DMIF, Università di Udine

Tecnologie Digitali per il Cibo e la Ristorazione

Basi di Dati Non Relazionali - NoSQL

Andrea Brunello
andrea.brunello@uniud.it

A.A. 2021–2022

Outline

Cosa si intende per NoSQL?

Il termine NoSQL (non solo SQL) si riferisce a sistemi di basi di dati che non sono relazionali, piuttosto che dare una definizione esplicita di cosa comprendano tali sistemi.

Una possibile (vaga) definizione: *"Next Generation DBs mostly addressing some of the points: being non-relational, distributed, open source and horizontally scalable"*.

Le soluzioni NoSQL sono piuttosto eterogenee per quanto riguarda il loro modello dei dati, le loro funzionalità e le loro soluzioni architetturali.

I sistemi NoSQL sono stati proposti a partire dal 2009, con l'intento di rispondere alle sfide emerse riguardo al trattamento dei dati.

L'ascesa del NoSQL - 1

Inizialmente, la stragrande maggioranza delle applicazioni enterprise erano supportate da basi di dati relazionali (MySQL, PostgreSQL, etc).

Nel corso degli anni i dati sono aumentati di volume, hanno iniziato a cambiare più rapidamente ed in generale ad essere strutturalmente più vari rispetto a quelli comunemente gestiti con i tradizionali RDBMS.

A mano a mano che il **Volume** dei dati è aumentato, è aumentato anche il tempo di esecuzione delle query, come conseguenza della dimensione delle tabelle coinvolte e dell'incremento del numero di operazioni di join (*join pain*).

L'ascesa del NoSQL - 2

La **Velocità** di generazione dei dati è raramente una metrica statica. Cambiamenti interni ed esterni a un sistema e al contesto in cui esso opera possono avere un notevole impatto sulla velocità dei dati.

La velocità variabile, unita a un volume elevato, richiedono soluzioni per la memorizzazione dei dati in grado di gestire livelli stabilmente elevati di carichi di lettura e scrittura, oltre che picchi.

I dati prodotti nel mondo reale sono molto più vari quelli tipicamente gestiti nel mondo relazionale.

La **Varietà** può essere definita come il grado di "regolarità" dei dati: strutturati, semi-strutturati, non strutturati.

Vantaggi del NoSQL

Scalabilità flessibile:

- RDBMS scalano **verticalmente**: maggior carico ⇒ server più potente
- NoSQL scalano **orizzontalmente**: dati distribuiti su più nodi, semplice aggiungere/rimuovere nodi

DBA Specialist:

- RDBMS richiede personale altamente addestrato per la sua implementazione e gestione
- Gestire un sistema NoSQL è meno complesso: modelli dei dati semplici, funzionalità di distribuzione automatiche

Big Data:

- Enorme aumento dei dati, RDBMS: capacità al limite
- Soluzioni NoSQL progettate appositamente per gestire grandi quantità di dati

Svantaggi del NoSQL

Diversi vincoli classici dei RDBMS non sono supportati (ad es. chiavi esterne).

Se non gestita correttamente, l'assenza di una struttura fissa dei dati può diventare un problema.

Il processo di progettazione non è così lineare e consolidato come quello del modello relazionale.

Mancanza del linguaggio SQL (sebbene vi siano alcuni linguaggi ispirati a SQL).

Acido VS basico

Molte persone che incontrano NoSQL hanno già familiarità con i database relazionali.

Oltre alle chiare differenze nei modelli dei dati, anche i modelli di consistenza utilizzati dai sistemi NoSQL possono essere molto diversi da quelli impiegati dai database relazionali.

I database NoSQL tipicamente utilizzano diversi modelli di consistenza per supportare le eterogeneità in volume, velocità e varietà di dati discussi in precedenza.

