Università Politecnica delle Marche

Dipartimento di Ingegneria dell'Informazione

Facoltà di Ingegneria Informatica e dell'Automazione

Corso di New Generation Databases

Progettazione NoSQL di DB per la gestione della riparazione di dispositivi elettronici

Professoressa:
Prof. Claudia Diamantini

Studenti:
Bernovschi Denis
Incicco Emanuele

A.A 2021/2022

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

La relazione corrente tratterà la progettazione e l'implementazione di una base di dati mediante l'utilizzo di sistemi NoSQL. La tecnologia NoSQL, sviluppata a partire dagli anni 2000, ha avuto come padre fondatore il professore Carlo Strozzi che per la prima volta nel 1998 ha coniato questo termine. La tecnologia NoSQL, pone l'ottica su DBMS non relazionali, tra i vantaggi salienti di questa tecnologia non possiamo non citare: la semplicità di design, la scalabilità orizzontale più semplice per i cluster di macchine, con conseguente maggior controllo sulla disponibilità e la gestione della distribuzione delle diverse copie. Inoltre, un'ulteriore aspetto da non sottovalutare sono le performance, il cosiddetto "*time for operation*", fondamentale anche in ottica *Big Data*, in cui i dati oltre ad essere acquisiti a velocità elevata, devono anche essere processati in tempi brevi. Ecco quindi il motivo della grande adozione di sistemi NoSQL in ambito business, social, etc.
La progettazione e lo sviluppo di questo lavoro sono iniziati prendendo le specifiche da un progetto sviluppato durante il percorso triennale per il corso **Sistemi Informativi e Basi di Dati**. Il progetto originale prevedeva l'utilizzo di un modello relazionale per la gestione dei dati di un centro riparazione, "*CG Repair*". L'azienda in questione si occupa di effettuare riparazioni per vari tipi di *devices* (smartphone, tablet e computer). Quindi, alcuni elementi come lo schema Entity-Relationship, la tavola dei volumi, la tavola delle operazioni e la tavola delle frequenze sono presi direttamente da tale progetto. Questi elementi costituiscono quindi il punto di partenza per il nostro progetto qui presentato.
Lo scopo del nostro progetto è quello di progettare ed implementare un sistema informativo, utilizzando le tecnologie NoSQL, a supporto della società "*CG Repair*".
Dato che il contesto del sistema informativo di partenza nasce e veste perfettamente l'opzione di base di dati relazionale, l'adattamento qui presentato ad una base di dati NoSQL è svolta ai soli fini didattici.
Comunque, attraverso degli accorgimenti e delle modifiche del contesto iniziale, siamo riusciti a soddisfare i requisiti e gli obiettivi prefissati inizialmente.

Capitolo 2

Analisi dei requisiti e contesto di riferimento

All'interno di questo capitolo verrà trattato e definito il contesto di riferimento del progetto da sviluppare, ponendo particolare attenzione a quelli che sono i requisiti e le specifiche richieste. Lo sviluppo del sistema informativo richiede la massima conoscenza del contesto in cui esso si trova ad operare. Va posta particolare attenzione a quello che viene definito *l'information noise*, cioè tutto ciò che non è rilevante per il problema applicativo, questo ci permette di ridurre la mole di dati da processare, rispettando le *best practice* dei Big Data.
Il contesto che rappresentiamo è quello del centro riparazioni dell'azienda "*CG Repair*". All'interno della sede **l'operatore**, una volta incontrato **il cliente** e visionato il **device** di quest'ultimo, stipula un eventuale contratto di **riparazione**. Terminata la fase contrattuale, l'operatore prevede ad effettuare la riparazione utilizzando i **ricambi** precedentemente acquistati da vari **fornitori**. Nel contesto da noi previsto, il sistema verrà utilizzato dal generico operatore, senza alcuna distinzioni di livello.
Nell'analisi dei requisiti si vuole individuare i problemi che il sistema da realizzare deve risolvere e le caratteristiche che tale sistema dovrà possedere. Essendo questi fortemente dipendenti dall'applicazione da implementare, abbiamo scelto di rappresentarli suddividendoli in requisiti funzionali e non funzionali.
I **requisiti funzionali** che il nostro sistema dovrà possedere sono:

- Gestione efficiente ed ottimale delle attività di CRUD [1]
- Semplicità di utilizzo, rendendo così l'adozione di tale sistema quanto più *user-friendly* possibile per i vari operatori.

[1] CRUD $\models$ Create, Read, Update, Delete

Figura 2.1: Mongo DB

I **requisiti non funzionali** che reputiamo importanti sono:

- Il sistema da implementare dovrà sfruttare tecnologie *Open-Source*, in modo da garantire sia la sicurezza che avere un impatto economico minore;
- Il sistema informativo deve adottare tecniche e tecnologie NoSQL, a tal proposito utilizzeremo il DBMS MongoDB (Fig. 2.1) e la sua relativa GUI [2] Compass;
- Il linguaggio di programmazione per la fase di ETL [3] del *dataset* è Python.

[2]GUI ⊨ Graphic User Interface
[3]ETL ⊨ Extract, Transform e Load

Capitolo 3

Progettazione

La progettazione è la fase principale e critica dell'intero lavoro realizzato. Questa ci porterà infatti ad implementare il nostro sistema NoSQL.
La progettazione si concentrerà proprio su come organizziamo i dati nel database (e quindi possiede i medesimi obiettivi della progettazione di database relazionali). Nel diagramma presente nella figura (Fig 3.1), sono riepilogati i 5 passaggi fondamentali della progettazione di un database NoSQL.

Figura 3.1: Processo di progettazione dei Database *NoSQL*

3.1 Progettazione Concettuale

Seguendo il processo descritto nella figura (Fig. 3.1), il primo task è relativo alla progettazione concettuale. Lo schema *entity - relationship* utilizzato per tale lavoro è visibile nella figura (Fig. 3.2).

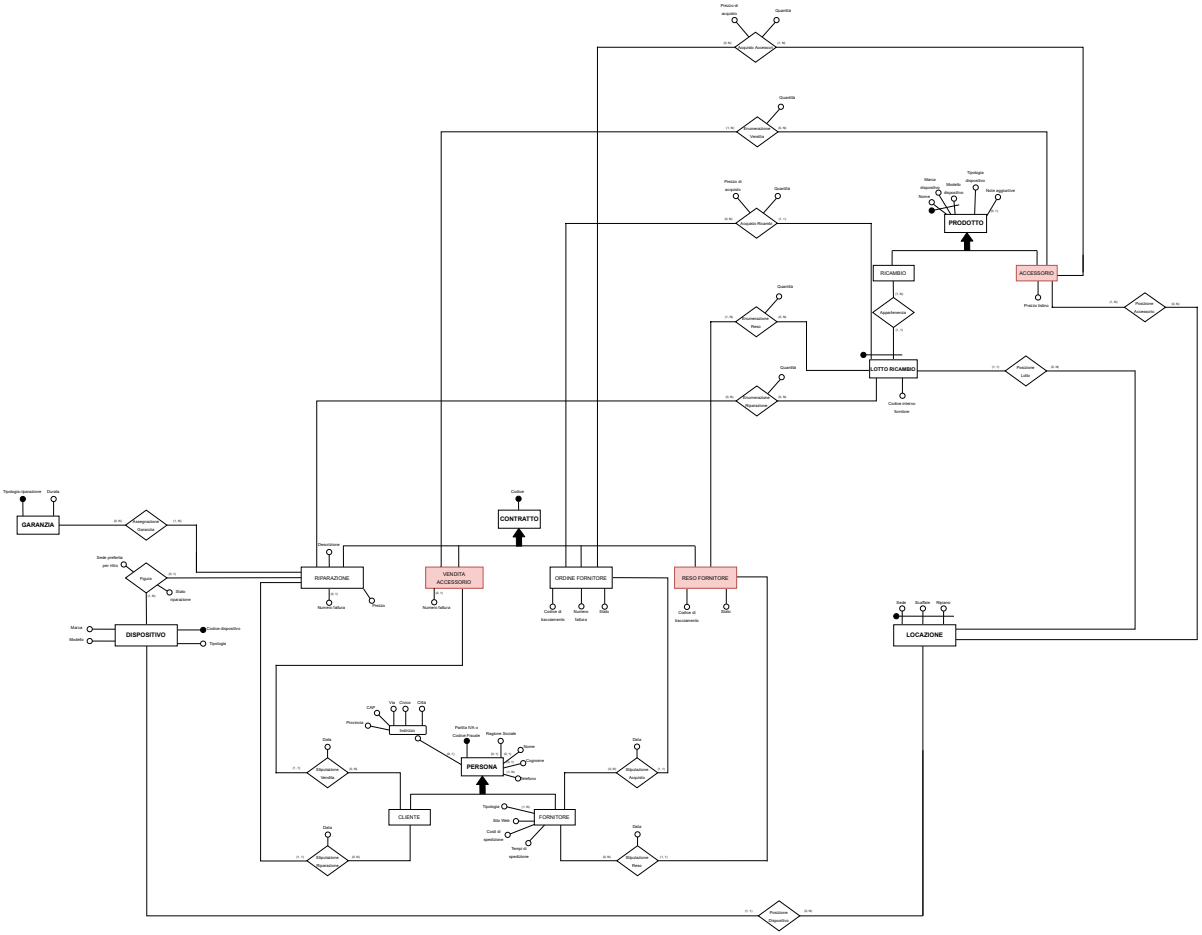

Figura 3.2: Entity Relationship Schema

Osservando la figura (Fig. 3.2) si può comprendere che l'entità principale, nonché core del nostro progetto, è l'entità **Riparazione**. Tale entità risulta infatti essere collegata alle altre entità fondamentali per lo sviluppo dell'elaborato come: Cliente, Dispositivo, Lotto Ricambio e, di conseguenza, Ricambio e Fornitore.

Dallo schema E-R di partenza abbiamo effettuato delle **semplificazioni**, questo ci ha permesso di ragionare più dettagliatamente sugli elementi centrali dello schema. Tali semplificazioni sono visibili nella figura (Fig. 3.3).

Come si può osservare nello schema semplificato si sono elise alcune entità, in particolare:

- L'entità Garanzia è stata rimossa in quanto secondaria;
- L'entità Accessorio e quanto ne consegue come vendita è stata rimossa per garantire una maggiore attenzione sullo sviluppo della parte core del progetto in esame;
- L'entità Reso e quanto ne consegue, è stata rimossa sempre in un'ottica di maggior semplificazione e attenzione alla parte centrale del progetto.

Tali semplificazioni sono state ritenute fondamentali per rimanere in linea con gli obiettivi di sviluppo di un sistema NoSQL prestabiliti.

