MongoDB - Udemy

Jacopo De Angelis

24 dicembre 2019

Indice

Capitolo 1

Perchè un database noSQL?

I database noSQL nascono per esigenze particolari con tre obiettivi principali:

- scalabilità: un sistema più comodo per i big data
- facilità dello sviluppo
- rappresentazione intuitiva: dati semi-strutturati, non strutturati, strutture dati complesse, polimorfismo

Scalabilità:

- verticale: potenziamento della macchina singola
- orizzontale: moltiplicazione delle macchine a disposizione

Per quanto molto vantaggiosa, la scalabilità orizzontale porta il problema di passaggio delle informazioni tra i vari server. MongoDB rinuncia a joins e transactions.

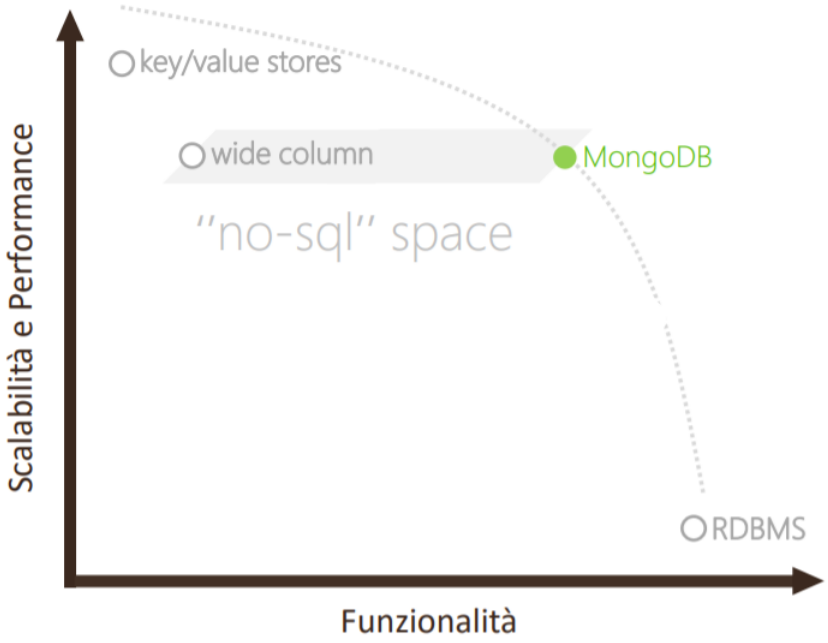

Figura 1.1: Rapporto funzionalità/scalabilità

Capitolo 2

Funzionalità

2.1 Join e transactions

Ha rinunciato a queste due funzioni in nome della scalabilità, infatti come si poterebbero ottimizzare le performance su più server?

> Copiando le tabelle ovunque? Non sarebbe vera scalabilità
>
> Dividendole su più server? Auguri nel caso si voglia accedere ad una tabella di join su più server

Si usa l'approccio "document-oriented", ovvero un'intuitiva rappresentazione dei dati (in questo caso i documenti sono uguali agli oggetti della programmazione orientata agli oggetti, sono contenitori di informazioni). Si usa quindi la codifica JSON[1]

2.1.1 JSON

Un oggetto è una serie non ordinata di nomi/valori. Un oggetto inizia con parentesi graffa sinistra e finisce con parentesi graffa destra. Ogni nome è seguito da :due punti e la coppia di nome/valore sono separata da ,virgola.