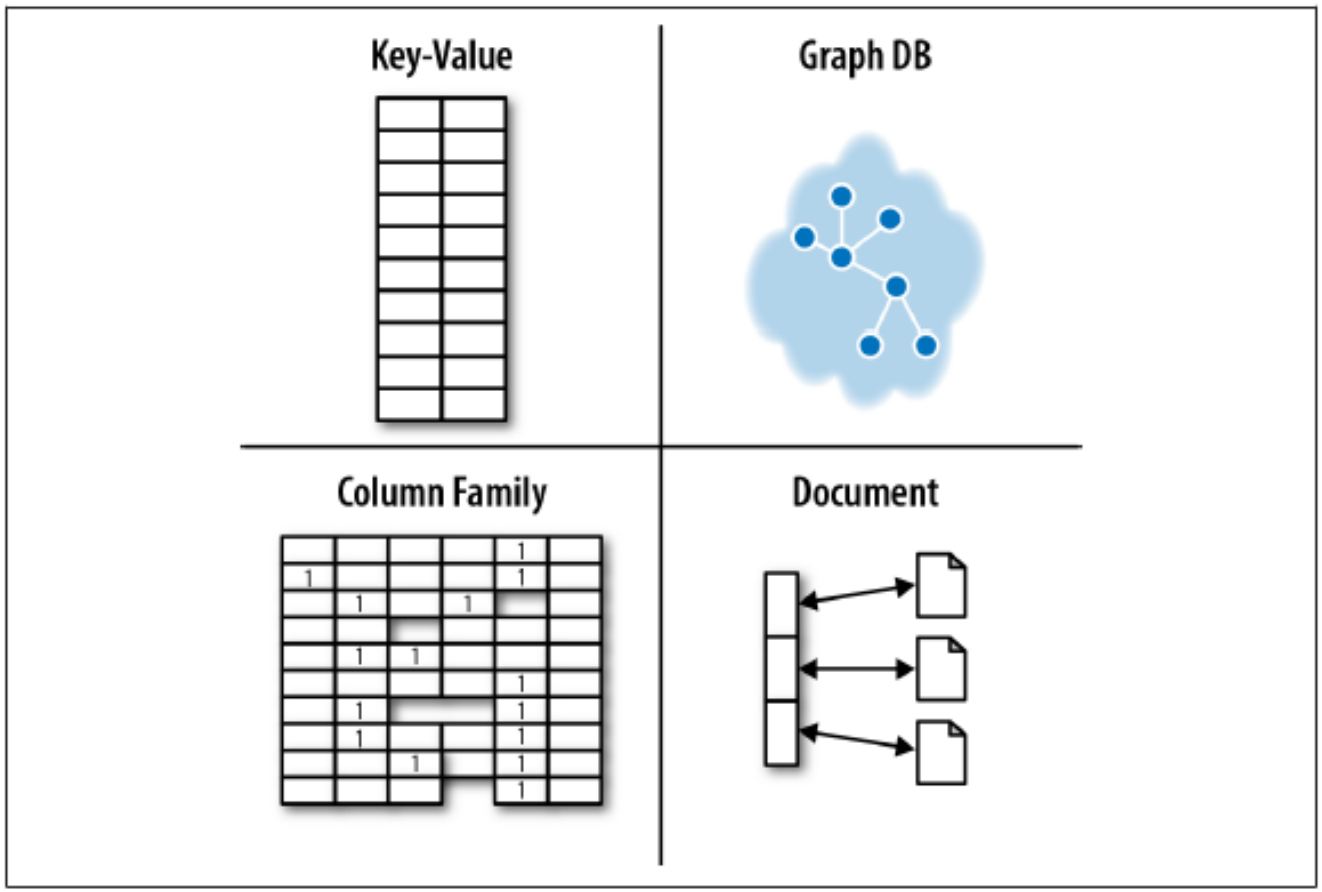
Proprietà “basiche” (BASE)

Per supportare al meglio le caratteristiche dei big data, i sistemi NoSQL si affidano alle proprietà BASE:

- *Basically available* Il sistema è accessibile per la maggior parte del tempo (es., tolleranza ai guasti dei nodi)
- *Soft state* Il sistema non deve essere in ogni dato istante consistente per quanto riguarda le scritture, né devono esserlo fra loro le diverse repliche dei dati memorizzate nei diversi nodi
- *Eventual consistency* Il sistema alla fine raggiunge sempre uno stato in cui i dati sono coerenti fra loro

Naturalmente, questo tipo di consistenza “rilassata” sarebbe un problema, ad esempio, in un database bancario. In altri casi, è perfettamente accettabile, ad esempio per i social network.