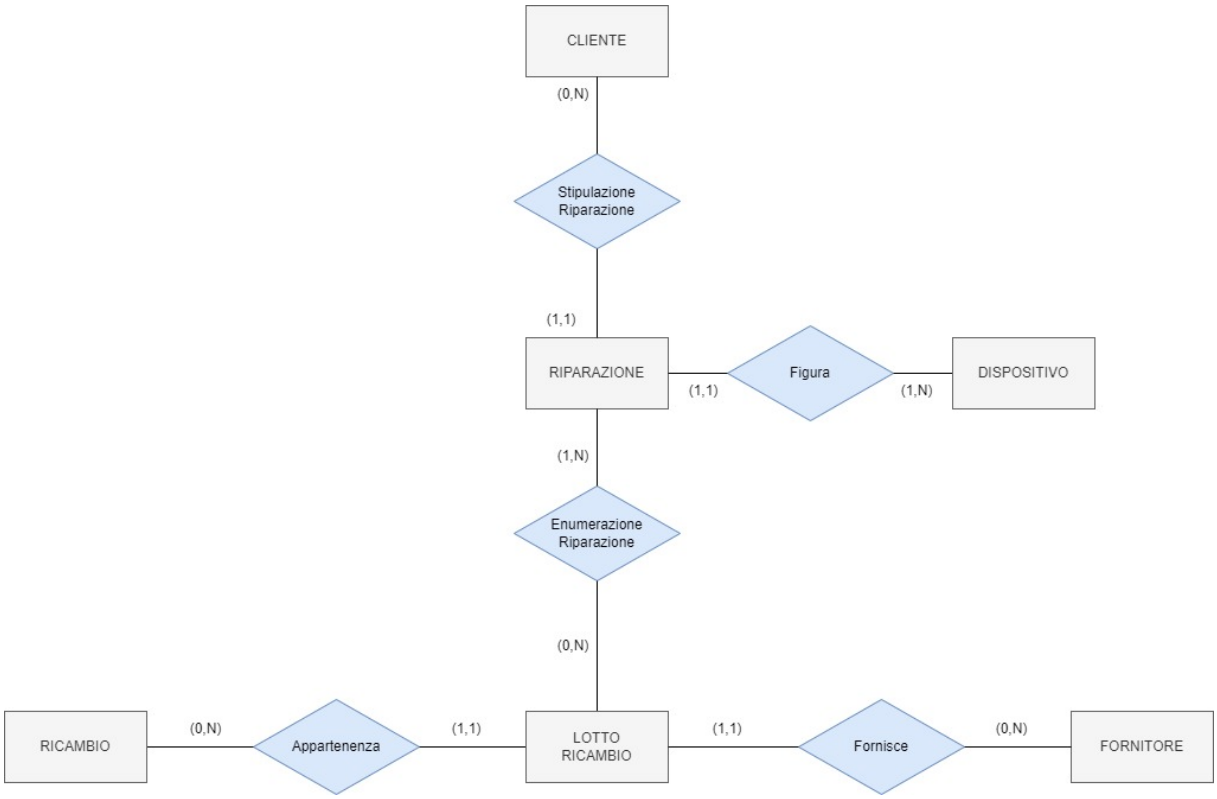

Figura 3.3: Entity Relationship Schema Semplificato

Alcune precisazioni

- L'entità **Ricambio** va intesa come classe di ricambio e.g. "SCHERMO IPHONE 13 PRO MAX";
- L'entità **Lotto Ricambio** va intesa come lo *specifico ricambio* ordinato presso un particolare fornitore ed appartenente ad una specifica classe di ricambio. Il *Lotto Ricambio* va inteso quindi come ricambio specifico riconoscibile ad esempio dal numero seriale "Y6E5FG4D0765H". Si è utilizzato il termine "lotto" in quanto possono essere ordinati più pezzi dal fornitore. Avremo quindi che ad una classe di Ricambio appartengono più Lotti Ricambio, ciascuno dei quali identificabili da un seriale. Tale scelta è stata ereditata dal progetto precedente, il centro di riparazione infatti ha la necessità, per fornire il servizio di garanzia sulla riparazione effettuata, di risalire a quale/quali lotti ricambio sono stati utilizzati per realizzare una particolare riparazione.

3.2 Progettazione di Aggregati

Nella seconda fase, del processo descritto in figura (Fig. 3.1), abbiamo realizzato la definizione degli aggregati. Definire un aggregato e la sua progettazione significa considerare due aspetti fondamentali, il primo è legato *all'entità radice*. Si considera infatti il concetto di annidamento, ovvero l'entità radice, è il macro-oggetto e contiene diversi micro-oggetti, suoi attributi. Mentre il secondo, è legato ai *confini* che dovrà avere l'aggregato, cioè quali delle entità e relazioni devono essere tenute insieme come elementi dell'entità radice.
Per una corretta definizione di questi aspetti va inoltre prestata particolare attenzione ai pattern di accesso ai dati, ovvero alla direzione con cui le operazioni di lettura e scrittura vengono effettuate. Nel nostro caso, le entità considerate ai fini della progettazione degli aggregati sono visibili nella figura (Fig. 3.3).
In tale analisi abbiamo quindi posto l'attenzione sulle possibili alternative nella scelta dell'entità radice e delle entità annidate per i vari aggregati riportati precedentemente.

Analisi Nei paragrafi precedenti abbiamo fatto le nostre prime considerazioni, senza però entrare nel merito del problema. Di seguito, attraverso una accurata analisi, andremo a definire gli aggregati che meglio si adattano alla realtà e che ha un *costo* minore per il contesto preso in esame.
Per realizzare tale analisi quantitativamente si utilizzerà la tavola dei volumi (Tab. 3.1), delle operazioni e le relative frequenze (Tab. 3.2).

CONCETTO	TIPO	VOLUME
Persona	E	1815
Cliente	E	1800
Fornitore	E	15
Contratto	E	24260
Riparazione	E	15000
Vendita Accessorio	E	6000
Ordine Fornitore	E	2980
Reso Fornitore	E	280
Dispositivo	E	5000
Garanzia	E	20
Prodotto	E	1730
Ricambio	E	600
Accessorio	E	1130
Lotto Ricambio	E	15000
Stipulazione Vendita	R	6000
Stipulazione Riparazione	R	15000
Stipulazione Reso	R	280
Figura	R	15000
Assegnazione Garanzia	R	15000
Enumerazione Riparazione	R	23000
Enumerazione Reso	R	340
Acquisto Ricambi	R	15000
Enumerazione Vendita	R	9000
Acquisto Accessori	R	4800
Appartenenza	R	15000

Tabella 3.1: Tavola dei volumi

OPERAZIONE	FREQUENZA
Inserimento di un nuovo fornitore	7 volte l'anno
Inserimento di un nuovo cliente	3 volte al giorno
Inserimento di un nuovo prodotto	3 volte al giorno
Inserimento di un nuovo lotto di ricambi	25 volte al giorno
Stipulazione di una riparazione	25 volte al giorno
Stipulazione di un ordine presso un fornitore	5 volte al giorno
Stipulazione di un reso presso un fornitore	3 volte alla settimana
Stipulazione di una vendita	10 volte al giorno
Modifica dati di un fornitore	3 volte all'anno
Modifica dati di un cliente	3 volte al mese
Aggiornamento dettagli di un reso	5 volte alla settimana
Aggiornamento dettagli di un ordine	5 volte al giorno
Aggiornamento dettagli di una riparazione	25 volte al giorno
Aggiornamento dettagli di un prodotto	3 volte al mese
Aggiornamento dettagli di un lotto di ricambio	5 volte al mese
Cancellazione di un prodotto	3 volte al mese
Annullamento di una richiesta di reso	3 volte all'anno
Annullamento di un ordine	10 volte all'anno
Visualizzazione dello stato di un ordine	10 volte al giorno
Visualizzazione dello stato di un reso	2 volte al giorno
Visualizzazione dettagli di una riparazione	20 volte al giorno
Visualizzazione di tutte le riparazioni effettuate da un cliente	10 volte al mese
Visualizzazione disponibilità di un prodotto	35 volte al giorno
Visualizzazione dei lotti ancora disponibili di un ricambio	15 volte al giorno
Visualizzazione di tutti i prodotti per un determinato tipo di dispositivo di una marca specifica	2 volte al mese
Visualizzazione prezzo di acquisto di un lotto di ricambio	2 volte al giorno
Visualizzazione dettagli di un prodotto	2 volte al giorno
Visualizzazione del fornitore di un lotto ricambio	3 volte alla settimana
Visualizzazione disponibilità di tutti i prodotti	2 volte alla settimana
Visualizzazione dati di un cliente	30 volte al giorno
Visualizzazione dati di un fornitore	3 volte al mese
Visualizzazione degli ordini effettuati presso uno specifico fornitore	3 volte al mese
Visualizzazione degli ordini effettuati presso tutti i fornitori in una certa data	15 volte al mese
Visualizzazione di tutti i resi effettuati	2 volte all'anno
Visualizzazione dei resi non ancora rimborsati	2 volte al mese
Visualizzazione delle riparazioni non ancora ritirate	2 volte alla settimana
Visualizzazione delle riparazioni in un determinato stato ("presa in carico" o "in riparazione" o "completata" o "ritirata" o "in garanzia")	8 volte al giorno
Statistiche sui venditori soggetti a più ricambi difettosi	2 volte all'anno
Statistiche sul numero di riparazioni effettuate in un arco di tempo	2 volte all'anno
Statistiche dei dispositivi (e relativi modelli) più riparati	5 volte all'anno
Statistiche sul tipo di riparazioni più effettuate	5 volte all'anno

Tabella 3.2: Tavola delle operazioni e relativa frequenza

Si noti che per la progettazione degli aggregati si sono considerate soltanto le operazioni più frequenti e relative alle entità rimanenti dello schema semplificato. Tali operazioni considerate sono evidenziate in grassetto nella tabella precedente (Tab. 3.2).
Oltre a tali operazioni si è deciso di introdurre ulteriori operazioni e le loro relative frequenze, allo scopo di adattare al meglio il progetto precedente per la progettazione e l'analisi di un sistema NoSQL. Le operazioni introdotte sono riportate di seguito:

- Visualizzazione lotti disponibili per il ricambio X (25 volte al giorno);
- Visualizzazione riparazioni realizzate per il dispositivo X (30 volte al giorno);
- Visualizza riparazioni realizzate per il cliente X (30 volte al giorno);
- Visualizza riparazioni realizzate utilizzando il lotto X (20 volte al giorno).

Un'ulteriore nota da fare è sulle entità Fornitore e Lotto Ricambio. Non avendo operazioni, fra quelle considerate, che prendono in considerazioni entrambe le entità si è scelto di considerare Fornitore come entità annidata all'entità radice Lotto Ricambio.
Nelle analisi realizzate di seguito si è considerato che il costo di una scrittura equivale al costo di due letture ($1S = 2L$).

3.2.1 Aggregato: Lotto Ricambio - Ricambio

In questa sezione si pone l'attenzione sull'aggregato composto dalle entità *Ricambio* e *Lotto Ricambio*.
Si sono analizzate le operazioni riguardanti tali entità così da calcolare e scegliere l'entità meno costosa, per le operazioni e le frequenze prese in considerazioni, a ricoprire il ruolo di entità radice. Si valuterà il costo di entrambe le alternative in cui un'entità ricopre il ruolo di entità radice e l'altra di entità annidata e viceversa.
Nella parte seguente si riportano i conti realizzati.

- Numero medio di lotti per ricambio $15.000/600 = 25$
- Numero medio di ricambi per lotto $15.000/15.000 = 1$

OPERAZIONE	**LOTTO in RICAMBIO**	**RICAMBIO in LOTTO**
Inserimento di un nuovo lotto di ricambi $(25/g)$	$1S \cdot 25 = 50L$	$1S \cdot 25 = 50L$
Visualizzazione lotti per ricambio X $(25/g)$	$1L \cdot 25 = 25L$	$25L \cdot 25 = 625L$
Totale	$75L$	$675L$

Tabella 3.3: Analisi Aggregato Lotto Ricambio - Ricambio

Conclusioni Dai calcoli precedenti si è scelto di considerare Ricambio come entità radice e Lotto Ricambio come entità annidata.

3.2.2 Aggregato: Dispositivo - Riparazione

In questa sezione vengono analizzate le entità Dispositivo e Riparazione. Valutando il costo delle operazioni che riguardano tali entità, come fatto precedentemente.

