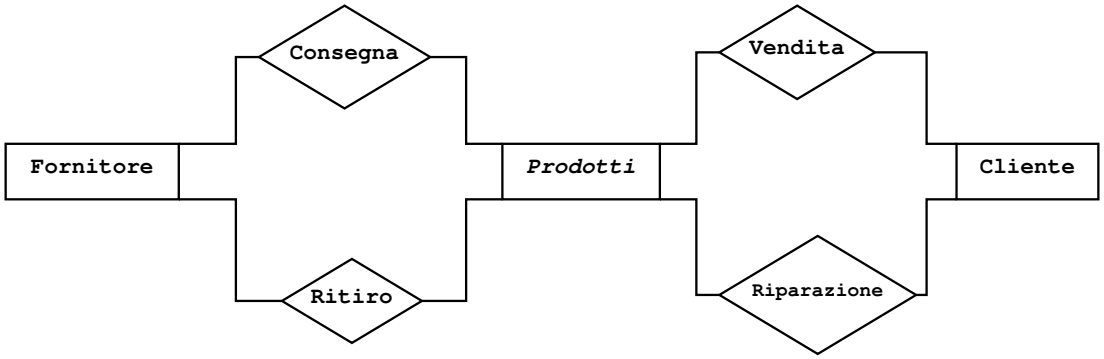

Figura 2.2: Schema dopo una prima ristrutturazione

2.2 Ulteriori raffinamenti

In tale paragrafo saranno riportati i successivi miglioramenti dello schema; per ogni elaborazione, sarà riportata solo la porzione di schema che viene modificata per rendere più scorrevole la lettura, lo schema completo si trova in figura 2.6

2.2.1 Generalizzazioni

In questo paragrafo si elencheranno le generalizzazioni presenti nello schema e le relative specializzazioni.
L'entità *Cliente* è infatti una generalizzazione delle due specializzazioni seguenti:

- *Utente privato*: evidenzia l'entità che rappresenta i singoli clienti che si rivolgono a Diesse Informatica per acquisti ridotti, come un pc, una scheda madre, un dvd o per riparazioni di varia natura;
- *Azienda*: evidenzia l'entità che rappresenta le aziende che si rivolgono a Diesse Informatica per acquisti consistenti o manutenzioni di server aziendali o ancora il rinnovamento della parte informatica dell'azienda;

Si nota immediatamente che tale generalizzazione è ti tipo **disgiunta totale**, poichè ogni cliente è esclusivamente o utente privato o azienda; la rappresentazione di tale generalizzazione è riportata in figura 2.3.

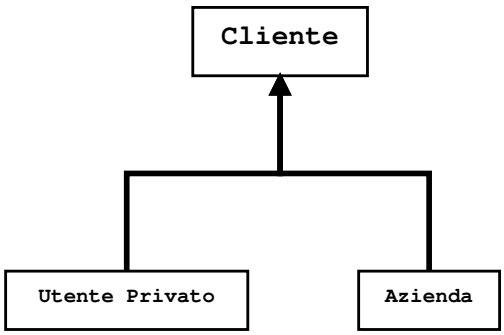

Figura 2.3: Generalizzazione *Cliente* e sue specializzazioni

Per quanto concerne l'entità *Prodotti*, da quanto è emerso dall'analisi dei requisiti nel paragrafo 1.1, si evidenzia che essa è una generalizzazione delle due entità specializzanti sotto elencate:

- *Vendibile*: evidenzia l'entità che rappresenta quella categoria di prodotti destinati alla vendita.

- *Prodotto per riparazioni*: evidenzia l'entità che rappresenta quella categoria di prodotti destinata alla riparazione di macchine.

Si può notare che tale generalizzazione, poiché ogni prodotto è o vendibile o destinato alle riparazioni (o entrambi), è una generalizzazione di tipo **sovrapposta totale**; si riporta la porzione di schema concettuale che evidenzia tale generalizzazione nella figura 2.4.
Per poter, a questo punto, rendere lo schema concettuale più chiaro, è necessario dividere le relazioni *Vendita* e *Riparazione* in due relazioni distinte ciascuna, in modo tale che sia palese a che tipologia di cliente è rivolto un certo servizio.
La relazione *Vendita* verrà quindi suddivisa in:

- *Vendita al dettaglio*: evidenzia il servizio di vendita ai clienti singoli;
- *Vendita all'ingrosso*: evidenzia il servizio di vendita alle aziende, cioè la vendita di grandi quantità di prodotti;

La relazione *Riparazione* verrà invece suddivisa in:

- *Riparazione-Azienda*: evidenzia il servizio di riparazione rivolto alle aziende, poichè queste hanno priorità rispetto ai clienti singoli;
- *Riparazione-Utente*: evidenzia il servizio di riparazione rivolto ai clienti singoli;

Lo schema risultante è riportato in figura 2.5, le linee tratteggiate servono semplicemente per evitare sovrapposizione poco comprensibili.

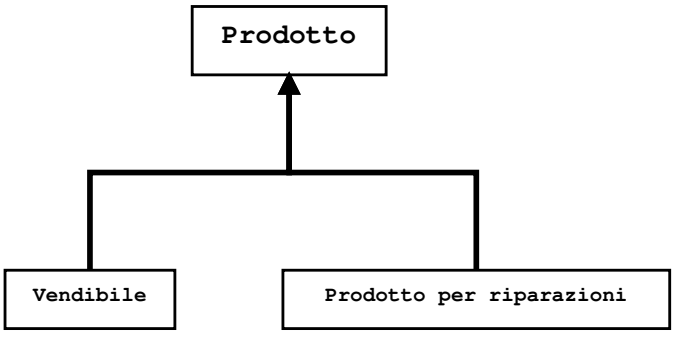

Figura 2.4: Generalizzazione *Prodotto* e sue specializzazioni

Si procede ora con lo studio delle entità e delle relazioni una alla volta, procedendo con l'evidenziarne gli attributi, definendone poi le chiavi e infine specificandone la cardinalità per le relazioni.