The NoSQL Quadrant



Basi di dati Chiave/Valore

Basi di dati Chiave/Valore

Le basi di dati chiave/valore sono fondate sul concetto di *array associativo*, il quale può essere visto come una lista formata da coppie <chiave, valore>; la chiave viene utilizzata per accedere ai valori.

Grazie alla modalità di implementazione degli array associativi (tabella di hash), recuperare il valore a partire da una chiave nota è praticamente istantaneo.

Le operazioni che possono essere svolte su un array associativo sono:

- *Add*: aggiunta di un elemento all'array
- *Remove*: rimozione di un elemento dall'array
- *Modify*: modifica di un valore associato ad una chiave
- *Find*: ricerca di un valore attraverso una chiave nota

Basi di dati Chiave/Valore - Focus

Le basi di dati chiave/valore supportano solo semplici operazioni sui dati (a loro volta semplici), che però vengono eseguite in un tempo costante (e molto veloce) grazie all'array associativo.

Alcune operazioni tipiche dei RDBMS, come i JOIN o l'applicazione di condizioni di filtraggio, non sono possibili (direttamente).

Inoltre, altre funzionalità dei RDBMS, come le chiavi esterne, non sono supportate nativamente.

In definitiva, questi sistemi offrono un modello dei dati piuttosto semplice. Per contro, possono essere estremamente sofisticati per quanto riguarda la modalità con cui viene implementata la scalabilità orizzontale.

Basi di dati Chiave/Valore - Berkeley DB

Berkeley DB è una base di dati chiave/valore ad elevate prestazioni sviluppata da Oracle Inc., con implementazioni in C, Java e XML/C++.

Supporta tipi complessi per le chiavi ed i valori, le proprietà ACIDE, un elevato accesso concorrente ai dati, e la distribuzione dei dati su più *repliche*.

Project Voldemort è una base di dati chiave/valore basata su Berkeley DB, sviluppata inizialmente da LinkedIn.

Progettata per garantire elevate prestazioni e tolleranza ai guasti.

I dati sono partizionati e replicati su diversi server. Ciascun nodo memorizza solo un sottoinsieme dell'intera quantità di dati. Ciò porta ad una migliore scalabilità orizzontale rispetto a Berkeley DB.

Le strategie di replicazione attuate consentono di garantire la tolleranza ai guasti dei singoli nodi.

Basi di dati document-oriented

Basi di dati Document-oriented

Le basi di dati document-oriented archiviano, recuperano e gestiscono informazioni orientate ai documenti, note anche come dati semi-strutturati.

I documenti non hanno una struttura fissa, ma contengono comunque tag o altri indicatori utilizzati per separare elementi semantici e imporre gerarchie di record e campi all'interno dei dati. Pertanto, possono essere considerati come una sorta di struttura auto-descrittiva.

Gli archivi document-oriented rappresentano un passo avanti rispetto agli archivi chiave-valore, in quanto definiscono una struttura su chiavi e valori, consentendo così agli utenti di effettuare operazioni ed interazioni più complesse.