- Numero medio di riparazioni per dispositivo $15.000/5.000 = 3$
- Numero medio di dispositivi per riparazione $15.000/15.000 = 1$

OPERAZIONE	**RIPARAZIONE in DISPOSITIVO**	**DISPOSITIVO in RIPARAZIONE**
Stipulazione di una riparazione $(25/g)$	$1S \cdot 1 \cdot 25 = 50L$	$1S \cdot 25 = 50L$
Aggiornamento dettagli di una riparazione $(25/g)$	$1S \cdot 1 \cdot 25 = 50L$	$1S \cdot 25 = 50L$
Visualizzazione dettagli di una riparazione $(20/g)$	$1L \cdot 20 = 20L$	$1L \cdot 20 = 20L$
Visualizzazione riparazioni del dispositivo X $(30/g)$	$1L \cdot 30 = 30L$	$3L \cdot 30 = 90L$
Totale	$150L$	$210L$

Tabella 3.4: Analisi Aggregato Dispositivo - Riparazione

Conclusioni Dai costi delle operazioni presenti nella tabella 3.4 si è deciso di scegliere Dispositivo come entità radice e Riparazione come entità annidata.

3.2.3 Aggregato: Cliente - Riparazione

Si sono analizzate poi le entità Cliente e Riparazione.

- Numero medio di di riparazioni per cliente $15.000/1.800 = 8,\bar{3}$
- Numero medio di clienti per riparazione $15.000/15.000 = 1$

Operazione	**RIPARAZIONE in CLIENTE**	**CLIENTE in RIPARAZIONE**
Stipulazione di una riparazione $(25/g)$	$1S \cdot 1 \cdot 25 = 50L$	$1S \cdot 25 = 50L$
Aggiornamento dettagli di una riparazione $(25/g)$	$1S \cdot 1 \cdot 25 = 50L$	$1S \cdot 25 = 50L$
Visualizzazione dettagli di una riparazione $(20/g)$	$1L \cdot 20 = 20L$	$1L \cdot 20 = 20L$
Visualizzazione riparazioni del cliente X $(30/g)$	$1L \cdot 30 = 30L$	$8,\bar{3}L \cdot 30 = 249L$
Totale	$150L$	$369L$

Tabella 3.5: Analisi Aggregato Cliente - Riparazione

Conclusioni Dai costi delle operazioni presenti nella tabella 3.5 si è deciso di scegliere Cliente come entità radice e Riparazione come entità annidata.

3.2.4 Aggregato: Lotto Ricambio - Riparazione

Infine, si sono analizzate le entità *Lotto Ricambio* e *Riparazione*, i conti svolti sono visibili nella tabella 3.6.

- Numero medio di lotti per riparazione $23.000/15.000 = 1,5\overline{3}$
- Numero medio di riparazioni per lotto $23.000/15.000 = 1,5\overline{3}$

OPERAZIONE	**Riparazione in Lotto**	**Lotto in Riparazione**	**K(Lotto) in Riparazione**
Stipulazione di una riparazione (25/g)	$1,5S \cdot 25 = 75L$	$1S \cdot 25 = 50L$	$1S \cdot 25 = 50L$
Aggiornamento dettagli di una riparazione (25/g)	$1,5S \cdot 25 = 75L$	$1S \cdot 25 = 50L$	$1S \cdot 25 = 50L$
Visualizzazione dettagli di una riparazione (20/g)	$1,5L \cdot 20 = 30L$	$1L \cdot 20 = 20L$	$1L \cdot 20 = 20L$
Inserimento di un nuovo lotto di ricambi (25/g)	$1S \cdot 25 = 50L$	$1,5S \cdot 25 = 75L$	$1S \cdot 25 = 50L$
Visualizzazione riparazioni del lotto X (20/g)	$1L \cdot 20 = 20L$	$1,5S \cdot 20 = 30L$	$1,5L \cdot 20 = 30L$
TOTALE	$250L$	$225L$	$200L$

Tabella 3.6: Analisi Aggregato Lotto Ricambio - Riparazione

In questo caso, oltre a considerare le alternative in cui ciascuna entità è l'entità radice si è considerata l'alternativa con i collegamenti, in particolare considerando la chiave di Lotto Ricambio dentro l'entità Riparazione.
Visti i cosi minori e per una maggiore completezza dell'elaborato, si è deciso di optare per la soluzione che sfrutta i *collegamenti*.

3.2.5 Aggregati finali

Ricapitolando, gli aggregati finali ottenuti dalle analisi svolte precedentemente sono:

- Fornitore in Lotto Ricambio in Ricambio, in questo primo aggregato abbiamo quindi che Ricambio è l'entità radice;
- Riparazione in Dispositivo in Cliente, in questo secondo aggregato abbiamo che l'entità Cliente è l'entità radice.

Inoltre, come detto precedentemente, si è previsto il collegamento tra l'aggregato relativo a *Lotto Ricambio* e *Riparazione*.

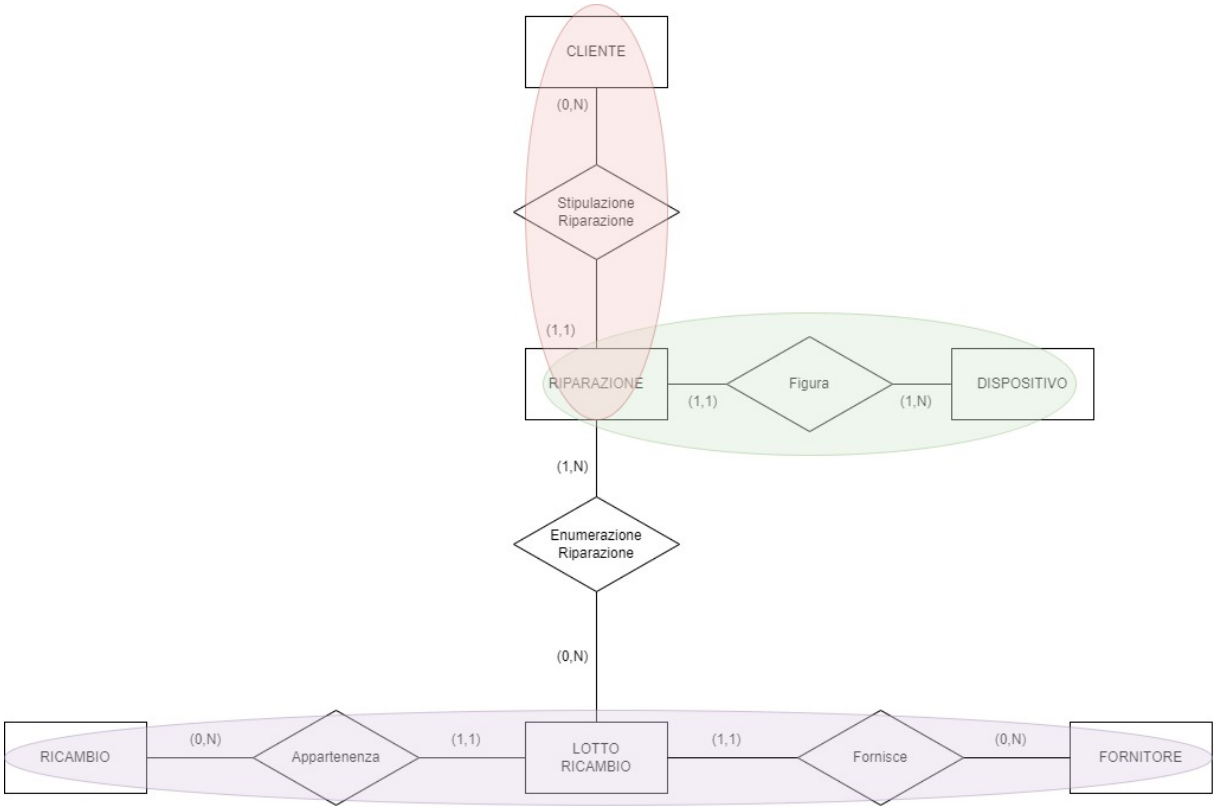

Figura 3.4: Aggregati a livello 1

I singoli aggregati che sono stati analizzati precedentemente sono rappresentati in figura 3.4. Nella figura 3.5 sono invece rappresentati i due macro-aggregati ottenuti e i collegamenti fra i due.

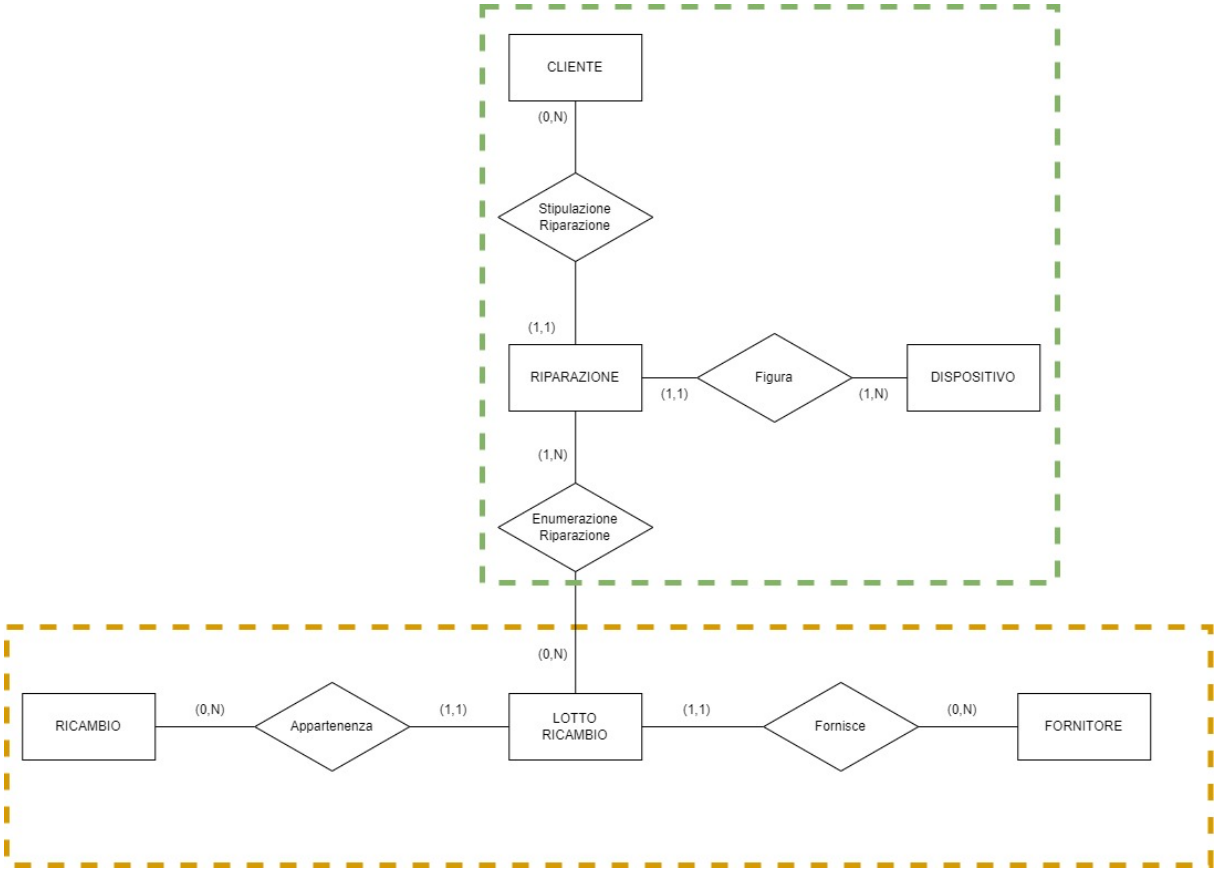

Figura 3.5: Aggregati a livello 2

3.3 Rappresentazione di Aggregati

Finora abbiamo compreso la natura degli aggregati e definito quest'ultimi a livello schematico. Ovvero, abbiamo definito quelli che sono i confini dell'aggregato, cioè quali entità vengono unite tra loro e quali invece rimangono distinte in altri aggregati.
In estrema sintesi, nel caso in esame possiamo concludere che, tenendo conto delle analisi svolte precedentemente, la soluzione ottimale si ha se consideriamo Fornitore in Lotto Ricambio che a sua volta si trova all'interno di Ricambio. Il secondo aggregato ha come entità radice Cliente e come entità annidate Dispositivo e Riparazione. Inoltre, abbiamo previsto il collegamento tra l'aggregato relativo alla *"fornitura"* e l'aggregato relativo alla *"riparazione"*.
Nella fase dedita alla rappresentazione va però definito come poter rappresentare gli aggregati, ovvero il modello di rappresentazione adottato. Nel nostro caso è stato adottato il modello *NoAM*[1].
Nella fattispecie, l'obiettivo sarà quello di delineare in modo puntuale la definizione di blocco e la sua composizione in termini di *entries*. Per la forma delle *entries* abbiamo scelto la strategia di rappresentazione ETF (*Entry Per Top Level Field*). In questo caso abbiamo un'entry per ogni attributo della radice e un'entry complessa per gli oggetti annidati. Dalla progettazione degli aggregati abbiamo ottenuto diversi livelli di annidamento. Nella parte seguente, vedremo i risultati di tale fase.