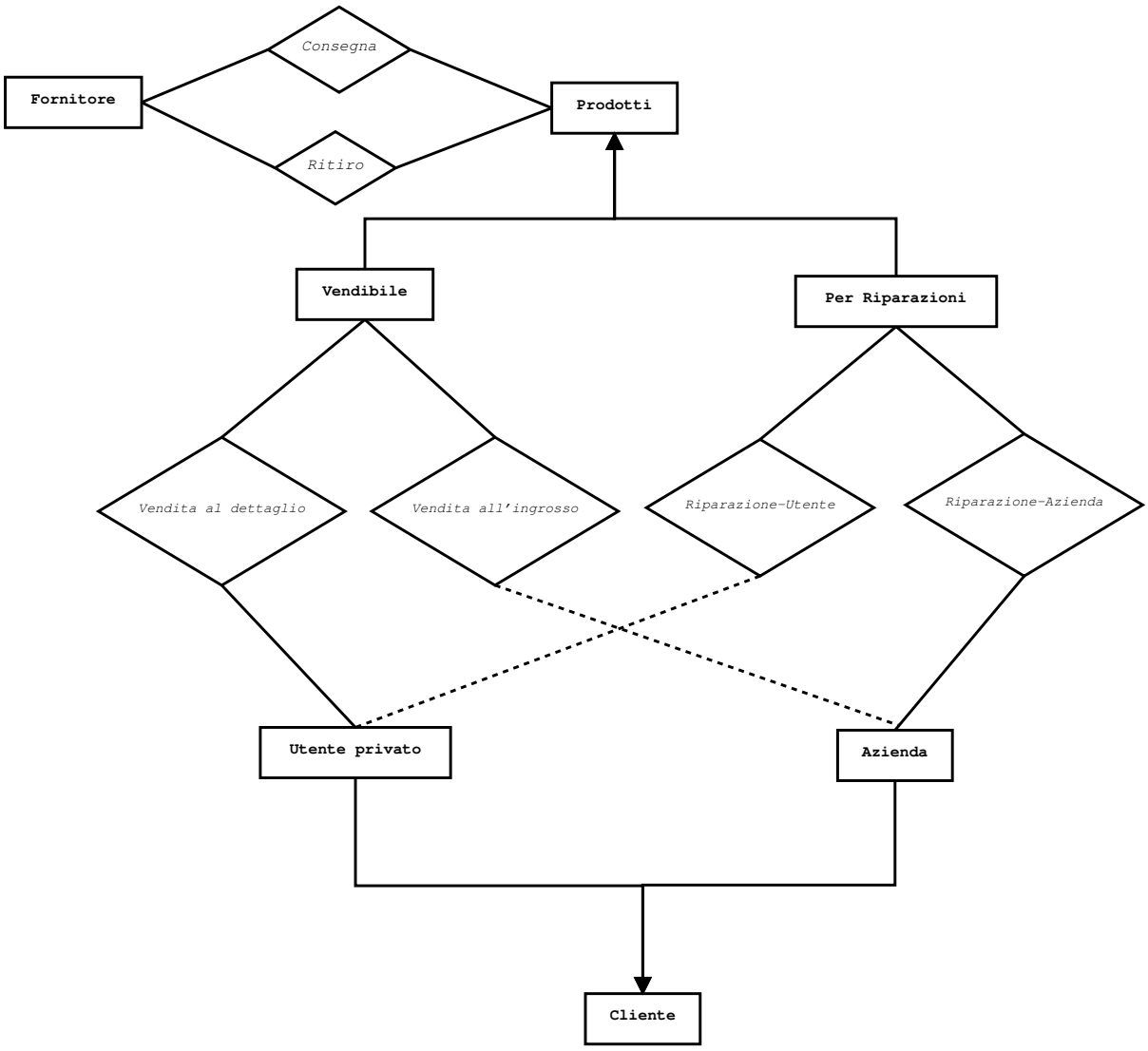

Figura 2.5: Schema progettuale in una seconda fase di raffinamento

2.3 Analisi entità: attributi e chiavi primarie

Per poter svolgere uno studio delle entità che risulti semplice alla lettura, si procede nel seguente modo: nelle tabelle vengono analizzate le entità, nella prima colonna viene riportato l'attributo (esso sarà sottolineato nel caso sia chiave primaria) nella seconda colonna sarà invece presente una breve descrizione dell'attributo stesso; nel caso in cui gli attributi fossero composti o multivalore viene evidenziata tale peculiarità tra parentesi tonde specificandone la tipologia.

Nella tabella 2.1 si evidenzia l'entità Fornitore.

Nella tabella 2.2 si riporta l'entità riguardante i prodotti destinati alla vendita.

Tabella 2.1: Fornitore

Attributo	**Breve Descrizione**
PartitaIVA	Poiché la partita IVA è unica per ogni azienda, questo attributo è stato scelto come chiave primaria
NomeFornitore	Campo in cui è presente il nome della società che fornisce il materiale
NomeTitolare	Indica Nome e Cognome del titolare
Indirizzo	Indirizzo presso cui è rintracciabile la ditta (ATTRIBUTO COMPOSTO)
Contatti	Campo contenente numeri di telefono, indirizzi mail (ATTRIBUTO MULTIVALORE).

Tabella 2.2: Prodotto vendibile

Attributo	**Breve Descrizione**
CodiceProdotto	Ogni prodotto ha un suo codice identificativo
Denominazione	Nome con cui è conosciuto un certo prodotto
Produttore	Necessario per sapere il referente in caso di problemi presentatisi nel corso della garanzia
Tipologia	Un attributo che evidenza la famiglia di appartenenza del prodotto (ATTRIBUTO MULTIVALORE)
Descrizione	Utile per sapere le funzionalità di base di un determinato prodotto
DurataGaranzia	Evidenzia la durata della garanzia
Prezzo	Tale attributo evidenzia il prezzo del prodotto
Qualità	Evidenzia con un indice compreso tra 0 e 10 il livello qualitativo del prodotto
Giacenza	In questo campo sarà presente la quantità di risorse presente in magazzino

Nella tabella 2.3 è presente l'entità dei prodotti per le riparazioni.

Nella tabella 2.4 è presente l'entità che concerne il cliente singolo, ovvero l'utente privato.

Nella tabella 2.5 si riporta invece l'entità che rappresenta il cliente inteso come azienda.

Tabella 2.3: Prodotto per riparazione

Attributo	**Breve Descrizione**
CodiceProdotto	Ogni prodotto ha un suo codice identificativo
Denominazione	Utile per rapportarsi con i clienti
Produttore	Utile in caso di problemi col prodotto
Descrizione	Utile per sapere le funzionalità di base di un determinato prodotto
DurataGaranzia	Evidenzia la durata della garanzia
Prezzo	In tale campo sarà presente il prezzo del prodotto
Preventivo	Evidenzia il costo ipotizzato per una certa riparazione
Qualità	Evidenzia con un indice compreso tra 0 e 10 il livello qualitativo del prodotto
Quantità giacenza	In questo campo sarà presente la quantità di risorse presente in magazzino
LavoroDaSvolgere	In tale attributo è presente l'uso che concerne il prodotto

Tabella 2.4: Utente Privato

Attributo	**Breve Descrizione**
CodiceCliente	Un codice sequenziale per identificare il cliente senza invaderne la privacy
Contatti	Un campo contenente un recapito con cui contattare il cliente (ATTRIBUTO MULTIVALORE)
DescrizioneLavoro	Una breve descrizione di ciò che l'utente ha richiesto

Per quanto riguarda le entità *Cliente* e *Prodotto* essi non hanno alcun attributo in più rispetto a quelli che derivano dalle specializzazioni.

2.4 Analisi relazioni: descrizione e cardinalità

Nel seguente paragrafo si affronta l'analisi delle relazioni presenti nello schema in figura 2.5. Per semplificare la lettura e la scorrevolezza del capitolo, si è deciso di procedere come nel capitolo precedente; ovvero riassumendo le relazioni in tabelle che risultino esplicative e sintetiche al contempo. In ogni tabella sarà presente

Tabella 2.5: Azienda

Attributo	**Breve Descrizione**
PartitaIVA	Serve per i pagamenti e per identificare l'azienda
CodiceAzienda	Un codice sequenziale per identificare l'ordine conologico degli interventi
NomeTitolare	Il nome completo del titolare
Contatti	Un campo contenente un recapito con cui contattare il cliente (ATTRIBUTO MULTIVALORE)
DescrizioneLavoro	Una breve descrizione del lavoro o dell'acquisto che l'azienda richiede

il nome della relazione, una breve descrizione e la cardinalità che essa possidere rispetto alle entità circostanti; inoltre nel caso in cui la relazione necessitasse di eventuali attributi essi sono riportati nell'ultima riga della tabella.
La tabella 2.6 illustra la relazione *Consegna*.

Tabella 2.6: Consegna

Descrizione	Assegna a ogni prodotto il relativo fornitore
Cardinalità	**Uno a Molti**; infatti a ogni fornitore sono associati più prodotti ma ogni prodotto è riconducibile a un solo fornitore
Attributi	
DataConsegna	Utile per sapere quando un determinato prodotto è giunto in magazzino

La tabella 2.7 evidenzia ciò che concerne la relazione *Ritiro*, ovvero la relazione esistente tra fornitori e quell'insieme di prodotti che per motivi diversi (un esempio sono i prodotti invenduti).

Nella tabella 2.8 è invece riportata la relazione che intercorre tra i prodotti vendibili e i clienti singoli.

Simmetricamente nella tabella 2.9 si trova la relazione chiamata *Vendita all'ingrosso* che mette in relazione i prodotti destinati alla vendita con le aziende.

Per quanto concerne invece l'analisi della relazione *Riparazione Utente*, esisitente tra il materiale per le riparazioni e gli interventi di riparazione fatti per gli

Tabella 2.7: Ritiro

Descrizione	Assegna a ogni prodotto da ritirare il fornitore che se ne occupa
Cardinalità	**Uno a Molti**; infatti ogni fornitore può ritirare più prodotti,in compenso ogni prodotto è ritirabile da un solo fornitore
Attributi DataRitiro	Utile per sapere quando un determinato prodotto è stato ritirato

Tabella 2.8: Vendita al Dettaglio

Descrizione	Assegna a ogni utente privato il prodotto o i prodotti acquistati
Cardinalità	**Uno a Molti**; infatti a ogni utente singolo possono esser associati più prodotti ma ogni prodotto è associato un solo cliente
Attributi DataVendita	Utile per sapere quando un determinato prodotto è stato venduto al cliente singolo

Tabella 2.9: Vendita all'ingrosso

Descrizione	Assegna a ogni prodotto l'azienda che l'ha acquistato
Cardinalità	**Uno a Molti**; infatti a ogni azienda sono associati più prodotti ma ogni prodotto è vendibile a una solo azienda
Attributi DataVendita	Utile per sapere quando un determinato prodotto è stato venduto ad un'azienda

utenti, si veda 2.10.