Formati

I documenti possono essere codificati in formati quali JSON e XML.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<studenti>
    <studente>
        <matricola>0000001</matricola>
        <cognome>Rossi</cognome>
        <nome>Paolo</nome>
    </studente>
    <studente>
        <matricola>0000002</matricola>
        <cognome>Verdi</cognome>
        <nome>Francesco</nome>
    </studente>
</studenti>
```

Basi di dati Document-oriented - Operazioni

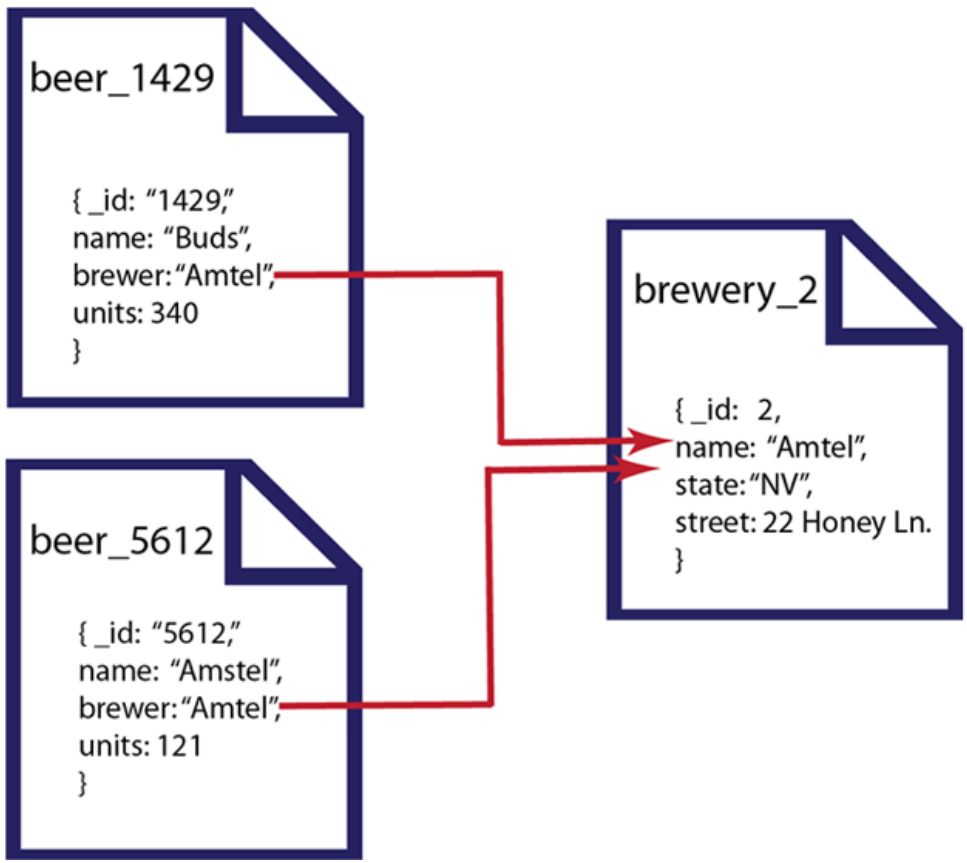
Le operazioni principali supportate da una base di dati document-oriented sono simili a quelle impiegate negli altri database, e prendono l'acronimo di CRUD:

- *Creation*: creazione di un nuovo documento
- *Retrieval*: ricerca basata su chiave, contenuto, metadati
- *Update*: aggiornamento del contenuto o dei metadati di un documento
- *Deletion*: cancellazione di un documento