[1]NoAM ⊨ NoSQL Abstract Data Model è una metodologia che è stata introdotta recentemente da P. Atzeni, F. Bugiotti et al.

3.3.1 Rappresentazione aggregato Cliente-Dispositivo-Riparazione

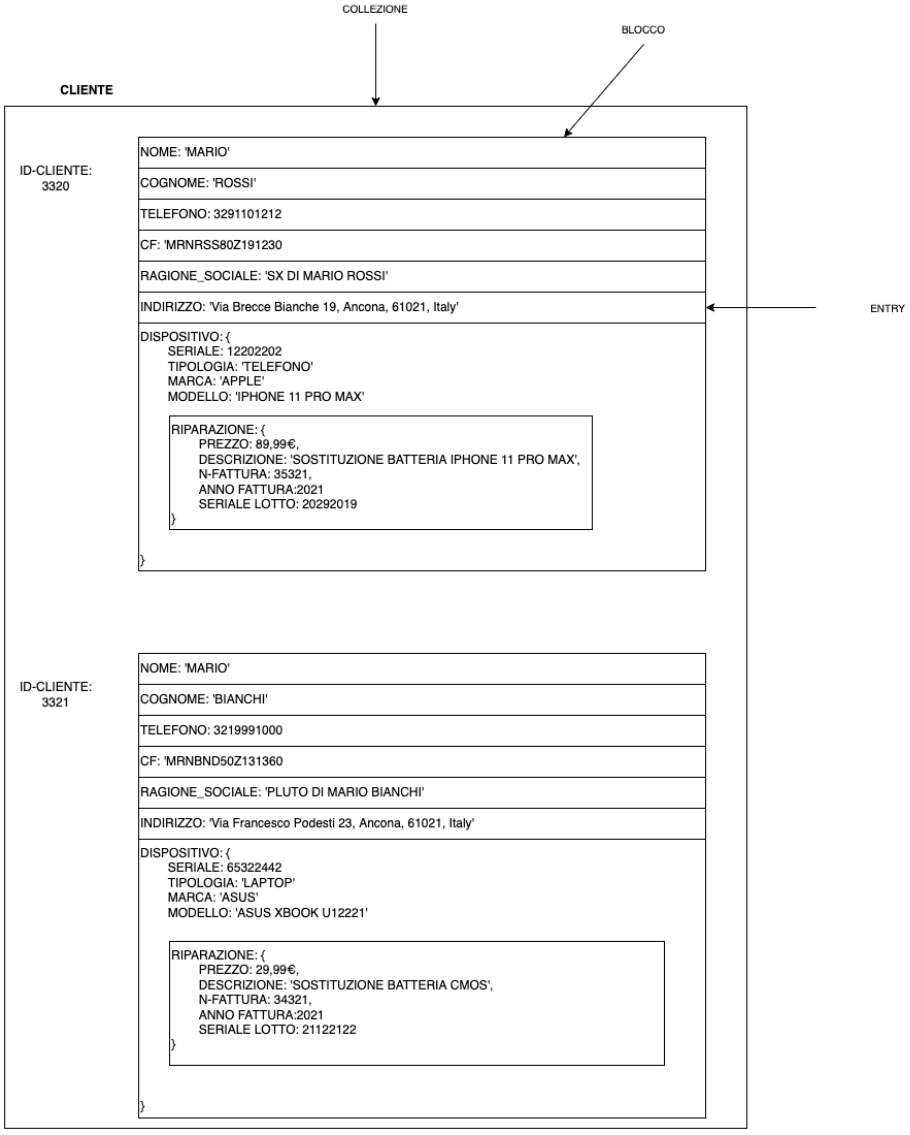

Figura 3.6: Rappresentazione Aggregato Cliente-Dispositivo-Riparazione

3.3.2 Rappresentazione aggregato Ricambio-Lotto Ricambio-Fornitore

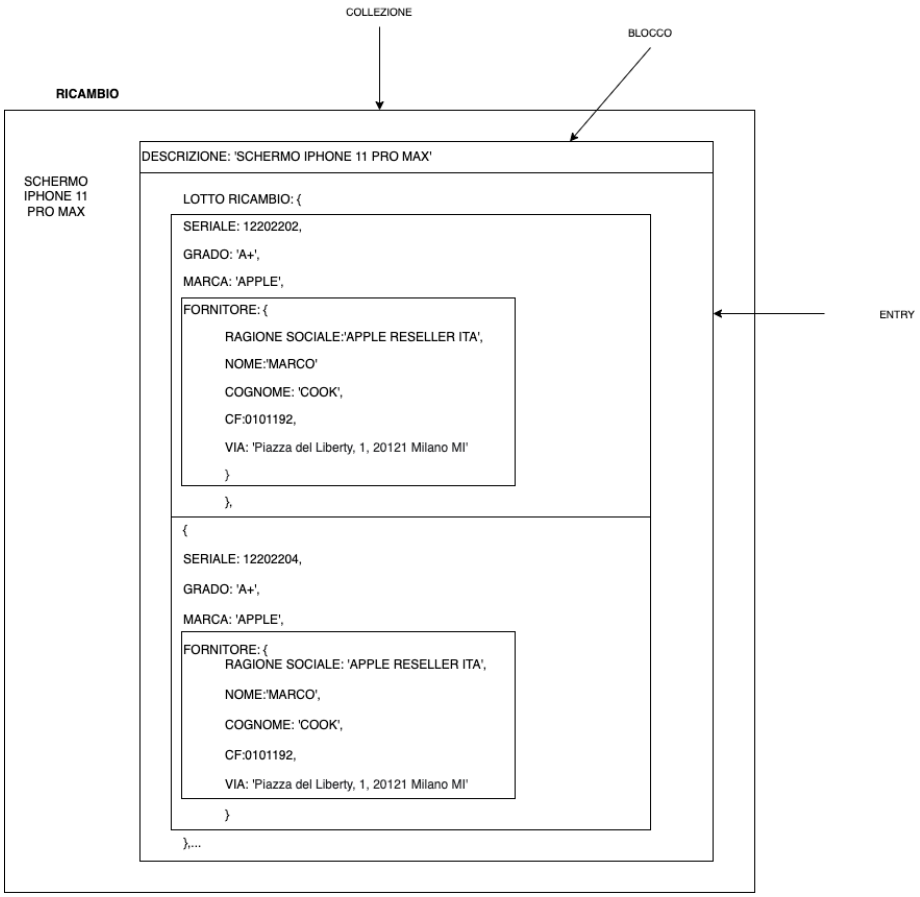

Figura 3.7: Rappresentazione Aggregato Ricambio-Lotto Ricambio-Fornitore

Pattern di Accesso Riportiamo quindi i pattern di accesso relativi all'aggregato users.

- Per accedere a riparazione il pattern è il seguente: Users → Dispositivo → Riparazione

Mentre per l'aggregato ricambi il pattern è il seguente:

- Ricambi → Lotto Ricambi → Fornitore

EAO vs ETF Come già visto tramite i pattern di accesso, l'idea di utilizzare la strategia EAO[2], andrebbe in conflitto con l'obiettivo di ridurre le computational performance, in quanto per accedere ad un singolo "attributo" dovremmo caricare l'intero blocco rappresentante l'oggetto e successivamente estrarre la porzione dei dati a noi necessaria. Ecco, quindi che riconfermiamo la scelta della strategia ETF.

Index Va precisato come MongoDB, sia tramite Shell che tramite GUI Compass, permette la creazione di indici. MongoDB crea di default un indice associato al campo *_id*, ovvero la chiave di ciascun oggetto. Questo campo è previsto di default e viene creato automaticamente nella fase di inserimento di nuovi dati. La sua funzione è quella di identificare univocamente gli oggetti

[2]EAO ⊨ Entry per Aggregate Object

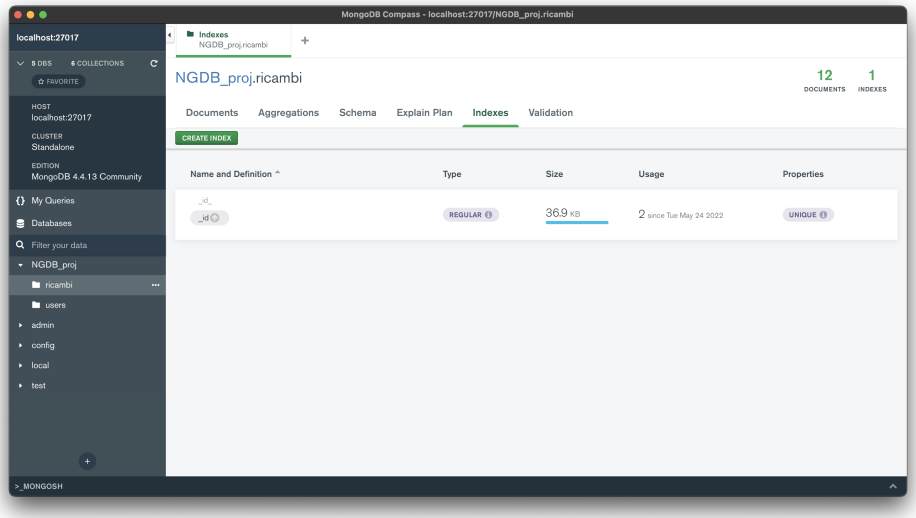

Figura 3.8: Creazione Index Tramite l'applicativo Compass

presenti all'interno del database. Va inoltre precisato che non è possibile eliminare tale indice, può soltanto subire variazioni.
Come detto, si possono creare nuovi indici oltre a quello di default, utilizzando la shell, sarà sufficiente digitare il seguente commando:

```
1 db.collection.createIndex( { name: -1 } )
```

Utilizzando invece la GUI Compass sarà sufficiente selezionare la collection, successivamente andare nella voce "Indexes" nella barra in alto e creare il nuovo indice. Nella figura Fig.3.8 è possibile vedere tale schermata. Nel nostro progetto è stato creato un indice per il campo "CF/PIVA" dell'utente. Questo può essere visto nella figura 3.9.