Simmetricamente la relazione *Riparazione-Azienda* è approfondita in tabella 2.11

Completata quest'analisi delle entità e delle relazioni, è possibile ora riportare lo schema completo, con attributi e chiavi, tale schema si trova in figura 2.6

Tabella 2.10: Riparazione-Utente

Descrizione	Assegna a ogni prodotto destinato alle riparazioni, il cliente che ha richiesto un certo intervento sulla propria macchina
Cardinalità	**Uno a Molti**; infatti a ogni cliente possono esser associati più prodotti per la riparazione richiesta, ma ogni prodotto è riconducibile a un solo cliente
Attributi	
DataRiparazione	Evidenzia quando è stata compiuta una certa riparazione per un cliente
PrezzoFinale	Indica il costo totale del lavoro e dei pezzi eventualmente sostituiti

Tabella 2.11: Riparazione-Azienda

Descrizione	Assegna a ogni prodotto l'azienda che ha fatto richiesta di un certo intervento di riparazione
Cardinalità	**Uno a Molti**; infatti a ogni azienda sono associati uno o più prodotti per la riparazione,ma ogni prodotto è associabile ad una sola azienda per quell'intervento di riparazione
Attributi	
DataRiparazione	Utile per sapere quando è stata fatta una certa riparazione per un'azienda
PrezzoFinale	Indica il costo totale del lavoro e dei pezzi eventualmente sostituiti

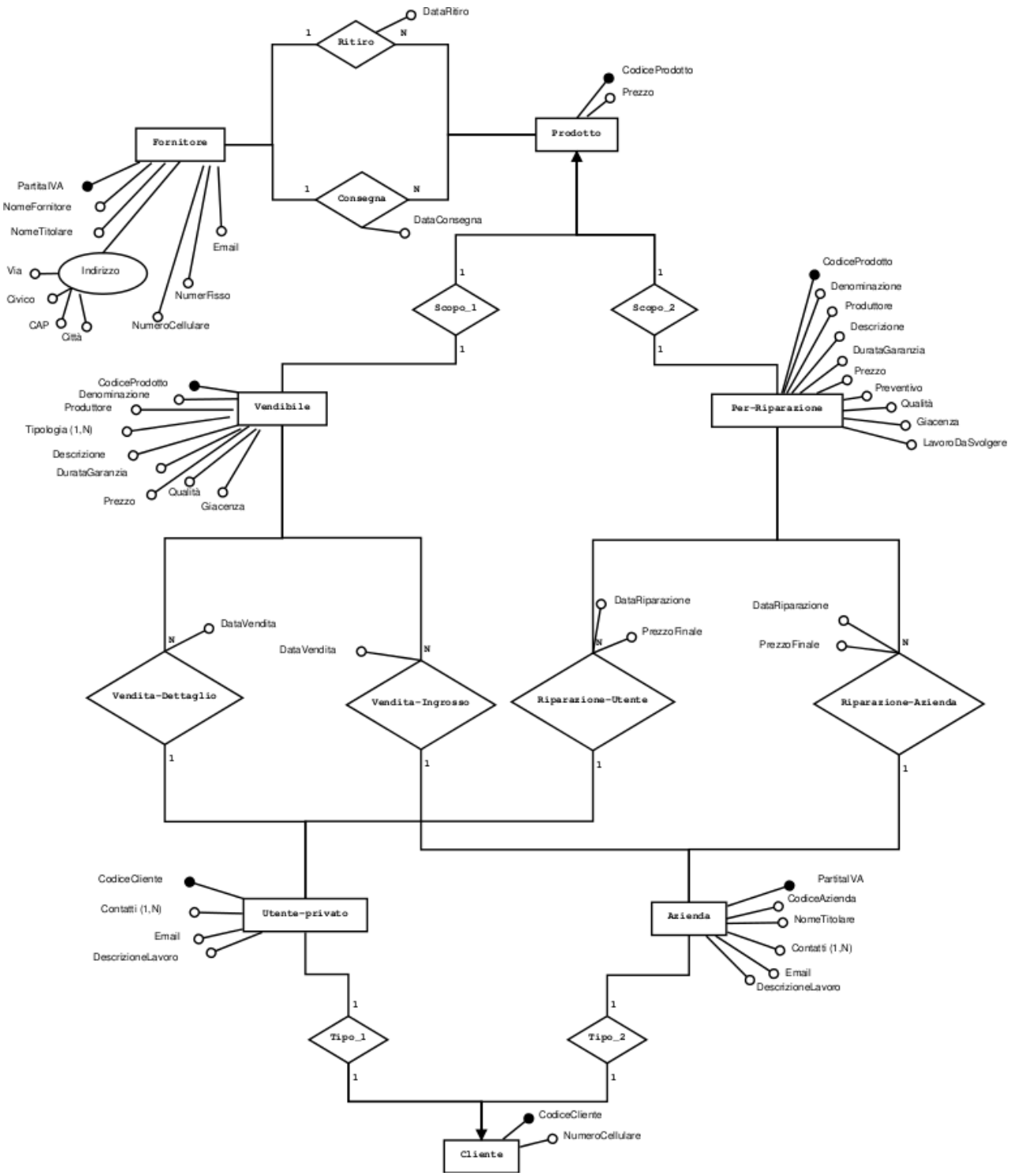

Figura 2.6: Schema E.R. completo

Capitolo 3

Progettazione logica

Questo capitolo si pone l'obbiettivo di affrontare la progettazione logica, ovvero, partendo dallo schema ER studiato e realizzato nel capitolo 2, tradurre quanto considerato nello schema logico corrispondente.
Per svolgere tale lavoro è necessario affrontare il suddetto problema in due passi:

- Ristrutturazione dello schema ER
- Traduzione nel modello relazionale

Per quanto riguarda il primo punto è necessario modificare lo schema eliminando tutte quelle particolarità che non possono essere tradotte immediatamente nello schema relazionale ovvero:

- Ridondanze
- Generalizzazioni o specializzazioni
- Attributi composti o multivalore
- Identificatori primari

3.1 Analisi delle ridondanze

Per quanto concerne le ridondanze, è evidente dallo schema riportato in figura 3.3 che non è presente alcuna ridondanza in quanto ogni entità che partecipa alle relazioni presenti, è necessaria. Questo punto è quindi già risolto e non necessita di ulteriori approfondimenti.

3.2 Geralizzazioni o specializzazioni

In questo paragrafo si spiega come eliminare le due generalizzazioni presenti nello schema 3.3, ovvero le generalizzazioni chiamate *Cliente* e *Prodotto*. Nel capitolo 2.2.1 è già stato evidenziato come le due entità sopra citate siano rispettivamente di tipo **Disgiunta totale** e **Sovrapposta totale**. Nel caso delle generalizzazioni vi sono due metodi di approccio alla loro risoluzione:

- Accorpamento dell'entità padre nelle entità figlie;
- Sostituzione delle generalizzazioni con associazioni;

Nel caso preso in esame si sceglie di adottare, per entrambe le generalizzazioni, la seconda soluzione; infatti si ritiene utile ai fine dello schema relazionale di tenere intalterate le entità **Prodotto** e **Cliente**.

3.2.1 Cliente

La sostituzione della generalizzazione cliente con un associazione prevede di sostituire tale generalizzazione con due associazioni che legano Cliente a Utente-Privato e Azienda. Si riporta lo schema modificato in figura 3.1, come si può notare da tale figura sono state introdotte le due associazione *Tipo_1* e *Tipo_2* che legano il cliente alla tipologia di cui fa parte, la loro cardinalità è 1:1 in quanto ogni utente è cliente e ogni cliente può essere solo un utente, in modo simmetrico si compie lo stesso ragionamento con l'azienda. L'attributo che identifica Cliente è il CodiceCliente o CodiceAzienda con cui si identificavano le entità figlie.

3.2.2 Prodotto

La sostituzione della generalizzazione prodotto con un associazione prevede di sostituire tale generalizzazione con due associazioni che legano Prodotto a Vendibile e Prodotto PerRiparazione. Si riporta la porzione di schema che viene modificata in figura 3.2, come si può notare da tale figura sono state introdotte le due associazione *Scopo_1* e *Scopo_2* che legano il prodotto allo scopo cui servono, la loro cardinalità è 1:1 in quanto ogni prodotto è vendibile (nella relazione Scopo_1) e ogni prodotto vendibile può essere solo un prodotto, in modo simmetrico si compie lo stesso ragionamento con I prodotti per le riparazioni. L'attributo che

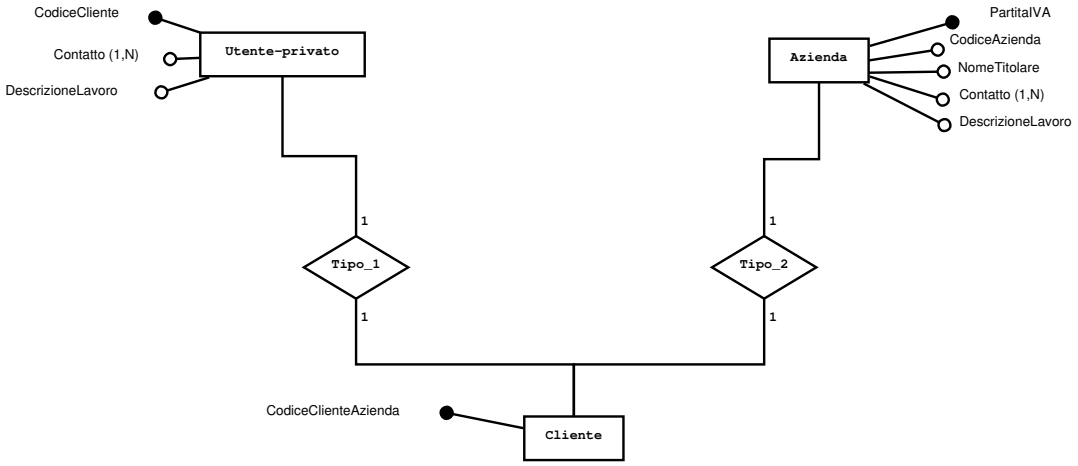

Figura 3.1: Rimozione della generalizzazione Cliente

identifica l'entità Prodotto è il CodiceProdotto con cui si identificavano le entità figlie.