Ogni documento nella base di dati è identificato univocamente da una chiave, che può essere utilizzata per recuperare il documento. Tipicamente è possibile definire indici sulle chiavi o su qualsiasi altro campo dei documenti per velocizzare le operazioni di recupero.

→ *MongoDB* è una diffusa base di dati document-oriented.

Documento - Esempio



Basi di dati column-oriented

Basi di dati column-oriented

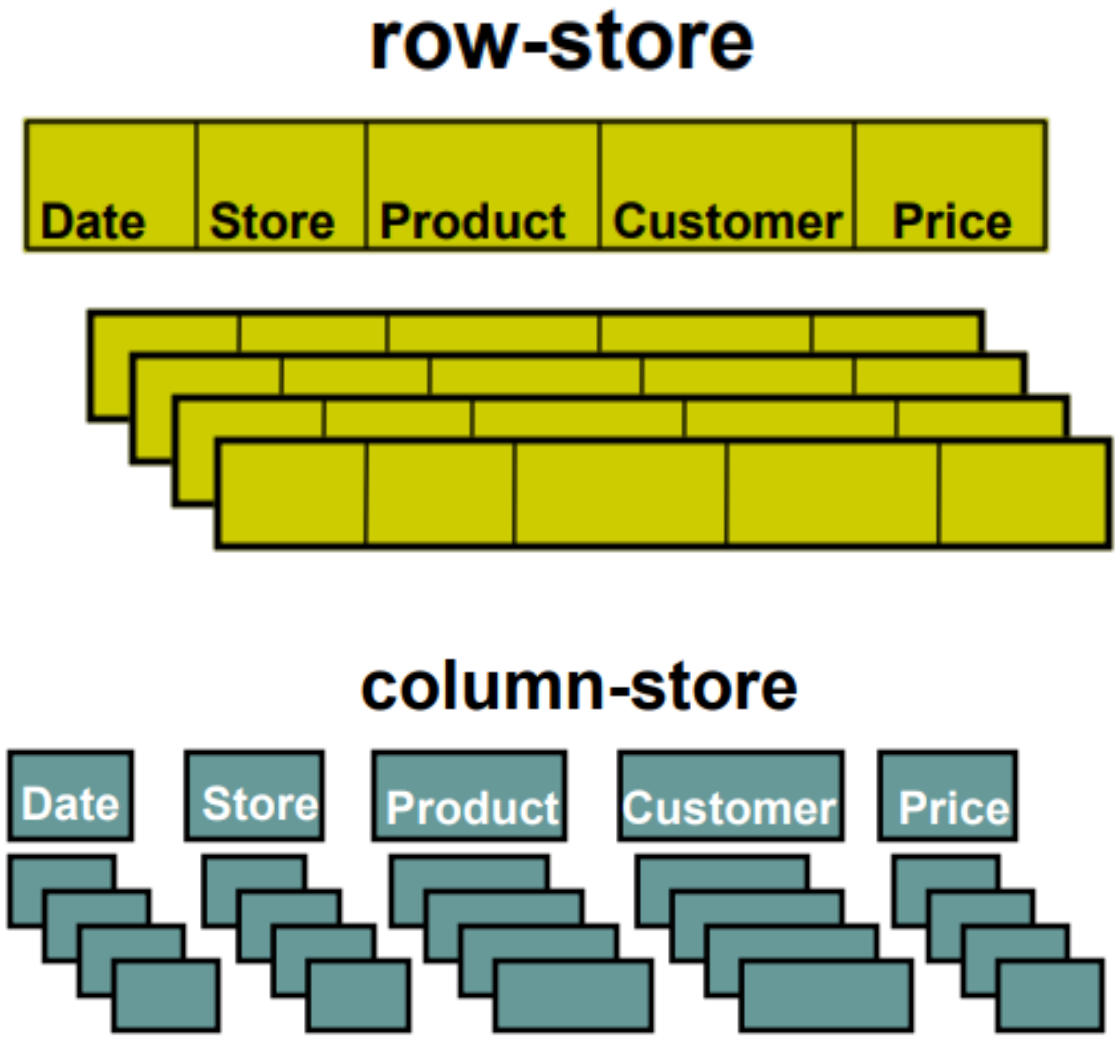
Un DBMS column-oriented memorizza ciascuna colonna di una data tabella separatamente, in diverse posizioni del disco.

I valori appartenenti ad una singola colonna sono quindi memorizzati assieme, a differenza della strategia adottata dai DBMS tradizionali che memorizzano i dati riga per riga.

Le origini delle basi di dati column-oriented possono essere fatte risalire agli anni '70.

Tuttavia, a causa delle esigenze di mercato e della tecnologia non ancora pronta, non si sono sviluppate fino agli anni 2000.

Righe VS Colonne



Righe VS Colonne

In una base di dati classica ogni campo è memorizzato accanto al successivo:

```
512,Seabiscuit,Book,10.95,201712241200,goodreads.com
513,Bowler,Apparel,59.95,201712241200,google.com
514.Cuphead,Game,20.00,201712241201,gamerassaultweekly.com
```

In un database column-oriented, l'informazione salvata su disco per i dati di cui sopra potrebbe avere il seguente aspetto:

```
512,513,514
Seabiscuit,Bowler,Cuphead
Book,Apparel,Game
10.95,59.95,20.00
201712241200,201712241200,201712241201
goodreads.com,google.com,gamerassaultweekly.com
```

Basi di dati column-oriented - Vantaggi

Memorizzare i dati in colonna porta a diversi vantaggi quando l'applicazione tipo è leggere i valori di un insieme di colonne, o applicare funzioni di aggregazione su di esse (AVG, MIN, MAX, ...).

Offre inoltre una maggiore efficienza di archiviazione grazie a una compressione più efficace dei dati.

I database column-oriented sono altamente scalabili e adatti per il *massively parallel processing* (MPP), che comporta il partizionamento e la distribuzione dei dati su un ampio cluster di macchine.

Basi di dati column-oriented - Svantaggi

Gli svantaggi principali riguardano le operazioni di scrittura e la ricostruzione delle tuple.

I nuovi dati devono essere scomposti in colonne (overhead), e ogni colonna deve essere scritta separatamente.

Anche la ricostruzione delle tuple comporta dei problemi, dal momento che l'informazione completa è frammentata su diverse locazioni del disco, o diversi nodi. Le query che accedono a molte colonne devono quindi affrontare dell'overhead dato dal processo di riassemblaggio.

Cassandra Column Store

Apache Cassandra è una base di dati column-oriented open-source, progettata per gestire grandi quantità di dati distribuiti su diversi server, garantendo un'elevata disponibilità del sistema.

- Scalabile orizzontalmente e quasi linearmente, resistente ai guasti del sistema e "eventually consistent"
- Sviluppata da Facebook, prima versione nel 2008
- Utilizzata da diverse compagnie quali Facebook, Twitter, Cisco, Rackspace, eBay, Twitter, Netflix

Query Language - CQL

Cassandra offre un linguaggio specifico per interagire con i dati, Cassandra Query Language (CQL):

- Comandi SQL-like: CREATE, ALTER, UPDATE, DROP, DELETE, TRUNCATE, INSERT, ...
- Molto più "grezzo" di SQL:
  - Non supporta le sotto-query ed operazioni quali i JOIN
  - Le clausole WHERE sono piuttosto semplici

CQL - Esempi