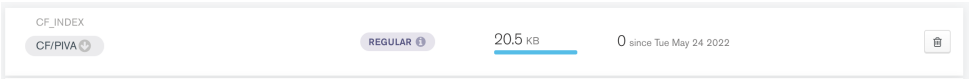

Figura 3.9: CF/PIVA Index

3.4 Partioning

Andando a considerare l'analisi del partizionamento orizzontale, nel nostro progetto sono presenti operazioni che accedono frequentemente ad un sottoinsieme delle occorrenze. Le suddette operazioni lavorano spesso su una stessa porzione di dati; questo pattern è tipicamente legato alle storicizzazioni (lavoriamo su dati odierni mantenendo comunque anche dati storici). L'idea del *partitioning* è quella di mantenerli separati in modo da permettere accessi più veloci alle query che lavorano su dati recenti, queste sono di solito quelle più frequenti.
Considerando però le elevate prestazioni raggiunte, abbiamo deciso di non intraprendere tale processo nel nostro progetto. Tale scelta è stata fatta anche nel ottica di non complicare la trattazione del presente elaborato.

3.5 Traduzione

L'ultima fase è quella di traduzione, dal modello NoAM al modello NoSQL scelto. Questo è un problema di traduzione dello schema, la soluzione è abbastanza immediata ed è basata sulla definizione del modello NoAM che avevamo precedentemente introdotto.
Abbiamo deciso di adottare il modello *document store* in cui il blocco radice è suddiviso in un insieme di coppie chiave-valore che rappresentano i singoli elementi. Come tecnologia abbiamo scelto *MongoDB*, questo rappresenta una delle soluzioni principali dei *document store*. Il risultato di questa fase è un JSON. Ad esempio, la struttura per l'aggregato *Cliente-Dispositivo-Riparazione* è la seguente:

```
{
  "CF/PIVA": "KJYT75082778426345",
  "Nome": "Lonnie",
  "Cognome": "Cisneros",
  "Telefono": "873.007.4663x81396",
  "Indirizzo": {
    "Via": "Ali Knolls",
    "Civico": "4257",
    "Città": "North Mariachester",
    "CAP": "73759",
    "Provincia": "LR"
  },
  "Dispositivo": [
    {
      "Tipologia": "Smartphone",
      "Marca": "Apple",
      "Modello": "iPhone XS",
      "Codice dispositivo": "3626075236259",
      "Riparazione": [
        {
          "Codice": 12,
          "Descrizione": "Ever stage become camera.",
          "Prezzo": "81 e",
          "Numero fattura": "2176707109839",
          "Data": "1977-06-25",
          "Seriale lotto": [
            "9243012793763"
          ],
          "Stato riparazione": "rientrata in garanzia",
          "Sede di ritiro": "sede2"
        }
        ]
    }
  ]
}
```

Listing 3.1: Implementazione in JSON

3.6 Precisazioni

Per una maggior fruizione della relazione, nelle figure 3.6 e 3.7 sono rappresentati soltanto una parte degli attributi. Analogo discorso per quanto concerne il listato 3.1.

Capitolo 4

Implementazione

Una volta conclusa la progettazione, fase principale dell'elaborato in esame, si è svolta la fase di implementazione, in particolare l'implementazione in un database NoSQL. In seguito si effettueranno e si riporteranno anche le interazioni con quest'ultimo realizzate sia tramite la *Shell* che tramite la GUI *MongoDB Compass*.
Tra i principali problemi avuti durante questa fase, non possiamo che non citare la creazione dei dati e la loro importazione all'interno del nostro database. Per risolvere tali problematiche abbiamo scelto di realizzare uno script Python che ci ha consentito di generare ed importare in modo automatico i dati all'interno del database.
Gli steps di setup intermedi, fondamentali per la buona implementazione del progetto, sono stati: l'installazione di MongoDB Community, così da avere il massimo supporto possibile e l'installazione della GUI MongoDB Compass.
Una particolare precisione va fatta nell'ordine d'uso delle due interfacce, shell e GUI, la prima ci ha permesso di attuare un lavoro più specifico, mentre la seconda ci ha permesso una maggior fruizione dei risultati delle varie query. Nelle figure sottostanti si riportano le schermate relative alla GUI Compass (Fig.4.2). Nella figura 4.1 è possibile invece visionare alcuni principali comandi per avviare e/o spegnere il *daemon* di MongoDB, e come avviare la shell, il parametro **-p** serve a specificare la porta utilizzata dal server.

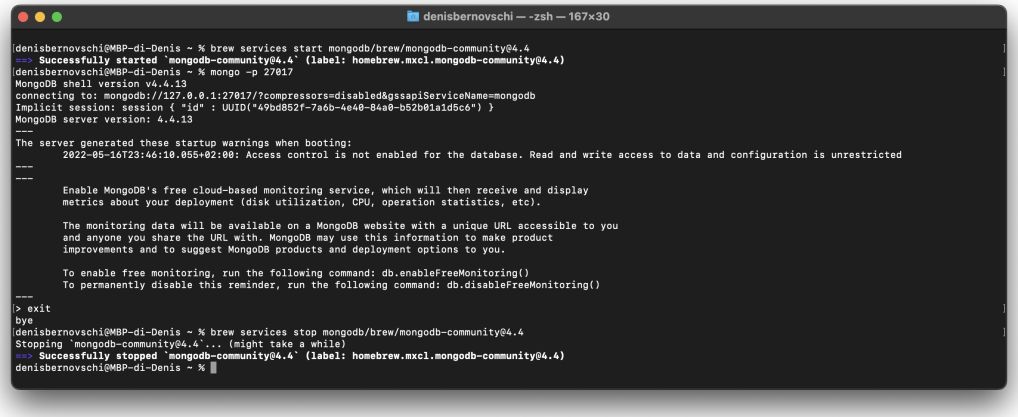

Figura 4.1: Shell Interface MongoDB

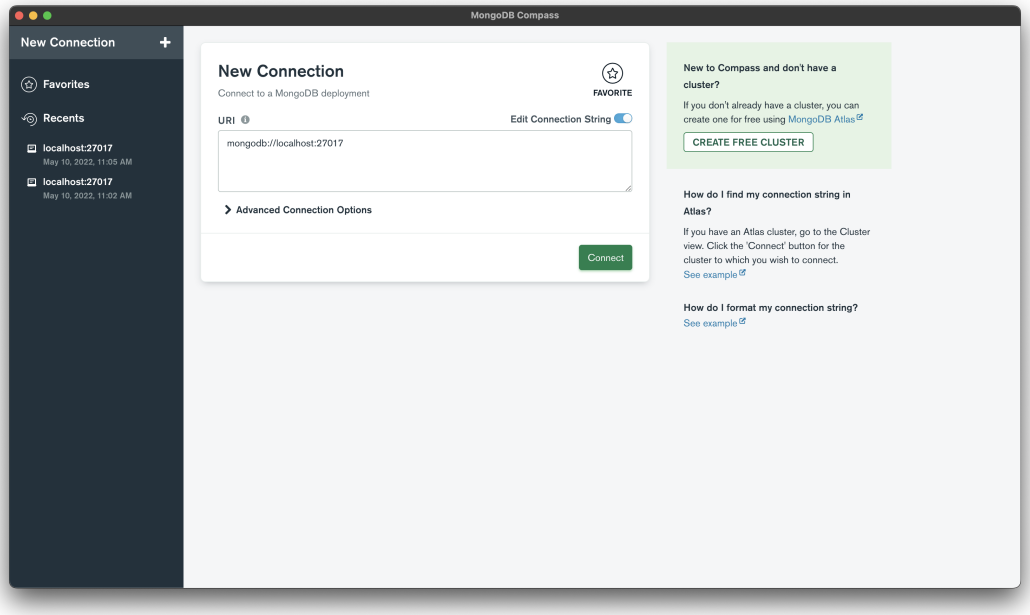

Figura 4.2: Graphic User Interface MongoDB Compass

4.1 Query

4.1.1 Inserimento di un nuovo lotto di ricambi

In questa prima query vediamo l'inserimento di un insieme di lotti ricambi appartenenti ad una nuova classe di ricambio, in questo caso la classe "Tastiera" per il MacBook Pro.

```
db.ricambi.insertOne({
"Nome": "Tastiera",
"Marca dispositivo": "Apple",
"Modello dispositivo": "MacBook Pro",
"Tipologia dispositivo": "Laptop",
"Note aggiuntive": "",
"Lotto Ricambio": [{
        "Seriale": "3405476226003",
        "Grado": "D",
        "Data": "2015-04-22",
        "Disponibilità": 3,
        "Fornitore": {
            "CF/PIVA": "BVAS43207354641967",
            "Ragione Sociale": "Christopher Hughes s.r.l",
            "Nome": "Christopher",
            "Cognome": "Hughes",
            "Telefono": "001-708-587-3055x87184",
            "Tipologia": "Laptop",
            "Sito Web": "dunn-galloway.com",
            "Costi di spedizione": "4 e",
            "Tempi di spedizione": "4 gg",
            "Indirizzo": {
                "Via": "Taylor Wells",
                "Civico": "69969",
                "Citta": "South Lisa",
                "CAP": "23984",
                "Provincia": "BT"
                }
            }
        },{
        "Seriale": "6372037189541",
        "Grado": "A",
        "Data": "2005-03-11",
        "Disponibilità": 2,
        "Fornitore": {
            "CF/PIVA": "OIZO88389291564793",
            "Ragione Sociale": "Christina Crawford s.r.l",
            "Nome": "Christina",
            "Cognome": "Crawford",
            "Telefono": "001-499-447-7876x933",
            "Tipologia": "Laptop",
            "Sito Web": "flores.com",
            "Costi di spedizione": "8 e",
            "Tempi di spedizione": "5 gg",
            "Indirizzo": {
                "Via": "Lamb Glen",
                "Civico": "244",
                "Citta": "South Paulstad",
                "CAP": "12233",
                "Provincia": "MD"
                }
            }
        },{
        "Seriale": "2324274245282",
```

```
        "Grado": "D",
        "Disponibilità": 5,
        "Fornitore": {
            "CF/PIVA": "BRNDNS96P06Z140X",
            "Ragione Sociale": "DB 2 s.r.l",
            "Nome": "Denis",
            "Cognome": "Bernovschi",
            "Telefono": "824-092-0329x0199",
            "Tipologia": "Laptop",
            "Sito Web": "flores.com",
            "Costi di spedizione": "8 e",
            "Tempi di spedizione": "5 gg",
            "Indirizzo": {
                "Via": "Tracey Parkway",
                "Civico": "51176",
                "Citta'": "North Amberton",
                "CAP": "25504",
                "Provincia": "MR"
                }
            }
        }]}
);

{ acknowledged: true, insertedId: ObjectId("628d2d10dd3057195be475f4") }
```

Listing 4.1: Inserimento lotto ricambi

Come si può vedere dal listato precedente nell'inserimento si è omesso di specificare il parametro "_id" in quanto questo verrà creato automaticamente da MongoDB.

Di seguito invece si riporta una query in cui si inseriscono soltanto le informazioni relative al lotto di ricambio, una volta identificata la classe di ricambio per cui vogliamo aggiungere un nuovo lotto.

```
db.ricambi.updateOne(
    {"Nome": "Tastiera",
      "Marca dispositivo": "Apple",
      "Modello dispositivo": "MacBook Pro",
      "Tipologia dispositivo": "Laptop"},
    {$push:
        {"Lotto Ricambio":
            {"Seriale": "2324274245283",
            "Grado": "D",
            "Disponibilità": 3,
                "Fornitore":{
                    "CF/PIVA": "BRNDNS96P06Z140X",
                    "Ragione Sociale": "DB 2 s.r.l",
                    "Nome": "Denis",
                    "Cognome": "Bernovschi",
                    "Telefono": "824-092-0329x0199",
                    "Tipologia": "Laptop",
                    "Sito Web": "flores.com",
                    "Costi di spedizione": "8 e",
                    "Tempi di spedizione": "5 gg",
                    "Indirizzo": {
                        "Via": "Tracey Parkway",
                        "Civico": "51176",
                        "Citta'": "North Amber25504",
                        "Provincia": "MR"
                    }
                }
            }
```

```
29          }
30      }
31 );
32 { "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

Listing 4.2: Inserimento (esclusivo) lotto ricambi

Come si evince dall'output (riga n.78 del listato 4.1 e riga n.32 del listato 4.2) di ambo le soluzioni proposte, il nuovo lotto o i nuovi lotti sono stati inseriti con successo.
Nei listati successivi, per verifica, si riporta la query e il relativo output per controllare il risultato degli inserimenti fatti precedentemente.