[1] JavaScript Object Notation

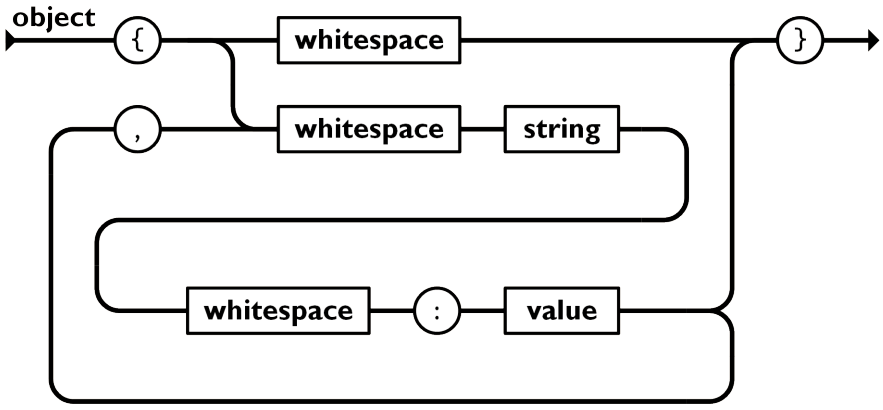

Figura 2.1: JSON: parser oggetto

Un array è una raccolta ordinata di valori. Un array comincia con [parentesi quadra sinistra e finisce con]parentesi quadra destra. I valori sono separati da ,virgola.

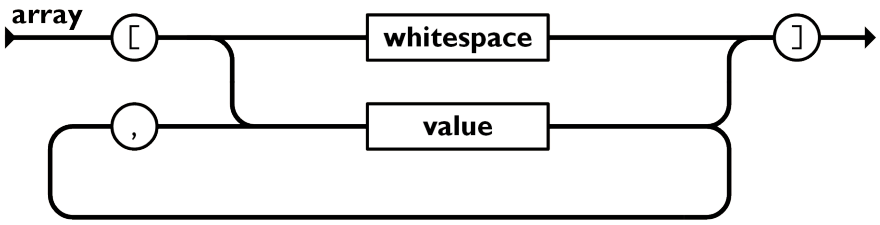

Figura 2.2: JSON: parser array

Un valore può essere una stringa tra virgolette, o un numero, o vero true o falso false o nullo null, o un oggetto o un array. Queste strutture possono essere annidate.

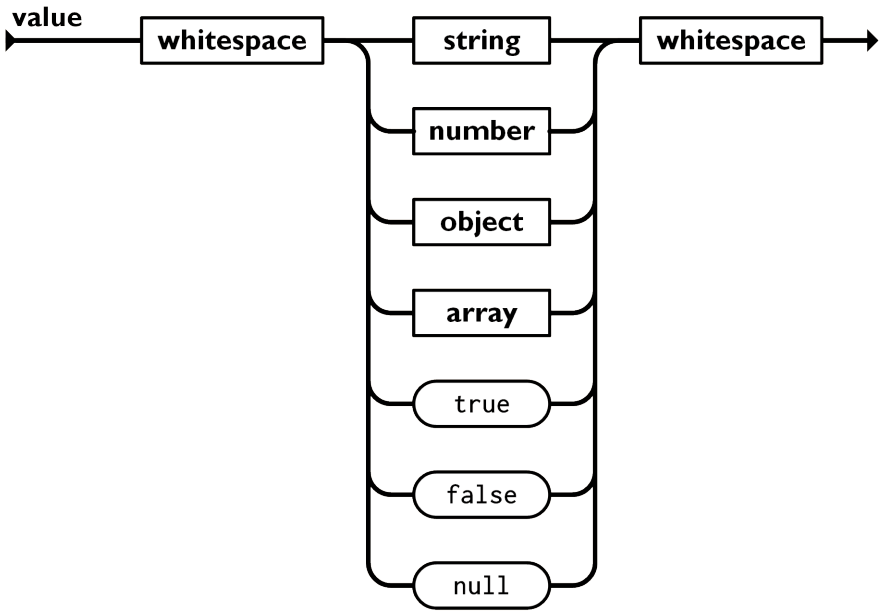

Figura 2.3: JSON: parser valore

Una stringa è una raccolta di zero o più caratteri Unicode, tra virgolette; per le sequenze di escape utilizza la barra rovesciata. Un singolo carattere è rappresentato come una stringa di caratteri di lunghezza uno. Una stringa è molto simile ad una stringa C o Java.