```
Example query 1:
        CREATE TABLE playlists(
        Id uuid,
        Song_order int,
        Song_id uuid,
        title text,
        album text,
        artist text,
        PRIMARY KEY (id, song_order));

Example query 2:
        INSERT INTO playlist (id, song_order, song_id, title, artist, album)
          VALUES (62c36092-82a1-3aoo-93d1-46196ee77204, 4,
                        7db1a490-5878-11e2-bcfd-o8oo2ooc9a66,
                        'ojo Rojo', 'Fu Manchu', 'No One Rides for Free');
Example query 3:
        SELECT * FROM playlists;
```

CQL - Tipi di dato

`ascii`	ASCII character string
`bigint`	64-bit signed long
`blob`	Arbitrary bytes (no validation)
`boolean`	true or false
`counter`	Counter column (64-bit long)
`decimal`	Variable-precision decimal
`double`	64-bit IEEE-754 floating point
`float`	32-bit IEEE-754 floating point
`int`	32-bit signed int
`text`	UTF8 encoded string
`timestamp`	A timestamp
`uuid`	Type 1 or type 4 UUID
`varchar`	UTF8 encoded string
`varint`	Arbitrary-precision integer

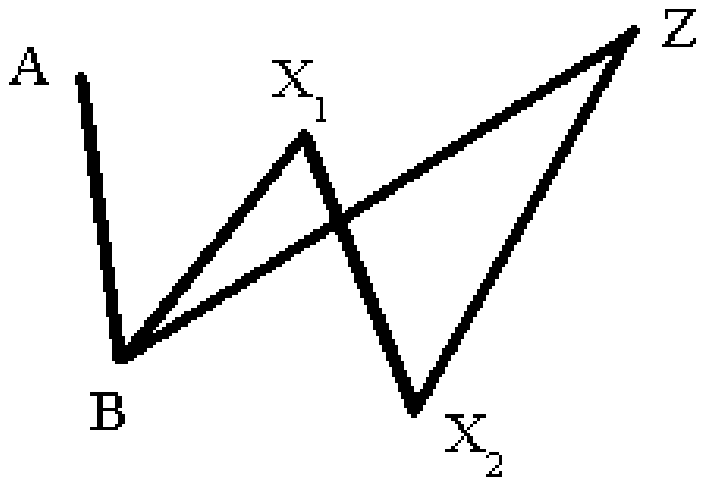
Altri DBMS column-oriented

- *Bigtable*: sistema proprietario Google basato su Google File System che offre alte prestazioni e avanzate funzionalità di compressione dati
- *HBase*: parte del framework open-source Hadoop, modellato su Bigtable
- *Vertica*: sistema commeciale con supporto al linguaggio SQL
- *Druid*: open-source, distribuito e scritto in linguaggio Java. Utilizzato da diverse compagnie come Alibaba, Airbnb, and Cisco
- *Accumulo*: costruito su Apache Hadoop e basato su Google Bigtable, è il terzo sistema column-oriented in ordine di popolarità, dietro Apache Cassandra e HBase

Basi di dati a grafo

Cos'è un grafo

Un grafo è un insieme di elementi detti *nodi o vertici* che possono essere collegati fra loro da linee chiamate *archi*. Più formalmente, si dice grafo una coppia ordinata $G = (V, E)$ di insiemi, con V insieme dei nodi e E insieme degli archi, tali che gli elementi di E sono coppie di elementi di V ($E \subseteq V \times V$).