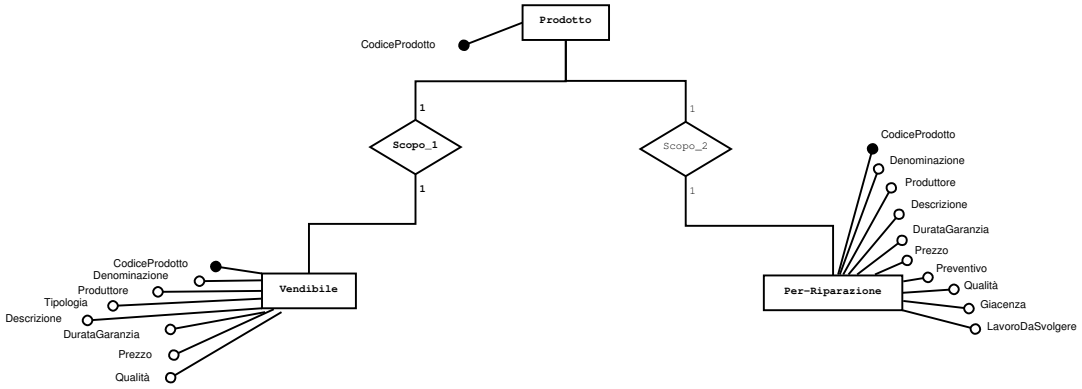

Figura 3.2: Rimozione della generalizzazione Prodotto

3.3 Attributi composti o multivalore

Si elencano qui sotto gli attributi composti o multivalore che devono essere adatti per la traduzione nello schema relazionale (tra parentesi si riporta l'entità di cui faceva parte):

- Indirizzo (Fornitore):Attributo composto;
- Contatti (Fornitore):Attributo multivalore;
- Tipologia (ProdottoVendibile): Attributo multivalore;

- Contatti (Utente-Privato): Attributo multivalore;
- Contatti (Azienda): Attributo multivalore;

3.3.1 Risoluzione attributo *Indirizzo*

Per quanto concerne l'attributo composto *Indirizzo*, la soluzione che si è deciso di prendere consiste nel dividere tale attributo nei quattro attributi di cui è composto, ovvero:

- Via;
- Numero civico;
- CAP;
- Città;

Attributi che saranno accorpati all'entità *Fornitore*.

3.3.2 Risoluzione attributo *Contatti*

Per i tre attributi multivalore *Contatti* presenti nello schema si è scelto di ridurre il numero di attributi che risultano utili a tre attributi, ovvero:

- Numero di Cellulare;
- Numero Fisso;
- E-mail;

Quindi nelle tre entità in cui è presente tale attributo si è scelto di sostituirlo con i tre attributi sopra elencati.

3.3.3 Risoluzione attributo *Tipologia*

L'attributo *Tipologia* è multivalore in quanto ogni prodotto potrebbe ricadere in diverse tipologie (ad esempio un hard-disk fa parte delle famiglie hardware, dischi di memoria e dischi con sistemi operativi), si sceglie quindi, onde evitare queste situazioni, di suddividere tale attributo in due soli attributi generici, i seguenti:

- Hardware;
- Software;

L'attributo sarà inoltre puramente descrittivo al fine di evitare possibili interferenze o anomalie.

3.4 Identificatori primari

Si nota dallo schema ER presente in figura 3.3, che ogni entità possiede un singolo identificatore, per cui da questo punto di vista, lo schema non richiede alcuna ristrutturazione.

In figura 3.3 si riporta lo schema ristrutturato.

3.5 Traduzione nel modello relazionale

Dopo l'analisi svolta nei capitoli precedenti, è ora possibile passare alla traduzione nel modello relazionale. Questo procedimento necessita di molta attenzione, infatti è necessario sviluppare un processo di mappatura che tenga conto di questi punti:

- Le entità;
- Le relazioni tra le entità;
- La cardinalità delle relazioni;

Si procede quindi considerando tutte le entità e le relazioni presenti, traducendole nell'equivalente schema logico.

3.5.1 Traduzione entità *Fornitore*

Tale entità non necessita di alcuna osservazione particolare,l'unica preoccupazione sussiste nel caso esista qualche vincolo di integrità referenziale verso un suo attributo; nel caso sorgesse una necessità di questo tipo (durante lo studio di entità o relazioni successive) si interverrà in un secondo momento. Nel frattempo si può quindi tradurre direttamente nel modo indicato in tabella 3.1.

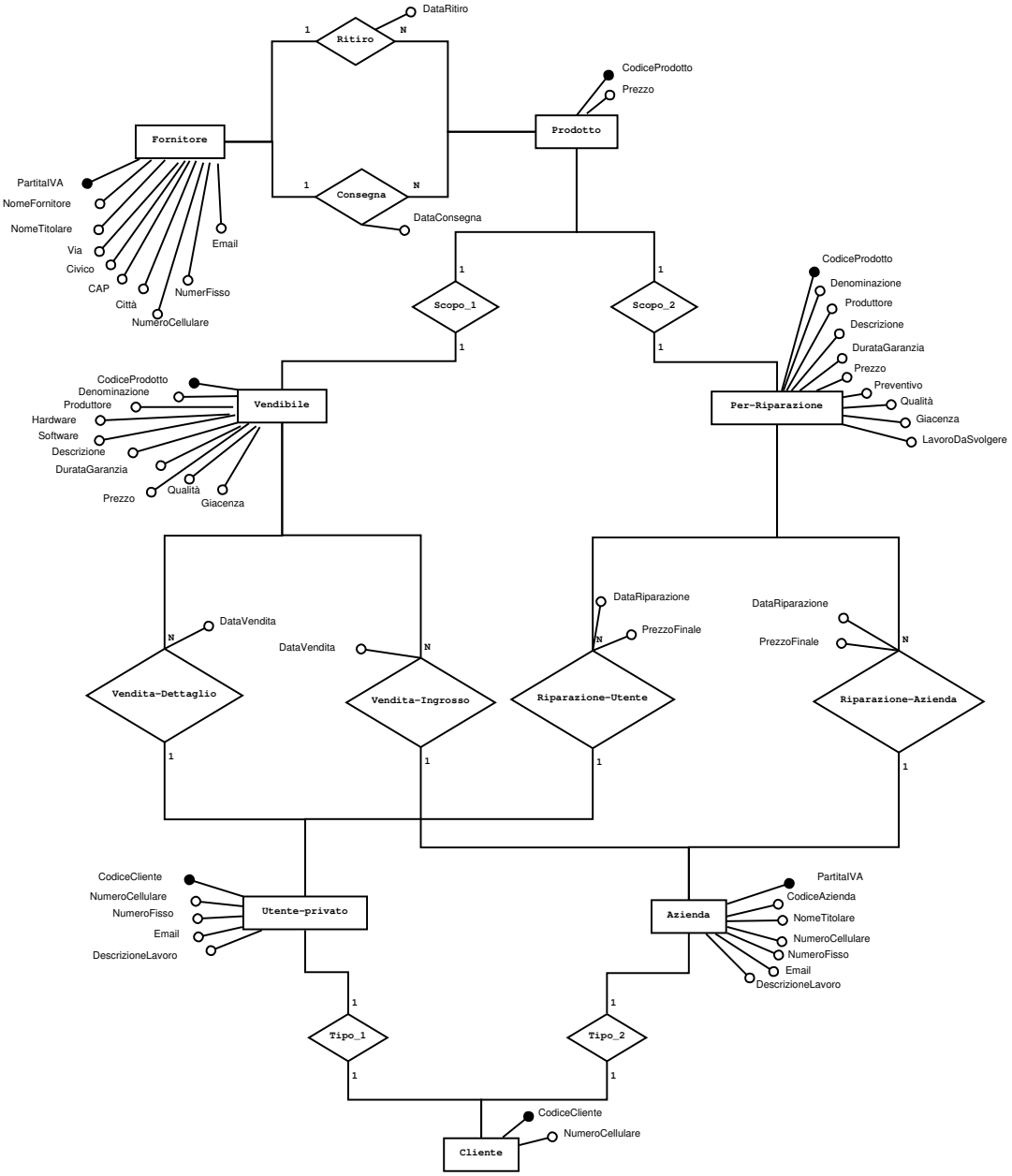

Figura 3.3: Schema ER ristrutturato

Tabella 3.1: Fornitore

PartitaIVA	NomeFornitore	NomeTitolare	Via	Civico	CAP	Città	NumeroCellulare	NumeroFisso	Email

3.5.2 Traduzione entità *Prodotto*

Per quanto concerne tale entità di è scelto di considerare come suoi attributi i seguenti:

- CodiceProdotto (che è pure chiave primaria)
- Prezzo.

