UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea
in Ingegneria Informatica

# Relazione Finale
# Titolo 1
# Titolo 2
# Titolo 3

**Relatore:** Chiar.mo Prof. Melchiori Michele

Laureando:
Rizzo Matteo
Matricola n. 727499

---

Anno Accademico 2021/2022

# SOMMARIO

# RINGRAZIAMENTI

# Indice

**1 DataBase NoSQL** **3**
1.1 section 1.1 . . . . . 3
1.2 nuova sezione . . . . . 3

**2 Progettazione Concettuale/Logica** **5**
2.1 Modellazione NoAM → NoSQL abstract model . . . . . 5
2.1.1 Modellazione Concettuale e design degli Aggregati . . . . . 5
2.1.2 Partizionamento degli Aggregati e modellazione NoSQL . . . . 6
2.1.3 Implementazione . . . . . 8

**3 Approfondimento Redis** **9**

**4 Interrogazioni SQL → *Key-Value*** **11**

**5 *ACID*** **13**

**6 API Redis** **15**

**7 Caso Concreto** **17**

**BIBLIOGRAFIA** **19**

# INTRODUZIONE

Analisi database noSQL chiave-valore, in particolare *REDIS* progettazione concettuale/logica database chiave-valore

# Capitolo 1

# DataBase NoSQL

Panoramica su database nosql

- documentali
- chiave-valore
- ...

## 1.1 section 1.1

questa é una section 2 ...

## 1.2 nuova sezione

# Capitolo 2

# Progettazione Concettuale/Logica

## 2.1 Modellazione NoAM → NoSQL abstract model

La metodologia NoAM é composta da:

- Modellazione Concettuale e design degli Aggregati
- Partizionamento degli Aggregati e modellazione NoSQL
- Implementazione

### 2.1.1 Modellazione Concettuale e design degli Aggregati

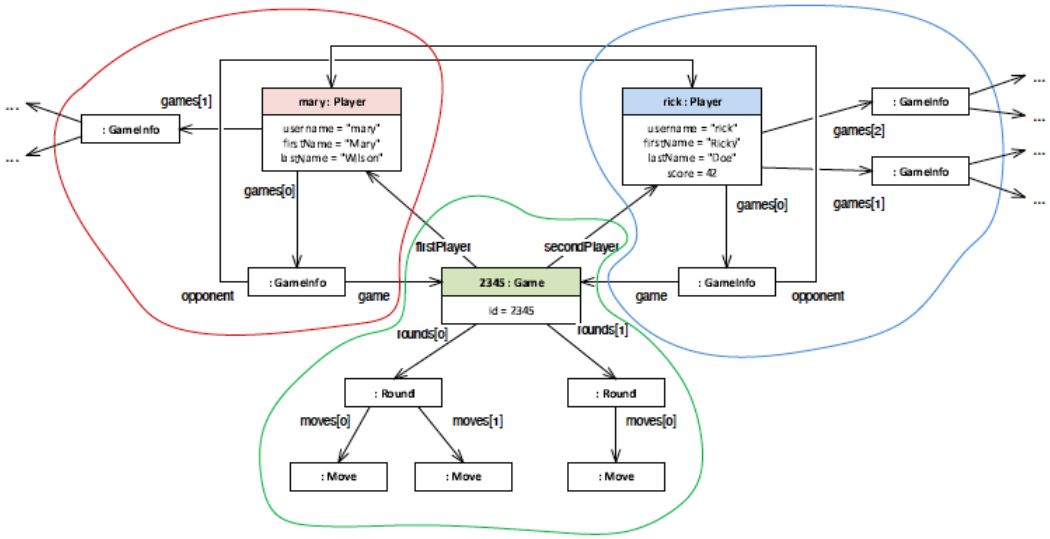
Riguarda la vera e propria progettazione del modello di dominio, e comporta l'identificazione delle diverse classi di aggregati necessari in un'applicazione. Sono possibili diversi approcci per identificare classi di aggregati per una particolare applicazione. L'approccio Domain-Driven Design($DDD$), attraverso il quale viene generato un diagramma UML delle classi, é guidato dai casi d'uso, ovvero dai requisiti funzionali, e da esigenze di scalabilitá e coerenza all'interno dell'aggregato. Si procede nel modo seguente:

- I dati persistenti di un'applicazione sono modellati in termini di entitá, oggetti di valore e relazioni. Un'entitá é un oggetto persistente che ha un'esistenza indipendente ed é caratterizzata da un'identificatore univoco, mentre un oggetto di valore é caratterizzato appunto da un suo valore senza un proprio identificatore
- Entitá e oggetti di valore vengono raggruppati in *aggregati*. Un aggregato ha un'entitá come radice e puó contenere molti oggetti di valore.