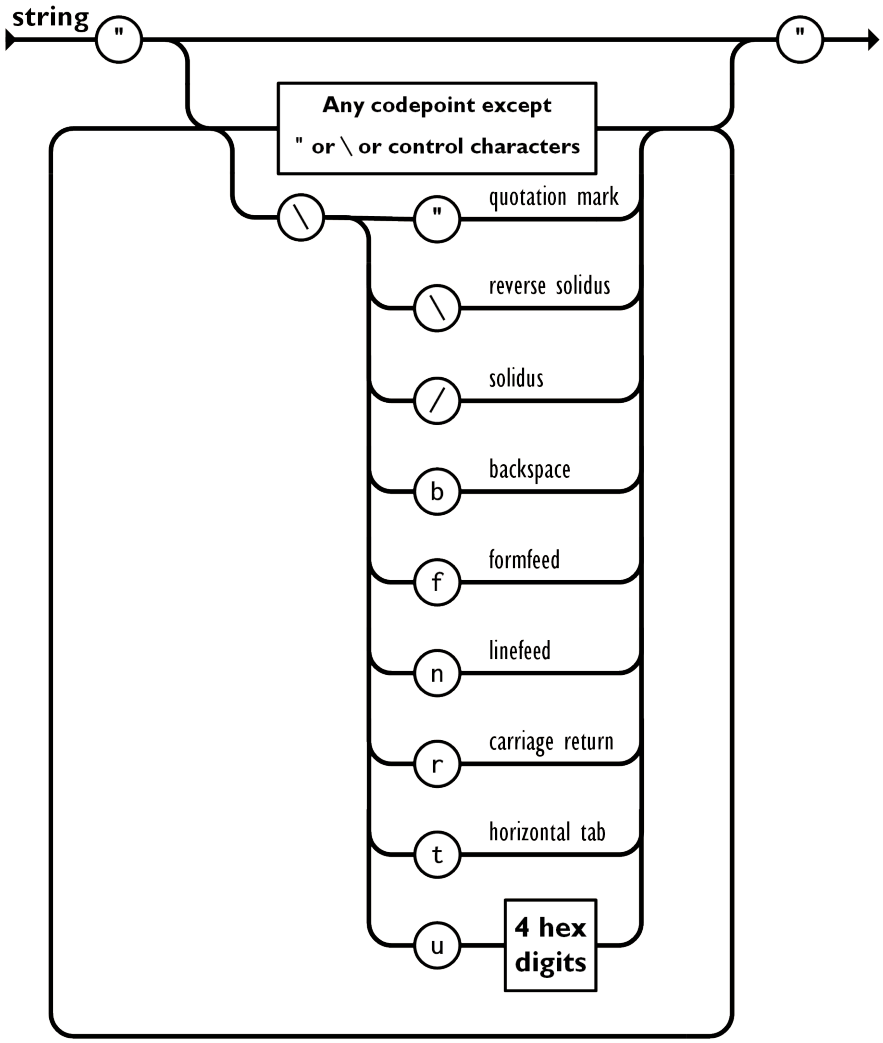

Figura 2.4: JSON: parser stringa

Un numero è molto simile ad un numero C o Java, a parte il fatto che i formati ottali e esadecimali non sono utilizzati.