Per cui la traduzione nell'equivalente modello relazionale è quella presente nella tabella 3.2.

Tabella 3.2: Prodotto

CodiceProdotto	Prezzo

3.5.3 Traduzione relazioni *Consegna* e *Ritiro*

Nel tradurre relazioni *Consegna* e *Ritiro* nello schema logico corrispondente, è utile notare che entrambe le relazioni posseggono un attributo, inoltre sono del tipo Uno a molti, per cui la forma equivalente in cui mappiamo queste due relazioni è presente rispettivamente nella tabella 3.3 e nella tabella 3.4.

Tabella 3.3: Consegna

PartitaIVA	CodiceProdotto	DataConsegna

Tabella 3.4: Ritiro

PartitaIVA	CodiceProdotto	DataRitiro

Per come sono costruite le due relazioni, è chiaro, che per quanto riguarda gli attributi *PartitaIVA* e *CodiceProdotto*, esiste un vincolo di integrità referenziale tra questi due attributi nelle relazioni *Consegna* e *Ritiro* e i corrispondenti attributi in *Fornitore* e *Attributo*.

3.5.4 Traduzione entità *Vendibile*

L'entità considerata in questo paragrafo non necessita di alcuna particolare osservazione, è sufficente mappare l'entità nello schema logico in modo naturale, il risultato è presente nella tabella 3.5.

Tabella 3.5: Vendibile

CodiceProdotto	Denominazione	Produttore	Hardware	Software	Descrizione	DurataGaranzia	Prezzo	Qualità	Giacenza

3.5.5 Traduzione entità *Per-Riparazione*

Per quanto concerne l'entità *Per-Riparazione* si procede come per l'entità Vendibile analizzata nel paragrafo precedente, ovvero è possibile mappare direttamente l'entità nella tabella relazionale corrispondente, il risultato di tale processo è evidenziato in tabella 3.6.

Tabella 3.6: Per-Riparazione

CodiceProdotto	Denominazione	Produttore	Descrizione	DurataGaranzia	Prezzo	Preventivo	Qualità	Giacenza	LavoroDaSvolgere

3.5.6 Traduzione entità *Cliente*

Nel corso dei capitoli precedente è stato appurato che l'entità cliente ha come attributi i seguenti:

- CodiceCliente / CodiceAzienda
- NumeroCellulare

In cui la chiave primaria è l'attributo CodiceCliente essendo esso univoco. La mappatura di tale entità si trova nella tabella 3.7.

Tabella 3.7: Cliente

CodiceCliente	NumeroCellulare

L'attributo *CodiceCliente* possiede un vincolo di integrità referenziale tra l'entità *Cliente* e le entità *Utente-Privato* e *Azienda*. L'integrità con quest'ultima è riferito all'attributo *CodiceAzienda*. Nello stesso modo l'attributo *NumeroCellulare* dell'entità *Cliente* ha un vincolo di integrità referenziale rispetto allo stesso attributo presente però nelle altre due entità.

3.5.7 Traduzione entità *Utente-Privato* e *Azienda*

Le due entità citate nel titolo, possono essere direttamente mappate nell'equivalente schema logico, nella tabella 3.8 è presente l'entità *Utente-Privato*, nella tabella 3.9 invece l'entità *Azienda*.

Tabella 3.8: Utente-Privato

CodiceCliente	NumeroCellulare	NumeroFisso	Email	DescrizioneLavoro

Tabella 3.9: Azienda

PartitaIVA	CodiceAzienda	NomeTitolare	NumeroCellulare	NumeroFisso	Email	DescrizioneLavoro

3.5.8 Traduzione relazioni *Vendita-Dettaglio* e *Vendita-Ingrosso*

Si affronta ora la traduzione nello schema logico delle due relazione sopra citate, esse sono del tipo *Uno a molti* e poiché possiedono già un attributo, si sceglie di dedicare loro una tabella ciascuno. Per quanto riguarda la relazione denominata *Vendita-Dettaglio*, essa possiede già l'attributo Data-Vendita che però non può essere considerato chiave primaria, che sarà altresì formata dai due attributi ereditati da *Cliente* e *Vendibile* ovvero CodiceProdotto e CodiceCliente, inoltre sarà necessario tenere memoria anche del prezzo del prodotto, in tabella 3.10 è mostrato lo schema logico corrispondente.

Tabella 3.10: Vendita-Dettaglio

CodiceProdotto	CodiceCliente	DataVendita	Prezzo

Esiste quindi un vincolo di integrità referenziale tra gli attributi di *Vendita-Dettaglio* e gli attributi omonimi presenti in *Cliente* e *Vendibile*. Per quanto concerne invece la relazione *Vendita-Ingrosso* il modus operandi è simmetrico, ovvero si scelgono come attributi identificatori PartitaIVA e CodiceProdotto ereditati dalle entità *Azienda* e *Vendibile* rispettivamente, cui aggiungere l'attributo Prezzo; anche in questo caso deve esistere per tali attributi il vincolo di integrità referenziale. In tabella 3.11 è riportata la traduzione nello schema logico.

Tabella 3.11: Vendita-Ingrosso

CodiceProdotto	PartitaIVA	DataVendita	Prezzo

3.5.9 Traduzione relazioni *Riparazione-Utente* e *Riparazione-Azienda*

Le ultime due relazioni da tradurre nello schema logico, riguardano la parte di assistenza tecnica che l'azienda offre, si sceglie di creare delle tabelle apposite. Per quanto riguarda la relazione *Vendita-Dettaglio*, si è deciso di aggiungere ai due attributi esistenti nella relazione (ovvero DataRiparazione e PrezzoFinale), gli attributi CodiceProdotto e CodiceCliente ereditati rispettivamente dalle entità *Per-Riparazione* e *Cliente* cui aggiungere l'attributo LavoroDaSvolgere per chiarire i problemi che dovessero esser stati risolti. Per gli attributi ereditati è necessario un vincolo di integrità referenziale rispetto agli attributi omonimi delle entità da cui sono stati presi. In tabella 3.12 è presente la traduzione di tale relazione.

Tabella 3.12: Riparazione-Utente

CodiceProdotto	CodiceCliente	DataRiparazione	PrezzoFinale	LavoroDaSvolgere

Infine la relazione *Riparazione-Azienda*, viene tradotta in modo similare; infatti oltre ai due attributi che già possiede si sceglie di aggiungere gli attributi PartitaIVA e CodiceProdotto proveniente da *Azienda* e da *Per-Riparazione* che sono la chiave primaria, e in aggiunta l'attributo LavoroDaSvolgere, per tali attributi deve esistere il vincolo di integrità referenziale. In tabella 3.13 è presente lo schema logico equivalente.

Tabella 3.13: Riparazione-Azienda

CodiceProdotto	PartitaIVA	DataRiparazione	PrezzoFinale	LavoroDaSvolgere

3.6 Normalizzazione dello schema logico

Si procede ora alla normalizzazione dello schema logico, è importante notare che tutte le tabelle si trovano già in **Prima forma normale**, in quanto durante la

ristrutturazione dello schema logico, tutti gli attributi sono stati scissi in attributi atomici; infatti la definizione di Prima forma normale è la seguente:

Si dice che una base dati è in 1NF (prima forma normale) se vale la seguente relazione per ogni relazione contenuta nella base dati; una relazione è in 1NF se e solo se:

- *non presenta gruppi di attributi che si ripetono (ossia ciascun attributo è definito su un dominio con valori atomici);*
- *esiste una chiave primaria (ossia esiste un insieme di attributi, che identifica in modo univoco ogni tupla della relazione) (cfr. [7])*

Per poter affrontare meglio lo studio della normalizzazione, è utile evidenziare anche le definizioni delle altre forme normali.