```
1 db.ricambi.find({
2 "Nome": "Tastiera",
3 "Marca dispositivo": "Apple",
4 "Modello dispositivo": "MacBook Pro",
5 "Tipologia dispositivo": "Laptop"
6 });
```

Listing 4.3: Query di verifica

```
1  { _id: ObjectId("628f8679c74ed1c0a44e4bae"),
2    Nome: 'Tastiera',
3    'Marca dispositivo': 'Apple',
4    'Modello dispositivo': 'MacBook Pro',
5    'Tipologia dispositivo': 'Laptop',
6    'Note aggiuntive': '',
7    'Lotto Ricambio':
8     [ { Seriale: '3405476226003',
9         Grado: 'D',
10        Data: '2015-04-22',
11        'Disponibilità': 3,
12        Fornitore:
13         { 'CF/PIVA': 'BVAS43207354641967',
14           'Ragione Sociale': 'Christopher Hughes s.r.l',
15           Nome: 'Christopher',
16           Cognome: 'Hughes',
17           Telefono: '001-708-587-3055x87184',
18           Tipologia: 'Laptop',
19           'Sito Web': 'dunn-galloway.com',
20           'Costi di spedizione': '4 e',
21           'Tempi di spedizione': '4 gg',
22           Indirizzo:
23            { Via: 'Taylor Wells',
24              Civico: '69969',
25              Citta: 'South Lisa',
26              CAP: '23984',
27              Provincia: 'BT' } } },
28      { Seriale: '6372037189541',
29        Grado: 'A',
30        Data: '2005-03-11',
31        'Disponibilità': 2,
32        Fornitore:
33         { 'CF/PIVA': 'OIZO88389291564793',
34           'Ragione Sociale': 'Christina Crawford s.r.l',
35           Nome: 'Christina',
```

```
36            Cognome: 'Crawford',
37            Telefono: '001-499-447-7876x933',
38            Tipologia: 'Laptop',
39            'Sito Web': 'flores.com',
40            'Costi di spedizione': '8 e',
41            'Tempi di spedizione': '5 gg',
42            Indirizzo:
43             { Via: 'Lamb Glen',
44               Civico: '244',
45               Citta: 'South Paulstad',
46               CAP: '12233',
47               Provincia: 'MD' } } },
48       { Seriale: '2324274245282',
49         Grado: 'D',
50         'Disponibilità': 5,
51         Fornitore:
52          { 'CF/PIVA': 'BRNDNS96P06Z140X',
53            'Ragione Sociale': 'DB 2 s.r.l',
54            Nome: 'Denis',
55            Cognome: 'Bernovschi',
56            Telefono: '824-092-0329x0199',
57            Tipologia: 'Laptop',
58            'Sito Web': 'flores.com',
59            'Costi di spedizione': '8 e',
60            'Tempi di spedizione': '5 gg',
61            Indirizzo:
62             { Via: 'Tracey Parkway',
63               Civico: '51176',
64               'Citta\'': 'North Amberton',
65               CAP: '25504',
66               Provincia: 'MR' } } },
67       { Seriale: '2324274245283',
68         Grado: 'D',
69         'Disponibilità': 3,
70         Fornitore:
71          { 'CF/PIVA': 'BRNDNS96P06Z140X',
72            'Ragione Sociale': 'DB 2 s.r.l',
73            Nome: 'Denis',
74            Cognome: 'Bernovschi',
75            Telefono: '824-092-0329x0199',
76            Tipologia: 'Laptop',
77            'Sito Web': 'flores.com',
78            'Costi di spedizione': '8 e',
79            'Tempi di spedizione': '5 gg',
80            Indirizzo:
81             { Via: 'Tracey Parkway',
82               Civico: '51176',
83               'Citta\'': 'North Amber25504',
84               Provincia: 'MR' } } } ] }
```

Listing 4.4: Risultato

4.1.2 Visualizzazione Lotti Disponibili per Ricambio X

Nella seguente query andiamo a visualizzare tutti i **Lotti Ricambi disponibili** per una determinata classe di ricambio.

```
db.ricambi.aggregate([
    {$match: {"Nome":"Tastiera", "Marca dispositivo": "Apple", "Modello
    dispositivo": "MacBook Pro", "Lotto Ricambio.Disponibilità": {$gt: 0}}},
    {$project: {
        "Lotto Ricambio": {$filter: {
            input: '$Lotto Ricambio',
            as: 'lotto',
            cond: {$gt: ['$$lotto.Disponibilità', 0]}
        }}
    }}
])
```

Listing 4.5: Visualizzazione Lotti per ricambio X

Il risultato della query presente nel listato 4.5 è il seguente:

```
{ _id: ObjectId("628f8679c74ed1c0a44e4bae"),
  'Lotto Ricambio':
   [ { Seriale: '3405476226003',
       Grado: 'D',
       Data: '2015-04-22',
       'Disponibilità': 3,
       Fornitore:
        { 'CF/PIVA': 'BVAS43207354641967',
          'Ragione Sociale': 'Christopher Hughes s.r.l',
          Nome: 'Christopher',
          Cognome: 'Hughes',
          Telefono: '001-708-587-3055x87184',
          Tipologia: 'Laptop',
          'Sito Web': 'dunn-galloway.com',
          'Costi di spedizione': '4 e',
          'Tempi di spedizione': '4 gg',
          Indirizzo:
           { Via: 'Taylor Wells',
             Civico: '69969',
             Citta: 'South Lisa',
             CAP: '23984',
             Provincia: 'BT' } } },
     { Seriale: '6372037189541',
       Grado: 'A',
       Data: '2005-03-11',
       'Disponibilità': 2,
       Fornitore:
        { 'CF/PIVA': 'OIZO88389291564793',
          'Ragione Sociale': 'Christina Crawford s.r.l',
          Nome: 'Christina',
          Cognome: 'Crawford',
          Telefono: '001-499-447-7876x933',
          Tipologia: 'Laptop',
          'Sito Web': 'flores.com',
          'Costi di spedizione': '8 e',
          'Tempi di spedizione': '5 gg',
          Indirizzo:
           { Via: 'Lamb Glen',
```

```
39              Civico: '244',
40              Citta: 'South Paulstad',
41              CAP: '12233',
42              Provincia: 'MD' } } },
43      { Seriale: '2324274245282',
44        Grado: 'D',
45        'Disponibilità': 5,
46        Fornitore:
47         { 'CF/PIVA': 'BRNDNS96P06Z140X',
48           'Ragione Sociale': 'DB 2 s.r.l',
49           Nome: 'Denis',
50           Cognome: 'Bernovschi',
51           Telefono: '824-092-0329x0199',
52           Tipologia: 'Laptop',
53           'Sito Web': 'flores.com',
54           'Costi di spedizione': '8 e',
55           'Tempi di spedizione': '5 gg',
56           Indirizzo:
57            { Via: 'Tracey Parkway',
58              Civico: '51176',
59              'Citta\'': 'North Amberton',
60              CAP: '25504',
61              Provincia: 'MR' } } },
62      { Seriale: '2324274245283',
63        Grado: 'D',
64        'Disponibilità': 3,
65        Fornitore:
66         { 'CF/PIVA': 'BRNDNS96P06Z140X',
67           'Ragione Sociale': 'DB 2 s.r.l',
68           Nome: 'Denis',
69           Cognome: 'Bernovschi',
70           Telefono: '824-092-0329x0199',
71           Tipologia: 'Laptop',
72           'Sito Web': 'flores.com',
73           'Costi di spedizione': '8 e',
74           'Tempi di spedizione': '5 gg',
75           Indirizzo:
76            { Via: 'Tracey Parkway',
77              Civico: '51176',
78              'Citta\'': 'North Amber25504',
79              Provincia: 'MR' } } } ] }
```

Listing 4.6: Risultato - Visualizzazione Lotti per ricambio X

In questa query può essere notato l'utilizzo dell'operatore "$gt" che ci ha permesso di estrarre solamente i lotti ricambio appartenenti alla classe di ricambio specificata ancora disponibili e quindi non utilizzati per altre riparazioni. Si è inoltre utilizzato l'aggregate per poter effettuare prima il match e poi la proiezione.

4.1.3 Stipulazione di una riparazione

Per quanto riguarda la stipulazione di una riparazione, mostriamo due possibile alternative:

1. Identifico il cliente (tramite "_id") e il dispositivo del cliente da riparare tramite seriale.

```
db.users.updateOne(
    {"_id" :ObjectId("628d318a2ef1dd8bc142462a"),
    "Dispositivo.Codice dispositivo" : "2531980246723"},
    {$push: {
        "Dispositivo.$.Riparazione": {
        "Codice": 28,
        "Descrizione": "Open Mr doctor deep show.",
        "Prezzo": "35 e",
        "Numero fattura": "5511730370462",
        "Data": "2018-11-05",
        "Seriale lotto": ["0769124747337"],
        "Stato riparazione": "rientrata in garanzia",
        "Sede di ritiro": "lab"}
        }
    });
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Listing 4.7: Stipulazione di una riparazione V2

2. In questo secondo esempio si identifica il cliente (attraverso il codice fiscale) ed il dispositivo attraverso il seriale.

```
db.users.updateOne(
    {"CF/PIVA":"BSRS42628511198154", "Dispositivo.Codice dispositivo" : "
    9742194491394"},
    {$push:
        {"Dispositivo.$.Riparazione": {
        "Codice": 28,
        "Descrizione": "Open Mr doctor deep show.",
        "Prezzo": "35 e",
        "Numero fattura": "5511730370462",
        "Data": "2018-11-05",
        "Seriale lotto": ["0769124747337"],
        "Stato riparazione": "rientrata in garanzia",
        "Sede di ritiro": "lab"}
        }
    }
);
```

Listing 4.8: Stipulazione di una nuova riparazione

Nella listato successivo presentiamo il risultato di tali implementazioni, ovvero l'inserimento della nuova riparazione.

```
{  "_id": {"$oid": "628d318a2ef1dd8bc142462a"},
"CF/PIVA": "BSRS42628511198154",
"Nome": "Ryan",
"Cognome": "Campbell",
"Telefono": "001-932-965-0808x8243",
"Indirizzo": {"Via": "Peter Ridge",
"Civico": "3250",
"Citta'": "Bentleyton",
"CAP": "59007",
"Provincia": "SR"},
"Dispositivo": {
    "Tipologia": "Tablet",
    "Marca": "Samsung",
    "Modello": "Galaxy Tab S6 Lite",
    "Codice dispositivo": "9742194491394",
```

```
    "Riparazione": [
    {...},
    {...},
    {
        "Codice": 28,
        "Descrizione": "Open Mr doctor deep show.",
        "Prezzo": "35 e",
        "Numero fattura": "5511730370462",
        "Data": "2018-11-05",
        "Seriale lotto": ["0769124747337"],
        "Stato riparazione": "rientrata in garanzia",
        "Sede di ritiro": "lab"}
    ]
}
```

Listing 4.9: Risultati

4.1.4 Aggiornamento dettagli di una riparazione

```
db.users.update({"Dispositivo.Riparazione.Codice": 76}, {$set: {"Dispositivo.$[i
    ].Riparazione.$[j].Descrizione": "Descrizione aggiornata"}}, {arrayFilters:
    [{"i.Modello": {$ne: ""}}, {"j.Codice": 76}]})
```