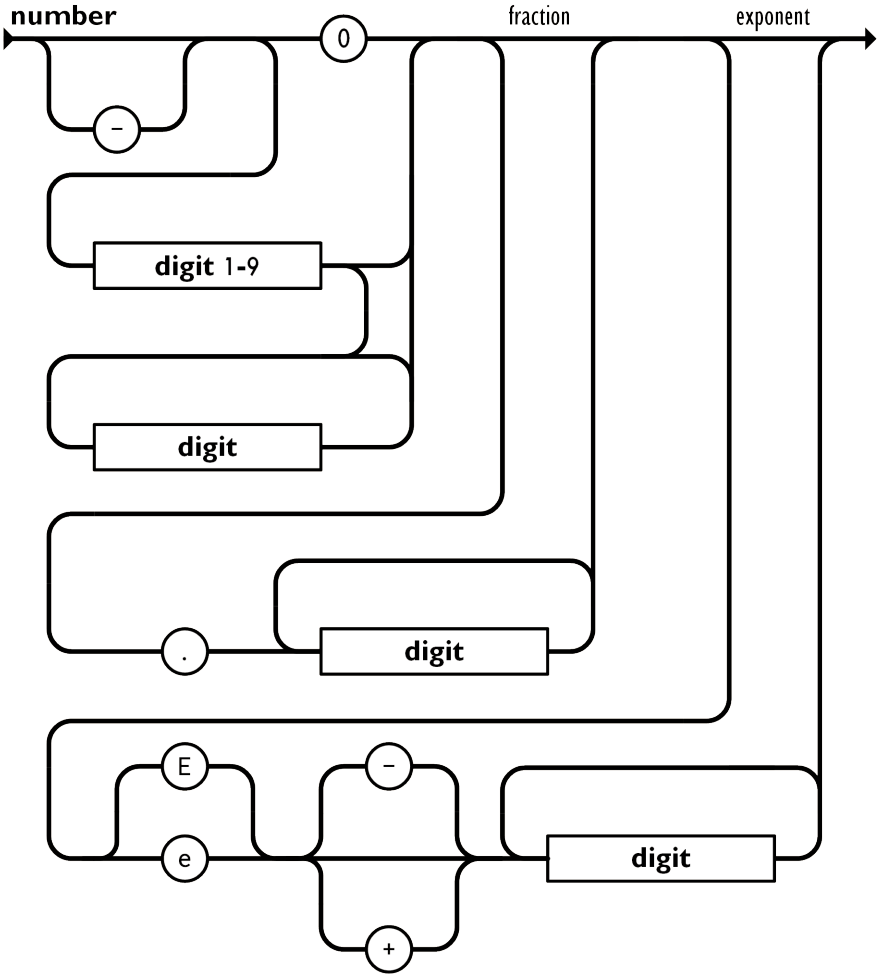

Figura 2.5: JSON: parser numero

I caratteri di spaziatura possono essere inseriti in mezzo a qualsiasi coppia di token. A parte alcuni dettagli di codifica, questo descrive totalmente il linguaggio.

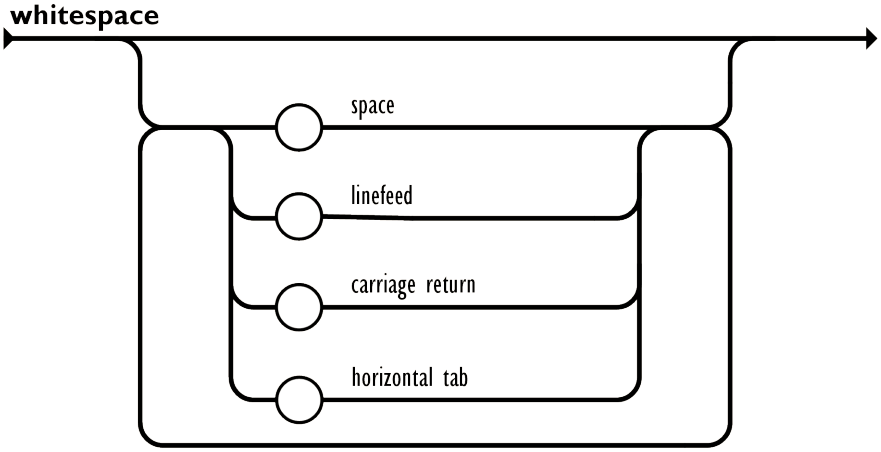

Figura 2.6: JSON: parser spaziatura

Esempio di oggetto JSON:

```
{"menu": {
    "header": "SVG Viewer",
    "items": [
        {"id": "Open"},
        {"id": "OpenNew", "label": "Open New"},
        null,
        {"id": "ZoomIn", "label": "Zoom In"},
        {"id": "ZoomOut", "label": "Zoom Out"},
        {"id": "OriginalView", "label": "Original View"},
        null,
        {"id": "Quality"},
        {"id": "Pause"},
        {"id": "Mute"},
        null,
        {"id": "Find", "label": "Find..."},
        {"id": "FindAgain", "label": "Find Again"},
        {"id": "Copy"},
        {"id": "CopyAgain", "label": "Copy Again"},
        {"id": "CopySVG", "label": "Copy SVG"},
        {"id": "ViewSVG", "label": "View SVG"},
        {"id": "ViewSource", "label": "View Source"},
        {"id": "SaveAs", "label": "Save As"},
        null,
        {"id": "Help"},
        {"id": "About", "label": "About Adobe CVG Viewer..."}
    ]
}}
```