Seconda forma normale: *Una base dati è invece in 2NF (seconda forma normale) quando è in 1NF e per ogni tabella tutti i campi non chiave dipendono funzionalmente dall'intera chiave composta e non da una parte di essa. (Cfr [8])*

Terza forma normale: *Una base dati è in 3NF (terza forma normale) se è in 2NF e per ogni dipendenza funzionale è vera una delle seguenti condizioni: X è una superchiave della relazione Y è membro di una chiave della relazione. (Cfr [9])*

Forma normale di Boyce e Codd: *Una relazione R è in forma normale di Boyce e Codd (BCNF) se e solo se è in 3NF e, per ogni dipendenza funzionale non banale , X è una superchiave per R. (Cfr [6]).*

Dopo aver evidenziato tali definizioni è ora possibile passare alla normalizzazione dello schema, per non appesantire la lettura, si è scelto di riportare solo il riferimento a ogni tabella (senza riportarla integralmente) in modo tale che il testo risulti maggiormente scorrevole.

3.6.1 Normalizzazione di Fornitore

In tabella 3.1 è presente la tabella della relazione *Fornitore*, come è già stato notato, essa è già in prima forma normale, viene quindi analizzato per le altre forme normali:

- 2FN: verificata, essendo la chiave della relazione formata da un solo attributo non possono esserci dipendenze di tipo parziale;

- 3FN: verificata, perchè si trova in 2FN e non ci sono dipendenze di tipo transitivo;
- BCFN: verificata perchè tutti gli attributi dipendono dalla sola chiave primaria.

3.6.2 Normalizzazione di Prodotto

Nella tabella 3.2 si trova la relazione *Prodotto*, si analizza ore come essa si comporta rispetto alle varie forme normali.

- 2FN: verificata, essendo la chiave della relazione formata da un solo attributo non possono esserci dipendenze di tipo parziale;
- 3FN: verificata, perché si trova in 2FN e non ci sono dipendenze di tipo transitivo;
- BCFN: verificata perché tutti gli attributi dipendono dalla sola chiave primaria.

3.6.3 Normalizzazione di Consegna e Ritiro

Le due relazioni *Consegna* e *Ritiro* si possono consultare presso le tabelle 3.3 e 3.4, di seguito si analizza il comportamento rispetto alle altre forme normali.

- 2FN: verificata, essendo la chiave della relazione formata da un solo attributo non possono esserci dipendenze di tipo parziale;
- 3FN: verificata, perché si trova in 2FN e non ci sono dipendenze di tipo transitivo;
- BCFN: verificata perché tutti gli attributi dipendono dalla sola chiave primaria.

3.6.4 Normalizzazione di Vendibile

La relazione *Vendibile* è riportata in tabella 3.5, si analizza ora il suo comportamento rispetto alle varie forme normali.

- 2FN: verificata, essendo la chiave della relazione formata da un solo attributo non possono esserci dipendenze di tipo parziale;
- 3FN: verificata, perché si trova in 2FN e non ci sono dipendenze di tipo transitivo;
- BCFN: verificata perché tutti gli attributi dipendono dalla sola chiave primaria.

3.6.5 Normalizzazione di Per-Riparazione

La relazione *Per-Riparazione* è riportata in tabella 3.6, di seguito si riporta lo studio delle varie forme normali.

- 2FN: verificata, essendo la chiave della relazione formata da un solo attributo non possono esserci dipendenze di tipo parziale;
- 3FN: verificata, perché si trova in 2FN e non ci sono dipendenze di tipo transitivo;
- BCFN: verificata perché tutti gli attributi dipendono dalla sola chiave primaria.

3.6.6 Normalizzazione di Cliente

La relazione *Cliente*, presente in tabella 3.7, viene analizzata rispetto alle forme normali qui di seguito:

- 2FN: verificata, essendo la chiave della relazione formata da un solo attributo non possono esserci dipendenze di tipo parziale;
- 3FN: verificata, perché si trova in 2FN e non ci sono dipendenze di tipo transitivo;
- BCFN: verificata perché tutti gli attributi dipendono dalla sola chiave primaria.

3.6.7 Normalizzazione di Utente-Privato e Azienda

La relazione *Utente-Privato* riportata nella tabella 3.8 ha i seguenti comportamenti rispetto alle altre forme normali.

- 2FN: verificata, essendo la chiave della relazione formata da un solo attributo non possono esserci dipendenze di tipo parziale;
- 3FN: verificata, perché si trova in 2FN e non ci sono dipendenze di tipo transitivo;
- BCFN: verificata perché tutti gli attributi dipendono dalla sola chiave primaria.

Allo stesso modo, la relazione *Azienda* (consultabile in tabella 3.9), rispetto alle varie forme normali, si comporta così:

- 2FN: verificata, essendo la chiave della relazione formata da un solo attributo non possono esserci dipendenze di tipo parziale;
- 3FN: verificata, perché si trova in 2FN e non ci sono dipendenze di tipo transitivo;
- BCFN: verificata perché tutti gli attributi dipendono dalla sola chiave primaria.

3.6.8 Normalizzazione di Vendita-Dettaglio e Vendita-Ingrosso

Le relazioni *Vendita-Dettaglio* e *Vendita-Ingrosso* sono presenti in tabella 3.10 e 3.11 e rispetto alle forme normali diverse dalla prima si comportano nel seguente modo:

- 2FN: verificata, essendo la chiave della relazione formata da un solo attributo non possono esserci dipendenze di tipo parziale;
- 3FN: verificata, perché si trova in 2FN e non ci sono dipendenze di tipo transitivo;
- BCFN: verificata perché tutti gli attributi dipendono dalla sola chiave primaria.

Potrebbe sembrare che la seconda forma normale non sia soddisfatta in quanto l'attributo 'Prezzo' potrebbe dipendere parzialmente dall'attributo 'Codice-Prodotto', in realtà, le aziende o i cliente più abituali possono ricevere forti sconti, ecco perchè l'attributo 'Prezzo' non dipende solo dallo specifico prodotto (anche se nella maggior parte dei casi sarà così) ma anche dal cliente che lo vuole comprare.

3.6.9 Normalizzazione di Riparazione-Utente e Riparazione-Azienda

Le due relazioni *Riparazione-Utente* e *Riparazione-Azienda* sono presenti nelle tabelle 3.12 e 3.13, di seguito si riporta l'analisi rispetto alle altre forme normali:

- 2FN: verificata, essendo la chiave della relazione formata da un solo attributo non possono esserci dipendenze di tipo parziale;
- 3FN: verificata, perché si trova in 2FN e non ci sono dipendenze di tipo transitivo;
- BCFN: verificata perché tutti gli attributi dipendono dalla sola chiave primaria.

Capitolo 4

Progettazione fisica

In questo capitolo si affronta la realizzazione delle tabelle in linguaggio SQL, questo poichè i tre database che si andranno ad analizzare supportano tale linguaggio; di seguito sono riportate le implementazioni delle tabelle

4.1 Implementazione database

Prima di tutto è necessario creare il database con il comando presente nel frammento di codice 4.1:

Listing 4.1: Creazione database

```
CREATE DATABASE DSINFORMATICADB;
```

Dopo questo primo passo, è possibile passare a definire ogni singola tabella, nel codice 4.2 è riportata la creazione della tabella riferita alla relazione *Fornitore*.

Listing 4.2: Tabella Fornitore

```
CREATE TABLE FORNITORE(
  "PartitaIVA" character(15) NOT NULL,
  "NomeFornitore" character varying(30) NOT NULL,
  "NomeTitolare" character varying(30) NOT NULL,
  "Via" character varying(30) NOT NULL,
  "Civico" numeric(4,0) NOT NULL,
  "CAP" numeric(5,0) NOT NULL,
```

```
  "CittÃ " character varying(20),
  "NumeroCellulare" character(14) NOT NULL,
  "NumeroFisso" character(14),
  "Email" character varying(30),
  CONSTRAINT "Fornitore_pkey" PRIMARY KEY ("PartitaIVA")
)
```

Nel codice 4.3 si riporta l'implementazione della relazione *Prodotto* in linguaggio SQL.

Listing 4.3: Tabella Prodotto

```
CREATE TABLE PRODOTTO(
  "CodiceProdotto" character(15) NOT NULL,
  "Prezzo" numeric(6,0) NOT NULL,
  CONSTRAINT "Prodotto_pkey" PRIMARY KEY ("CodiceProdotto")
)
```

A questo punto si prosegue con l'implementazione della tabella *Vendibile*, evidenziata nel codice 4.4.