Estraendo le informazioni della riparazione aggiornata con la query precedente si ottiene il seguente risultato, ovvero la descrizione della query risulta essere stata aggiornata.
In questa query sono da notare gli **operatori posizionali** necessari in quanto abbiamo un array innestato in un ulteriore array.

```
{ _id: ObjectId("628dfbe1861ad4dfe01fb4c5"),
  Dispositivo:
   { Riparazione:
      { Codice: 888,
        Descrizione: 'Descrizione aggiornata',
        Prezzo: '75 e',
        'Numero fattura': '9045620128377',
        Data: '1993-04-01',
        'Seriale lotto':
         [ '7886980787949',
           '7886980787949',
           '1362659263854',
           '0254111752499' ],
        'Stato riparazione': 'rientrata in garanzia',
        'Sede di ritiro': 'lab' } } }
```

Listing 4.10: Risultati

4.1.5 Visualizzazione dettagli di una riparazione

Nella query seguente si ottengono i dettagli della riparazione specificando direttamente il codice della riparazione.

```
db.users.aggregate([
    {$unwind: '$Dispositivo'},
    {$unwind: '$Dispositivo.Riparazione'},
  {$match: {"Dispositivo.Riparazione.Codice": 59}},
  {$project: {"Dispositivo.Riparazione":1}}
```

```
6   ]);
```

Listing 4.11: Visualizzazione dettagli di una riparazione

L'output è il seguente:

```
{ "_id" : ObjectId("628b9b901544c1679d42a800"),
"Dispositivo" : {
    "Riparazione" : [{
        "Codice" : 59,
        "Descrizione" : "Drug much the old subject body.",
        "Prezzo" : "8 e",
        "Numero fattura" : "5510322061771",
        "Data" : "2010-08-05",
        "Seriale lotto" : [ "1043724530056" ],
        "Stato riparazione" : "in riparazione",
        "Sede di ritiro" : "sede2" }]
    }
}
```

Listing 4.12: Risultati

4.1.6 Visualizzazione riparazioni del dispositivo

Una volta identificato il dispositivo dell'utente mediante il codice andiamo a visualizzare le riparazioni di quest'ultimo.

```
db.users.find({
     "Dispositivo.Codice dispositivo" : "8898593453769"},{"Dispositivo.
    Riparazione.$":1}
     );
```

Listing 4.13: Visualizzazione riparazioni del dispositivo X

Nella parte seguente, vediamo il risultato di tale query.

```
{ "_id" : ObjectId("628b9b901544c1679d42a800"),
"Dispositivo" : {
    "Riparazione" : [
        { "Codice" : 33,
        "Descrizione" : "Hit couple consumer quickly check.",
        "Prezzo" : "47 e",
        "Numero fattura" : "7966800913434",
        "Data" : "2010-05-05",
        "Seriale lotto" : [ "6930212581964", "0769124747337",
    ↪ "6590333680894" ], "Stato riparazione" : "in riparazione",
        "Sede di ritiro" : "lab" },
        { "Codice" : 74,
        "Descrizione" : "Rock really them common education lead.",
        "Prezzo" : "54 e",
        "Numero fattura" : "3220887360685",
        "Data" : "1991-10-26",
        "Seriale lotto" : [ "1406060453708", "7984444871071",
    ↪ "6590333680894" ],
        "Stato riparazione" : "rientrata in garanzia", "Sede di ritiro"
    ↪ : "sede2" },
        { "Codice" : 41,
        "Descrizione" : "Near to town myself traditional.",
        "Prezzo" : "56 e",
```

```
        "Numero fattura" : "3633337522234",
        "Data" : "1999-02-22",
        "Seriale lotto" : [ "5378928999221", "4793041408008" ],
        "Stato riparazione" : "ritirata",
        "Sede di ritiro" : "sede2" },
        { "Codice" : 59,
        "Descrizione" : "Drug much the old subject body.",
        "Prezzo" : "8 e",
        "Numero fattura" : "5510322061771",
        "Data" : "2010-08-05",
        "Seriale lotto" : [ "1043724530056" ],
        "Stato riparazione" : "in riparazione", "Sede di ritiro" :
    ↪ "sede2"
        }
    ]}
}
```

Listing 4.14: Risultati

4.1.7 Visualizzazione riparazioni del cliente X

Qui andiamo a ricercare le riparazione effettuate da uno specifico cliente, identificato tramite il *codice fiscale*.

```
db.users.find(
    {"CF/PIVA":"CSXA27017031092200"},
    {"Dispositivo.Riparazione":1}
);
```

Listing 4.15: Visualizzazione riparazioni del cliente X

Nella parte seguente riportiamo il risultato di tale query.

```
{
  "_id": ObjectId("628900132d1e934d2895d437"),
  "Dispositivo": {
    "Riparazione": [
      {
        "Codice": 71,
        "Descrizione": "Prepare see service fine discuss water.",
        "Prezzo": "94 e",
        "Numero fattura": "1743026166686",
        "Data": "1973-09-27",
        "Seriale lotto": [
          "1951404145387",
          "1193827376980"
        ]
      },
      {
        "Codice": 82,
        "Descrizione": "Each report international remember.",
        "Prezzo": "89 e",
        "Numero fattura": "1253457662188",
        "Data": "2020-03-13",
        "Seriale lotto": [
          "6217628745952",
          "6217628745952",
```

```
          "1951404145387"
        ]
      },
      {
        "Codice": 91,
        "Descrizione": "May adult son economy sound suffer perform.",
        "Prezzo": "34 e",
        "Numero fattura": "9769502422260",
        "Data": "2000-09-11",
        "Seriale lotto": [
          "6765576288996",
          "0656287697190",
          "1744456813782",
          "2777497961838"
        ]
      },
      {
        "Codice": 92,
        "Descrizione": "Inside all later eat.",
        "Prezzo": "64 e",
        "Numero fattura": "8150506347191",
        "Data": "2014-08-04",
        "Seriale lotto": [
          "7934836390574",
          "7934836390574"
        ]
      }
    ]
  }
}
```

Listing 4.16: Risultati

4.1.8 Visualizza riparazioni effettuate con il lotto X

Qui andiamo ricercare le riparazioni effettuate con un particolare lotto ricambio, identificato tramite il seriale del lotto.

```
db.users.aggregate([
    {$unwind: '$Dispositivo'},
    {$unwind: '$Dispositivo.Riparazione'},
  {$match: {"Dispositivo.Riparazione.Seriale lotto": '1784841427576'}},
  {$project: {"Dispositivo.Riparazione":1}}
])
```

Listing 4.17: Visualizza riparazioni effettuate con il lotto X

Nel listato successivo, andiamo ad osservare il risultato della query sopra citata.

```
{ _id: ObjectId("628d318a2ef1dd8bc142462b"),
  Dispositivo:
   { Riparazione:
      { Codice: 4,
        Descrizione: 'Significant really direction new employee.',
        Prezzo: '79 e',
        'Numero fattura': '0629644644373',
        Data: '1989-05-24',
        'Seriale lotto': [ '1784841427576' ],
```

```
        'Stato riparazione': 'rientrata in garanzia',
        'Sede di ritiro': 'lab' } } }
{ _id: ObjectId("628d318b2ef1dd8bc1424632"),
  Dispositivo:
   { Riparazione:
      { Codice: 22,
        Descrizione: 'Air week Mrs activity player during hope.',
        Prezzo: '84 €',
        'Numero fattura': '5095910481648',
        Data: '1991-01-14',
        'Seriale lotto': [ '1784841427576', '1923143500475',
    ↪ '1409427350979' ],
        'Stato riparazione': 'in riparazione',
        'Sede di ritiro': 'sede2' } } }
```

Listing 4.18: Risultati

Come si può osservare dai risultati vengono riportate le informazioni delle riparazioni effettuate con il particolare lotto ricambio specificato (identificato tramite il seriale).

4.2 Operazioni Statistiche

4.2.1 Sede preferita per il ritiro

Qui andiamo a verificare quale è la preferenza di ritiro dei vari clienti, contando il numero di riparazioni ritirate presso ciascuna delle possibili sedi di ritiro. Si noti che i valori del parametro "Sedi di ritiro" possono essere "lab" o "sede2".

Nella query seguente si conta il numero di riparazioni che hanno il valore dell'attributo "Sede di ritiro" pari a "lab" o pari a "sede2". Di seguito si riporta la query e il risultato della stessa.

```
db.users.aggregate([
    {$unwind: '$Dispositivo'},
    {$unwind: '$Dispositivo.Riparazione'},
  {$group : { "_id" : "$Dispositivo.Riparazione.Sede di ritiro", count : {"$sum"
     : 1}}},
  {$sort: {count:-1}}
])



{ _id: 'lab', count: 24 }
{ _id: 'sede2', count: 22 }
```

Listing 4.19: Visualizzazione preferenza di ritiro utenti(Lab)

4.2.2 Visualizzazione Numero di Riparazioni per tipologia di dispositivo

Nella query successiva si conta il numero di riparazioni effettuate per le differenti tipologie di dispositivi.

```

db.users.aggregate([
    {$unwind: '$Dispositivo'},
    {$unwind: '$Dispositivo.Riparazione'},
  {$group: { "_id" : "$Dispositivo.Tipologia", count : {"$sum" : 1}}},
  {$sort: {count:-1}}
```

```
7 ])
```

Listing 4.20: Visualizzazione numero di riparazioni per tipologia di dispositivo

Di seguito si riporta l'output della query precedente.

```
1 { _id: 'Smartphone', count: 18 }
2 { _id: 'Laptop', count: 16 }
3 { _id: 'Tablet', count: 12 }
```

Listing 4.21: Risultati

4.2.3 Date

Nella query seguente, è possibile vedere l'implementazione per la visualizzazione delle riparazioni effettuate in una specifica data.

```
1 db.users.aggregate([
2     {$unwind: '$Dispositivo'},
3     {$unwind: '$Dispositivo.Riparazione'},
4     {$match: {"Dispositivo.Riparazione.Data": '2001-08-16'}},
5     {$project: {"Dispositivo.Riparazione":1}}
6 ])
```

Listing 4.22: Visualizzazione Riparazioni effettuate in una specifica data

```
1  { _id: ObjectId("628dfbe2861ad4dfe01fb4c8"),
2    Dispositivo:
3     { Riparazione:
4        { Codice: 60,
5          Descrizione: 'Rather cell than worker.',
6          Prezzo: '67 e',
7          'Numero fattura': '8062885021787',
8          Data: '2001-08-16',
9          'Seriale lotto': [ '8405089907035', '4106482894611' ],
10         'Stato riparazione': 'completata',
11         'Sede di ritiro': 'lab' } } }
```

Listing 4.23: Risultati

Nella query successiva si restituisce invece il conteggio del numero di riparazioni effettuate nella data specificata.

```
1 db.users.aggregate([
2 { "$unwind" : { "path" : "$Dispositivo" }},
3 { "$unwind" : { "path" : "$Dispositivo.Riparazione" }},
4 { "$match" : {"Dispositivo.Riparazione.Data": "2001-06-02"}},
5 { "$group" : { "_id" : "$Dispositivo.Riparazione.Data" , "count" : { "$sum": 1 }
      }}
6 ])
```

Listing 4.24: Visualizzazione Riparazioni effettuate in una certa data

```
1 {'_id': '2001-06-02', 'count': 3}
```