A causa delle loro caratteristiche, la progettazione degli aggregati comporta un compromesso per quanto riguarda la loro granularitá. infatti:

- Gli aggregati dovrebbero essere abbastanza grandi per poter includere tutti i dati coinvolti da certi vincoli di integritá.
- Gli aggregati dovrebbero essere i piú piccoli possibile, in quanto dimensioni ridotte consentono di soddisfare requisiti di prestazioni e scalabilitá.

Preso come esempio un dominio in cui vanno salvati in modo persistente dati su giocatori e giochi




Nella figura sopra riportata l'oggetto con lo scomparto superiore colorato é un'entitá, altrimenti é un oggetto di valore. La linea chiusa denota il confine di un aggregato. Pertanto avremo due classi aggregate principali: Player e Game.

### 2.1.2 Partizionamento degli Aggregati e modellazione NoSQL

In questa fase viene utilizzato *NoAM* come modello intermedio tra gli aggregati e i database NoSQL, quindi potrebbe essere visto come un equivalente della *progettazione concettuale* fatta nei database relazionali. Il modello NoAM → modello di dati astratti, ha il compito di sfruttare i punti in comune dei vari modelli di dati, ma introduce anche astrazioni per bilanciare le variazioni che vi sono tra diversi modelli di NoSQL. Vi sono due nozioni distinte di *unitá* di accesso ai dati:

- `blocco`, unitá di dimensione maggiore, ha massima consistenza

- `entry`, unitá di dimensione minore, che permetto l'accesso ai dati

Con riferimento in particolare ai database chiave-valore una `entry` corrisponde a una coppia chiave-valore, mentre un `blocco` corrisponde a un gruppo di coppie chiave-valore correlate tra loro. Modello generale:

- Un `database` é un insieme di `collections`. Ogni `collection` ha un nome distinto

- Una `collection` é un insieme di `blocchi`, ogni `blocco` all'interno della `collection` é identificato da una chiave di blocco, che deve essere univoca

- Un `blocco` é un insieme non vuoto di `entries`, ogni `entry` é composta da una coppia chiave-valore, la quale é univoca all'interno del blocco, il valore puó essere anche complesso.

Per effettuare il passaggio da aggregati a modellazione NoSQL:

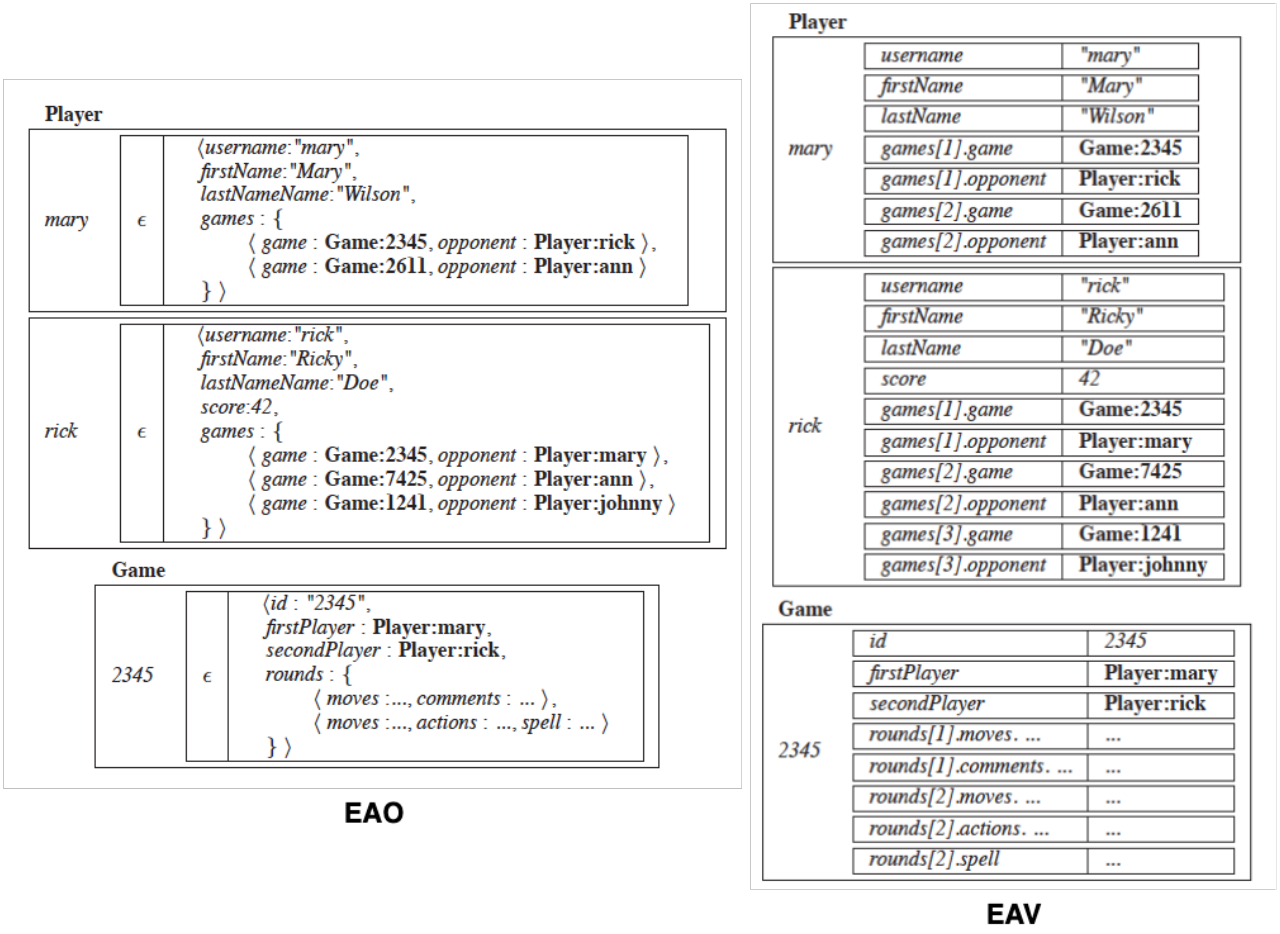
- Ogni classe di aggregati viene rappresentata da una `collection`

- Ogni singolo aggregato viene rappresentato da un `blocco`

Vi sono due modalitá principali per rappresentare gli aggregati:

- *Entry per Aggregate Object (EAO)*: Rappresenta ogni aggregato utilizzando una singola `entry`, la chiave della `entry` é vuota, il valore contiene l'intero aggregato

- *Entry per Atomic Value (EAV)*: Rappresenta ogni aggregato per mezzo di piú `entry`, le chiavi di ogni `entry` devono rappresentare il percorso per accedere al valore di un certo componente(quindi possono esserci anche chiavi con nomi strutturati a piú livelli), i valori di ogni `entry` sono atomici




Nella figura *EAO* si ha per ogni `blocco` *Player* o *Game* una singola `entry` con chiave vuota, valore strutturato in modo complesso
Nella figura *EAV* si ha per ogni `blocco` piú `entry` in modo da avere valori atomici al suo interno

### 2.1.3 Implementazione

# Capitolo 3

# Approfondimento Redis

Parlare di Redis in generale vantaggi/svantaggi perché utilizzarlo strutture dati String Map Set Json ...

# Capitolo 4

# Interrogazioni SQL $\rightarrow$ *Key-Value*

parlare del tipo di interrogazioni che vengono fatte, c'é un vero e proprio linguaggio strutturato come SQL?

# Capitolo 5

# *ACID*

Parlare delle proprietá *ACID* per quanto riguarda questo tipo di database Redis quali di proprietá dispone? É possibile decidere se avere una certa proprietá o meno ...

# Capitolo 6

# API Redis

implementazione librerie Redis in linguaggi di programmazione Parlare di Jedis, libreria Java. Riportare esempi di codice che implementa DataBase

# Capitolo 7

# Caso Concreto

sviluppare un caso concreto partendo dalla sua progettazione. parto da progettazione concettuale, passo a quella logica Bisognerá strutturare in modo intelligente le chiavi key:id ...

Far vedere delle possibili interrogazioni, magari in codice Java oppure con CLI.

# BIBLIOGRAFIA

Bibliografia