Listing 4.4: Tabella Vendibile

```
CREATE TABLE VENDIBILE(
  "CodiceProdotto" character(15) NOT NULL,
  "Denominazione" character varying(30),
  "Produttore" character varying(30) NOT NULL,
  "Hardware" character varying(20),
  "Software" character varying(20),
  "Descrizione" character varying(20),
  "DurataGaranzia" numeric(3,0) NOT NULL,
  "Prezzo" numeric(6,0) NOT NULL,
  "QualitÃ " smallint,
  "Giacenza" integer NOT NULL,
  CONSTRAINT "Vendibile_pkey" PRIMARY KEY ("CodiceProdotto"),
  CONSTRAINT "Vendibile_CodiceProdotto_fkey" FOREIGN KEY ("CodiceProdotto")
      REFERENCES "Prodotto" ("CodiceProdotto") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
)
```

Il prossimo passo riguarda la relazione *Per-Riparazione*, la cui implementatazione è mostrata nel frammento di codice 4.5

Listing 4.5: Tabella Per-Riparazione

```
CREATE TABLE PER-RIPARAZIONE(
  "CodiceProdotto" character(15) NOT NULL,
  "Denominazione" character varying(30),
  "Produttore" character varying(30) NOT NULL,
  "Descrizione" character varying(20),
  "DurataGaranzia" numeric(3,0) NOT NULL,
  "Prezzo" numeric(6,0) NOT NULL,
  "Preventivo" numeric(6,0) NOT NULL,
  "QualitÃ " smallint,
  "Giacenza" integer NOT NULL,
  "LavoroDaSvolgere" character varying(100),
  CONSTRAINT "Per-Riparazione_pkey" PRIMARY KEY ("CodiceProdotto"),
  CONSTRAINT "Per-Riparazione_CodiceProdotto_fkey" FOREIGN KEY
("CodiceProdotto")
      REFERENCES "Prodotto" ("CodiceProdotto") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE

)
```

Nel frammento di codice 4.6 è riportata invece l'implementazione della relazione *Consegna*.

Listing 4.6: Tabella Consegna

```
CREATE TABLE CONSEGNA(
  "PartitaIVA" character(15) NOT NULL,
  "CodiceProdotto" character(15) NOT NULL,
  "DataConsegna" date NOT NULL,
  CONSTRAINT "Consegna_pkey" PRIMARY KEY ("PartitaIVA", "CodiceProdotto"),
  CONSTRAINT "Consegna_CodiceProdotto_fkey" FOREIGN KEY ("CodiceProdotto")
      REFERENCES "Prodotto" ("CodiceProdotto") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT "Consegna_PartitaIVA_fkey" FOREIGN KEY ("PartitaIVA")
      REFERENCES "Fornitore" ("PartitaIVA") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE

)
```

Simmetricamente si riporta l'implementazione di *Ritiro* nel codice 4.7

Listing 4.7: Tabella Ritiro

```
CREATE TABLE RITIRO(
  "PartitaIVA" character(15) NOT NULL,
  "CodiceProdotto" character(15) NOT NULL,
  "DataRitiro" date NOT NULL,
  CONSTRAINT "Ritiro_pkey" PRIMARY KEY ("PartitaIVA", "CodiceProdotto"),
  CONSTRAINT "Ritiro_CodiceProdotto_fkey" FOREIGN KEY ("CodiceProdotto")
      REFERENCES "Prodotto" ("CodiceProdotto") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT "Ritiro_PartitaIVA_fkey" FOREIGN KEY ("PartitaIVA")
      REFERENCES "Fornitore" ("PartitaIVA") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
)
```

Nei frammenti di codice 4.8, 4.9 e 4.10 ,è possibile vedere l'implementazione delle relazioni *Cliente*, *Utente-Privato* e *Azienda*.

Listing 4.8: Tabella Cliente

```
CREATE TABLE CLIENTE(
  "CodiceCliente" character(15) NOT NULL,
  "NumeroCellulare" character(14) NOT NULL,
  CONSTRAINT "Cliente_pkey" PRIMARY KEY ("CodiceCliente"),
  CONSTRAINT "Cliente_NumeroCellulare_key" UNIQUE ("NumeroCellulare")

)
```

Listing 4.9: Tabella Utente

```
CREATE TABLE UTENTE-PRIVATO(
  "CodiceCliente" character(15) NOT NULL,
  "NumeroCellulare" character(14),
  "NumeroFisso" character(14),
  "Email" character varying(50),
  "DescrizioneLavoro" character varying(100),
  CONSTRAINT "Utente-Privato_pkey" PRIMARY KEY ("CodiceCliente"),
  CONSTRAINT "Utente-Privato_CodiceCliente_fkey" FOREIGN KEY ("CodiceCliente")
      REFERENCES "Cliente" ("CodiceCliente") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
```

```
  CONSTRAINT "Utente-Privato_NumeroCellulare_fkey" FOREIGN KEY
("NumeroCellulare")
      REFERENCES "Cliente" ("NumeroCellulare") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
)
```

Listing 4.10: Tabella Azienda

```
CREATE TABLE AZIENDA(
  "PartitaIVA" character(15) NOT NULL,
  "CodiceAzienda" character(15),
  "NumeroCellulare" character(14),
  "NumeroFisso" character(14) NOT NULL,
  "Email" character varying(50),
  "DescrizioneLavoro" character varying(100),
  CONSTRAINT "Azienda_pkey" PRIMARY KEY ("PartitaIVA"),
  CONSTRAINT "Azienda_CodiceAzienda_fkey" FOREIGN KEY ("CodiceAzienda")
      REFERENCES "Cliente" ("CodiceCliente") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
)
```

Nei frammenti di codice 4.11 e 4.12 sono invece implementate le relazioni riguardanti la vendita, ovvero *Vendita-Dettaglio* e *Vendita-Ingrosso*.

Listing 4.11: Tabella Vendita-Dettaglio

```
CREATE TABLE VENDITA-DETTAGLIO(
  "CodiceProdotto" character(15) NOT NULL,
  "CodiceCliente" character(15) NOT NULL,
  "DataVendita" date NOT NULL,
  "PrezzoFinale" numeric(6,0) NOT NULL,
  CONSTRAINT "Vendita-Dettaglio_pkey" PRIMARY KEY ("CodiceProdotto",
"CodiceCliente"),
  CONSTRAINT "Vendita-Dettaglio_CodiceCliente_fkey" FOREIGN KEY
("CodiceCliente")
      REFERENCES "Utente-Privato" ("CodiceCliente") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT "Vendita-Dettaglio_CodiceProdotto_fkey" FOREIGN KEY
("CodiceProdotto")
      REFERENCES "Vendibile" ("CodiceProdotto") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
)
```

Listing 4.12: Tabella Vendita-Ingrosso

```
CREATE TABLE VENDITA-INGROSSO(
  "CodiceProdotto" character(15) NOT NULL,
  "PartitaIVA" character(15) NOT NULL,
  "DataVendita" date NOT NULL,
  "PrezzoFinale" numeric(6,0) NOT NULL,
  CONSTRAINT "Vendita-Ingrosso_pkey" PRIMARY KEY ("CodiceProdotto",
"PartitaIVA"),
  CONSTRAINT "Vendita-Ingrosso_CodiceProdotto_fkey" FOREIGN KEY
("CodiceProdotto")
      REFERENCES "Vendibile" ("CodiceProdotto") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT "Vendita-Ingrosso_PartitaIVA_fkey" FOREIGN KEY ("PartitaIVA")
      REFERENCES "Azienda" ("PartitaIVA") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
)
```

Infine si riportano le relazioni *Riparazione-Utente* e *Riparazione-Azienda*, implementate nei codici 4.13 e 4.14.

Listing 4.13: Tabella Riparazione-Utente

```
CREATE TABLE RIPARAZIONE-UTENTE(
  "CodiceProdotto" character(15) NOT NULL,
  "CodiceCliente" character(15) NOT NULL,
  "DataRiparazione" date NOT NULL,
  "PrezzoFinale" numeric(6,0) NOT NULL,
  "LavoroSvolto" character varying(100),
  CONSTRAINT "Riparazione-Utente_pkey" PRIMARY KEY ("CodiceProdotto",
"CodiceCliente", "DataRiparazione"),
  CONSTRAINT "Riparazione-Utente_CodiceCliente_fkey" FOREIGN KEY
("CodiceCliente")
      REFERENCES "Utente-Privato" ("CodiceCliente") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT "Riparazione-Utente_CodiceProdotto_fkey" FOREIGN KEY
("CodiceProdotto")
      REFERENCES "Per-Riparazione" ("CodiceProdotto") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
)
```

Listing 4.14: Tabella Riparazione-Azienda

```
CREATE TABLE RIPARAZIONE-AZIENDA(
```

```
  "CodiceProdotto" character(15) NOT NULL,
  "PartitaIVA" character(15) NOT NULL,
  "DataRiparazione" date NOT NULL,
  "PrezzoFinale" numeric(6,0) NOT NULL,
  "LavoroSvolto" character varying(100),
  CONSTRAINT "Riparazione-Azienda_pkey" PRIMARY KEY ("CodiceProdotto",
"PartitaIVA", "DataRiparazione"),
  CONSTRAINT "Riparazione-Azienda_CodiceProdotto_fkey" FOREIGN KEY
("CodiceProdotto")
      REFERENCES "Per-Riparazione" ("CodiceProdotto") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT "Riparazione-Azienda_PartitaIVA_fkey" FOREIGN KEY ("PartitaIVA")
      REFERENCES "Azienda" ("PartitaIVA") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
)
```

4.2 Aggiornamento e Interrogazioni

Si mostreranno ora alcuni esempi di operazioni di aggiornamento e interrogazioni sul database (in linguaggio SQL).