Listing 4.25: Risultato

Di seguito si riporta l'implementazione in Python della query vista precedentemente. Si è deciso di implementarla in python così da coprire anche tale aspetto, reso possibile dal pacchetto Python pymongo.

```
1
2 from pymongo import MongoClient
3
4 client = MongoClient('mongodb://localhost:27017')
5 db = client.NGDB_proj
6
7
8 match = { "$match" : {"Dispositivo.Riparazione.Data": "2001-06-02"}}
9 unwind = { "$unwind" : { "path" : "$Dispositivo" }}
10 unwind2 = { "$unwind" : { "path" : "$Dispositivo.Riparazione" }}
11 group = { "$group" : { "_id" : "$Dispositivo.Riparazione.Data" , "count" : { "
     $sum": 1 } }}
12 pipeline = [unwind, unwind2, match, match , group ]
13 results = db.users.aggregate(pipeline)
14
15 for x in results:
16     print(x)
```

Listing 4.26: Visualizzazione riparazione effettuate in una determinata data

Eseguendo lo script precedente il risultato che si ottiene è quello riportato di seguito.

```
1 {'_id': '2001-06-02', 'count': 3}
```

Listing 4.27: Risultato

Capitolo 5

Conclusioni e Sviluppi Futuri

In conclusione possiamo osservare come gli obiettivi prefissati di progettazione ed implementazione di un sistema NoSQL sono stati raggiunti.
Come detto, dal progetto iniziale è stato necessario realizzare delle modifiche che rendessero il progetto "interessante" da analizzare in un'ottica NoSQL e raggiungere risultati significativi.
Oltre alle fasi di progettazione è stata realizzata l'implementazione dei risultati ottenuti, così da testare in pratica ciò che si è precedentemente progettato. Nella fase di implementazione si sono utilizzati strumenti e tecniche viste durante le esercitazioni del corso, così da applicare ciò che si è appreso durante le lezioni.
Come sviluppo futuro di tale lavoro si potrebbe pensare di realizzare la progettazione e l'implementazione degli elementi secondari e meno importanti che si sono tralasciati. Così facendo si possono confrontare i risultati ottenuti nei due casi.
Un ulteriore sviluppo potrebbe essere quello di utilizzare una diversa tipologia di *store*. Come detto, in questo lavoro di tesi si è utilizzato il *document store* MongoDB. Utilizzare una differente tipologia di database renderebbe possibile un confronto fra i risultati ottenuti, evidenziando i *PROs* e *CONs* delle diverse alternative proposte.

Capitolo 6

Appendice

Per una maggiore completezza si riporta, nel listato successivo, lo script Python utilizzato per la generazione e l'importazione dei dati all'interno di MongoDB.

```
from pymongo import MongoClient
import time
from faker import Faker
from random import randrange
import random

# PARAMS
NUMBER_OF_USERS = 10
NUMBER_OF_RICAMBI = 10
MAX_NUMBER_OF_DEVICES = 5
MAX_NUMBER_OF_REPARATIONS = 5
MAX_NUMBER_OF_LOTS = 5
MAX_NUMBER_OF_LOTS_PER_REPARATIONS = 5
MAX_LOTS_DISP = 5
LOTS_IDS = []
SEDI_RIT = ['lab', 'sede2']
STATI_RIP = ['presa in carico', 'in riparazione', 'completata', 'ritirata', '
    rientrata in garanzia']
DEV_TYPES = ['Smartphone', 'Laptop', 'Tablet']
BRANDS = ['Apple', 'Asus', 'Lenovo', 'Samsung', 'Huawei']
MODELS = {
    'Apple': {
        'Smartphone': ['iPhone 11', 'iPhone 13 Mini', 'iPhone XS'],
        'Laptop': ['MacBook Pro', 'iMac', 'MacBook Air'],
        'Tablet': ['iPad Pro 10,5"', 'iPad Air', 'iPad Pro 12,9"'],
    },
    'Asus': {
        'Smartphone': ['Zenfone 8', 'ROG Phone'],
        'Laptop': ['Zenbook', 'Vivobook', 'Zenbook Pro'],
        'Tablet': ['Tablet 10.4"', 'Tablet 12"'],
    },
    'Lenovo': {
        'Smartphone': ['moto g22', 'motorola edge 30'],
        'Laptop': ['IdeaPad 3', 'IdeaPad 3i', 'ThinkPad L14'],
        'Tablet': ['Tab M10', 'Tab P12 Pro', 'Yoga Tab'],
    },
    'Samsung': {
        'Smartphone': ['S22', 'A14'],
        'Laptop': ['Galaxy Book', 'Galaxy Book 2'],
        'Tablet': ['Galaxy Tab S6', 'Galaxy Tab S6 Lite'],
    },
    'Huawei': {
```

```
42          'Smartphone': ['P30', 'P40 Pro', 'P40'],
43          'Laptop': ['Matebook 14', 'Matebook 15', 'Matebook 14 Pro'],
44          'Tablet': ['MatePad 10.4', 'MatePad Pro 12.6', 'MatePad 11'],
45      }
46  }
47  GRADES = ['A++', 'A+', 'A', 'B', 'C', 'D']
48  RIC_NAMES = ['Display', 'Speaker', 'Microfono', 'Tastiera', 'Batteria', '
        Motherboard']
49
50  client = MongoClient('mongodb://localhost:27017')
51
52  db = client.NGDB_proj
53
54
55  def create_users(fake):
56      for x in range(NUMBER_OF_USERS):
57          genCF = fake.bban()
58          genName = fake.first_name()
59          genSurname = fake.last_name()
60          genPhone = fake.phone_number()
61          genAddrNumber = fake.building_number()
62          genAddrCity = fake.city()
63          genAddrProv = fake.country_code()
64          genAddrPostCode = fake.postcode()
65          genAddrStreetName = fake.street_name()
66          # rand_dev_type = DEV_TYPES[randrange(len(DEV_TYPES))]
67
68          objToInsert = {
69              'CF/PIVA': genCF,
70              'Nome': genName,
71              'Cognome': genSurname,
72              'Telefono': genPhone,
73              'Indirizzo': {
74                  'Via': genAddrStreetName,
75                  'Civico': genAddrNumber,
76                  'Città': genAddrCity,
77                  'CAP': genAddrPostCode,
78                  'Provincia': genAddrProv
79              },
80              'Dispositivo': [], # TODO
81          }
82
83          # generate a random number of devices for each user
84          NUMBER_OF_DEVICES = randrange(MAX_NUMBER_OF_DEVICES)
85          for _ in range(NUMBER_OF_DEVICES):
86              # generate a random device
87              rand_dev_type = DEV_TYPES[randrange(len(DEV_TYPES))]
88              rand_dev_brand = BRANDS[randrange(len(BRANDS))]
89              rand_dev_model = MODELS[rand_dev_brand][rand_dev_type][randrange(len
        (MODELS[rand_dev_brand][rand_dev_type]))]
90              rand_dev_SN = fake.ean()
91
92              objToInsert['Dispositivo'].append({
93                  'Tipologia': rand_dev_type,
94                  'Marca': rand_dev_brand,
95                  'Modello': rand_dev_model,
96                  'Codice dispositivo': rand_dev_SN,
97                  'Riparazione': []
98              })
99
100             # generate a random number of reparations
```

```
101             NUMBER_OF_REPARATIONS = randrange(1, MAX_NUMBER_OF_REPARATIONS)
102
103             for _ in range(NUMBER_OF_REPARATIONS):
104                 # generate a random reparations
105                 rand_rep_id = randrange(100) + 1
106                 rand_rep_date = fake.date(pattern='%Y-%m-%d')
107                 rand_price = randrange(100)
108                 rand_bill_nr = fake.ean()
109                 rand_state = STATI_RIP[randrange(len(STATI_RIP))]
110                 rand_sede = SEDI_RIT[randrange(len(SEDI_RIT))]
111
112                 objToInsert['Dispositivo'][-1]['Riparazione'].append({
113                     'Codice': rand_rep_id,
114                     'Descrizione': fake.paragraph(nb_sentences=1),
115                     'Prezzo': f'{rand_price} €',
116                     'Numero fattura': f'{rand_bill_nr}',
117                     'Data': rand_rep_date,
118                     'Seriale lotto': [],
119                     'Stato riparazione': rand_state,
120                     'Sede di ritiro': rand_sede
121                 })
122
123                 NUMBER_OF_LOTS = randrange(1, MAX_NUMBER_OF_LOTS_PER_REPARATIONS
    )
124                 for _ in range(NUMBER_OF_LOTS):
125                     idx_lot = randrange(len(LOTS_IDS))
126                     objToInsert['Dispositivo'][-1]['Riparazione'][-1]['Seriale
    lotto'].append(LOTS_IDS[idx_lot])
127
128         result = db.users.insert_one(objToInsert)
129         print('id: ' + str(result.inserted_id) + ' name: ' + genName)
130         time.sleep(0.2)
131
132
133 def create_ricambio(fake):
134
135     for x in range(NUMBER_OF_RICAMBI):
136         # genera una classe di ricambio casuale
137         rand_dev_brand = BRANDS[randrange(len(BRANDS))]
138         rand_ric_name = RIC_NAMES[randrange(len(RIC_NAMES))]
139         rand_dev_type = DEV_TYPES[randrange(len(DEV_TYPES))]
140         rand_dev_model = MODELS[rand_dev_brand][rand_dev_type][randrange(len(
    MODELS[rand_dev_brand][rand_dev_type]))]
141         rand_note = ''
142         if bool(random.getrandbits(1)):
143             rand_note = fake.paragraph(nb_sentences=1)
144
145         objToInsert = {'Nome': rand_ric_name,
146                        'Marca dispositivo': rand_dev_brand,
147                        'Modello dispositivo': rand_dev_model,
148                        'Tipologia dispositivo': rand_dev_type,
149                        'Note aggiuntive': rand_note,
150                        'Lotto Ricambio': []}
151
152         # generate a random number of lots for each ricambio
153         NUMBER_OF_LOTS = randrange(MAX_NUMBER_OF_LOTS)
154         for _ in range(NUMBER_OF_LOTS):
155             rand_lot_id = fake.ean()
156             LOTS_IDS.append(rand_lot_id)
157             rand_grade = GRADES[randrange(len(GRADES))]
158             # generate a random supplier
```

```
159             genCF = fake.bban()
160             genName = fake.first_name()
161             genSurname = fake.last_name()
162             genAddrNumber = fake.building_number()
163             genAddrCity = fake.city()
164             genAddrProv = fake.country_code()
165             genAddrPostCode = fake.postcode()
166             genAddrStreetName = fake.street_name()
167             genPhone = fake.phone_number()
168             genWebSite = fake.domain_name()
169             genShipDays = randrange(1, 7)
170             genShipPrice = randrange(15)
171 
172             objToInsert['Lotto Ricambio'].append({
173                 'Seriale': rand_lot_id,
174                 'Grado': rand_grade,
175                 'Data': fake.date(pattern='%Y-%m-%d'),
176                 'Disponibilità': randrange(0, MAX_LOTS_DISP),
177                 'Fornitore': {
178                     'CF/PIVA': genCF,
179                     'Ragione Sociale': genName + ' ' + genSurname + ' s.r.l',
180                     'Nome': genName,
181                     'Cognome': genSurname,
182                     'Telefono': genPhone,
183                     'Tipologia': rand_dev_type,
184                     'Sito Web': genWebSite,
185                     'Costi di spedizione': f'{genShipPrice} e',
186                     'Tempi di spedizione': f'{genShipDays} gg',
187                     'Indirizzo': {
188                         'Via': genAddrStreetName,
189                         'Civico': genAddrNumber,
190                         'Città': genAddrCity,
191                         'CAP': genAddrPostCode,
192                         'Provincia': genAddrProv
193                     }
194                 }
195             })
196 
197         result = db.ricambi.insert_one(objToInsert)
198 
199         print('id: ' + str(result.inserted_id))
200         time.sleep(0.2)
201 
202 
203 if __name__ == '__main__':
204     fake = Faker()
205     create_ricambio(fake)
206     create_users(fake)
```

Listing 6.1: Script Python