Nel codice 4.15 sono mostrati due esempi diversi dell'operazione **INSERT**, in un primo caso è mostrato il comando applicato a tutti gli attributi, in un secondo caso invece l'aggiornamento è applicato ad un sottoinsieme di attributi.

Listing 4.15: Esempio di INSERT

```
INSERT INTO AZIENDA
VALUES ('AAAAAAAAAAAAAAA','123456789098765',
     '11122233344455','00112233445555','azienda1@mail.it','Acquisto␣laptop');

INSERT INTO GIACENZA(CodiceProdotto,Produttore,DurataGaranzia,Prezzo,Giacenza)
VALUES ('123321456654789','ProducedBy', '24', '120','30');
```

Nel codice 4.16 sono riportati due esempi distinti dell'operazione **DELETE** applicate al database.

Listing 4.16: Esempio di DELETE

```
DELETE FROM RIPARAZIONE-UTENTE
```

```
WHERE DataRiparazione='2010-03-21';

DELETE FROM VENDITA-DETTAGLIO
WHERE CodiceCliente='111111122222333'
```

Si riportano invece nei frammenti di codice 4.17 esempi di interrogazioni del database.

Listing 4.17: Esempio di Interrogazioni

```
SELECT DescrizioneLavoro
FROM UTENTE-PRIVATO
WHERE CodiceCliente='111111111122222';

SELECT LavoroSvolto
FROM RIPARAZIONE-AZIENDA
WHERE PrezzoFinale > '250';

SELECT CodiceProdotto
FROM CONSEGNA C, VENDIBILE V
WHERE V.Prezzo < '100' AND C.DataConsegna = '2009/04/25';

UPDATE CLIENTE
SET NumeroCellulare = '00393333334444'
WHERE CodiceCliente = '999998888877777';
```

Capitolo 5

Conclusioni

Nei capitoli precedenti è stata affrontata la progettazione concettuale, logica e fisica del database; scrivendo le tabelle in linguaggio SQL. Terminato questo approccio si è deciso di implementare tale lavoro con tre diversi database opensource, in particolare con

- *MySQL*
- *Postgres attraverso l'interfaccia pgAdmin III*
- *HSQLDB attraverso l'interfaccia OpenOffice-Database*

Poichè la mole più consistente del lavoro ha riguardato la progettazione, si è scelto di non riportare i particolari di tali implementazioni in quanto avrebbero appesantito la lettura del lavoro, è comunque possibile visionare i database.

Si analizza ora per ogni database pro e contro dell'implementazione.

5.1 MySQL

L'implementazione con MySQL si è rivelata semplice, in quanto esiste in rete un'ampia documentazione a supporto, resta però un problema abbastanza sostanziale, infatti l'implementazione mediante terminale è utile (soprattutto da un punto di vista didattico) ma macchinosa e lenta, è quindi necessario l'uso di tool esterne non sempre compatibili tra loro. Inoltre sebbene MySQL sia molto veloce in database locali, in rete perde velocità rispetto a molti altri database. Infine si presenta un altro problema, da quando MySQL è sta acquistata da Sun (ovvero

da Oracle), si ha l'impressione che l'edizione community abbia sempre meno importanza a favore dell'edizione enterprise, da questo fatto derivano una serie di fork di MySQL che prendono strade distinte; ragion per cui l'edizione opensource community è poco seguita e aggiornata. Questo aspetto rappresenta un punto a sfavore di MySQL, infatti si suppone che il sistema informativo sia utilizzabile per un periodo di qualche anno, quindi la scelta di MySQL risulterebbe sbagliata poiché non si avrebbe la certezza di avere un supporto in termini di aggiornamento e assistenza nel lungo periodo.

5.2 Postgres

Per l'implementazione in *Postgres* ci si è avvalsi dell'aiuto dell'interfaccia grafica pgAdmin III, interfaccia che si è rivelata molto utile per velocizzare le operazioni di inserimento, aggiornamento e query. Postgres presenta numerevoli vantaggi:

- Programmabilità: infatti è possibile manipolare il database orientandolo agli oggetti;
- Incremento delle prestazioni: in quanto la logica viene applicata direttamente dal server di database in una volta, riducendo il passaggio di informazioni tra il client ed il server;
- Incremento dell'affidabilità: dovuto alla centralizzazione del codice di controllo sul server, non dovendo gestire la sincronizzazione della logica tra molteplici client e i dati memorizzati sul server;
- Livelli di astrazione dei dati direttamente sul server: il codice del client può essere più snello e semplice.

(Cfr [4]).

Per quanto concerne la programmabilità, è interessante notare come questo aspetto permetta di arricchire il database in un secondo momento senza stravolgere il progetto originale.
Infine l'utilizzo di pgAdmin III, ha reso semplice sia la creazione delle tabelle che le interrogazioni di testing; inoltre realizzando il sistema informattivo con Postgres si ha la sicurezza di un'assistenza tecnica. Questo aspetto in particolare, unito alla velocità di esecuzione delle interrogazioni e aggiornamento, rende Postgres preferibile a MySQL.

5.3 HSQLDB mediante OpenOffice-Database

HSQLDB (HSQL Database Engine) è un gestore di database relazionale scritto completamente in Java. È molto leggero (circa 600 kB) e può essere utilizzato sia come server (al quale le applicazioni si collegano tramite il relativo driver JDBC), sia come istanza interna ad un'applicazione. I dati possono essere salvati su disco (permettendone il ripristino ad ogni avvio) o in memoria (come contenitore temporaneo di dati, i quali vengono perduti quando il server o l'applicazione vengono chiusi). Supporta le modalità embedded (incorporato) o server (Cfr [5]). Per l'implementazione ci si è avvalsi dell'interfaccia grafica offerta da *OpenOffice Database*. Il sistema informativo non è risultato molto performante, infatti a causa dell'interfaccia è piuttosto lento e macchinoso, nonostante la semplicità d'uso dell'interfaccia stessa; in secondo luogo i vincoli di chiave esterna sono risultati difficilmente costruibili poiché OpenOffice Database richiede la costruzione di tale vincolo in veste grafica, il risultato è una difficoltà oggettiva a collegare le chiavi soprattutto nel caso di un numero elevato di tabelle.

Date le analisi precedentemente riportare, è chiaro quindi che il sistema informativo realizzato attraverso il database implementato mediante *Postgres*, risulta essere il più adatto alla situazione cercata per i seguenti motivi:

- velocità;
- semplicità di utilizzo;
- programmabilità;
- supporto tecnico.

Ecco perché si è scelto di implementare il sistema informativo attraverso tale database.

Bibliografia

[1] Diesse informatica, prodotti in vendita. http://www.diesseweb.it/e-commerce.html.

[2] Diesse informatica s.r.l. http://www.diesseweb.it/home/profilo-aziendale.html.

[3] Wikipedia. http://it.wikipedia.org/wiki/DBMS.

[4] Wikipedia. http://it.wikipedia.org/wiki/PostgreSQL.

[5] Wikipedia. http://it.wikipedia.org/wiki/HSQLDB.

[6] Wikipedia, forma normale di boyce e codd. http://it.wikipedia.org/wiki/Normalizzazione_(informatica).

[7] Wikipedia, prima forma normale. http://it.wikipedia.org/wiki/Normalizzazione_(informatica).

[8] Wikipedia, seconda forma normale. http://it.wikipedia.org/wiki/Normalizzazione_(informatica).

[9] Wikipedia, terza forma normale. http://it.wikipedia.org/wiki/Normalizzazione_(informatica).

[10] Ramez A. Elmasri and Shamkant B. Navathe. Modellazione Entità-Associazione estesa EER. In *Sistemi di basi di dati. Fondamenti*, chapter 4.

Elenco delle figure

Elenco delle tabelle