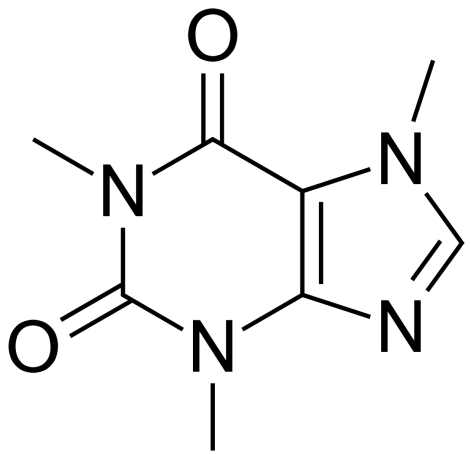


Un grafo con vertici A, B, X_1, X_2 e Z
e archi AB, BX_1, BZ, X_1X_2 e X_2Z

Utilità dei grafi

Attraverso i grafi è possibile modellare dati appartenenti a molteplici domini:

- Stutture di molecole
- Percorsi stradali
- Reti sociali
- Ontologie



Basi di dati a grafo

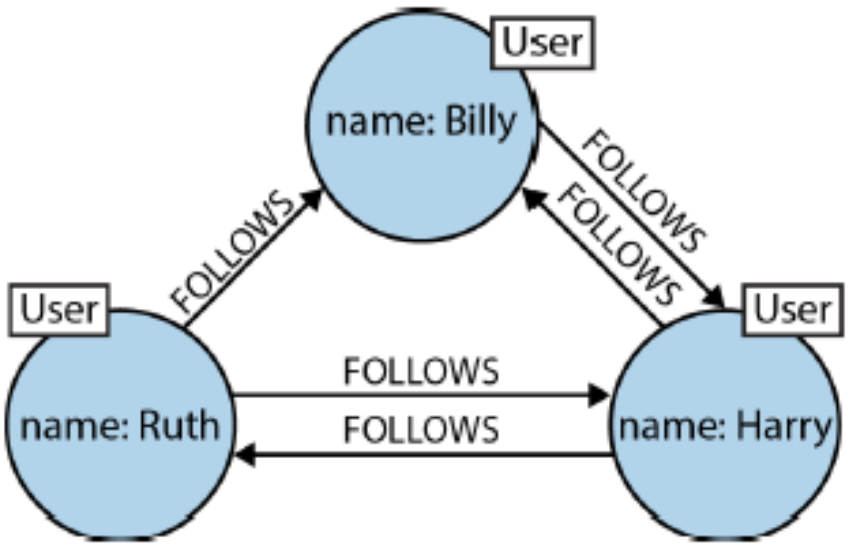
Una base di dati a grafo utilizza strutture a grafo con nodi, archi e proprietà per rappresentare e archiviare i dati.

I database a grafo possono rappresentare in modo naturale alcuni tipi di dati semi-strutturati e altamente interconnessi, come quelli presenti nei social network o nelle applicazioni geospaziali e biotecnologiche.

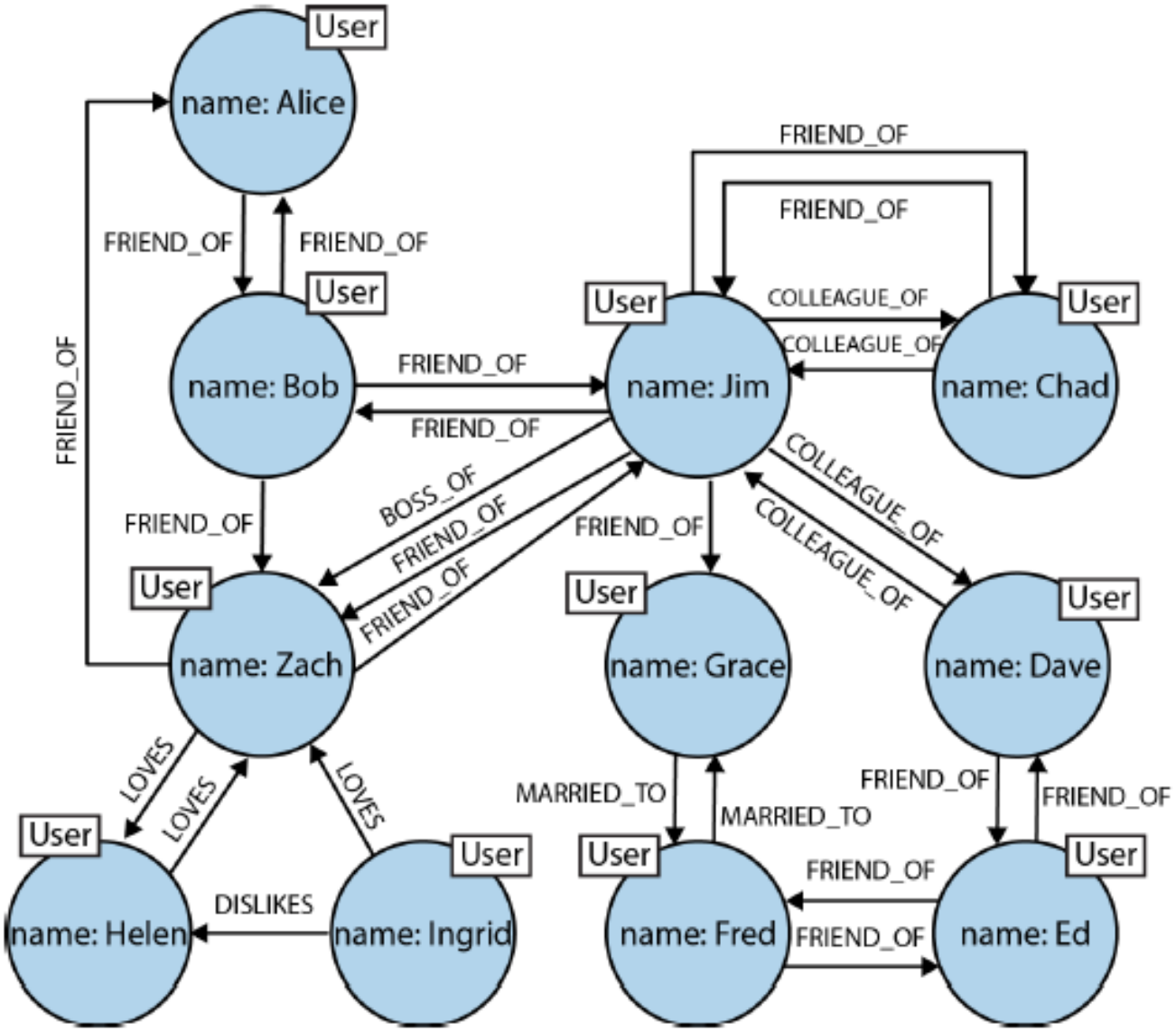
I database a grafo sono tipicamente progettati per supportare le proprietà acide.

Esempi: *Amazon Neptune*, *Neo4j*, *Spark GraphX*, *Apache Giraph*.

Esempio di grafo - 1



Esempio di grafo - 2



Basi di dati a grafo – Vantaggi

Chiaramente, i grafi appena visti potrebbero essere modellati utilizzando altri tipi di approcci NoSQL, nonché RDBMS.

Tuttavia, i database risultanti risulterebbero molto difficili da interrogare, aggiornare e popolare.

Al contrario, le basi dati a grafo possono facilmente rispondere a interrogazioni come:

- Qual è il cammino più breve che connette il nodo X con il nodo Y?
- Chi sono gli amici di *Billy*?
- Chi sono gli amici degli amici di *Billy*?

Ci sono due proprietà fondamentali che caratterizzano le basi di dati a grafo:

- **Modalità di memorizzazione fisica dell'informazione**: alcuni database a grafo sono *nativi*, mentre altri convertono i dati in modo da memorizzarli secondo il modello relazionale, o sfruttando altre soluzioni NoSQL
- **Modalità di processamento dell'informazione**: nei database nativi, i nodi connessi puntano fisicamente l'uno all'altro (puntatori alla loro locazione); negli altri, occorre tradurre le query su grafo in query SQL (nel caso di implementazione relazionale)

Database nativi e puntatori

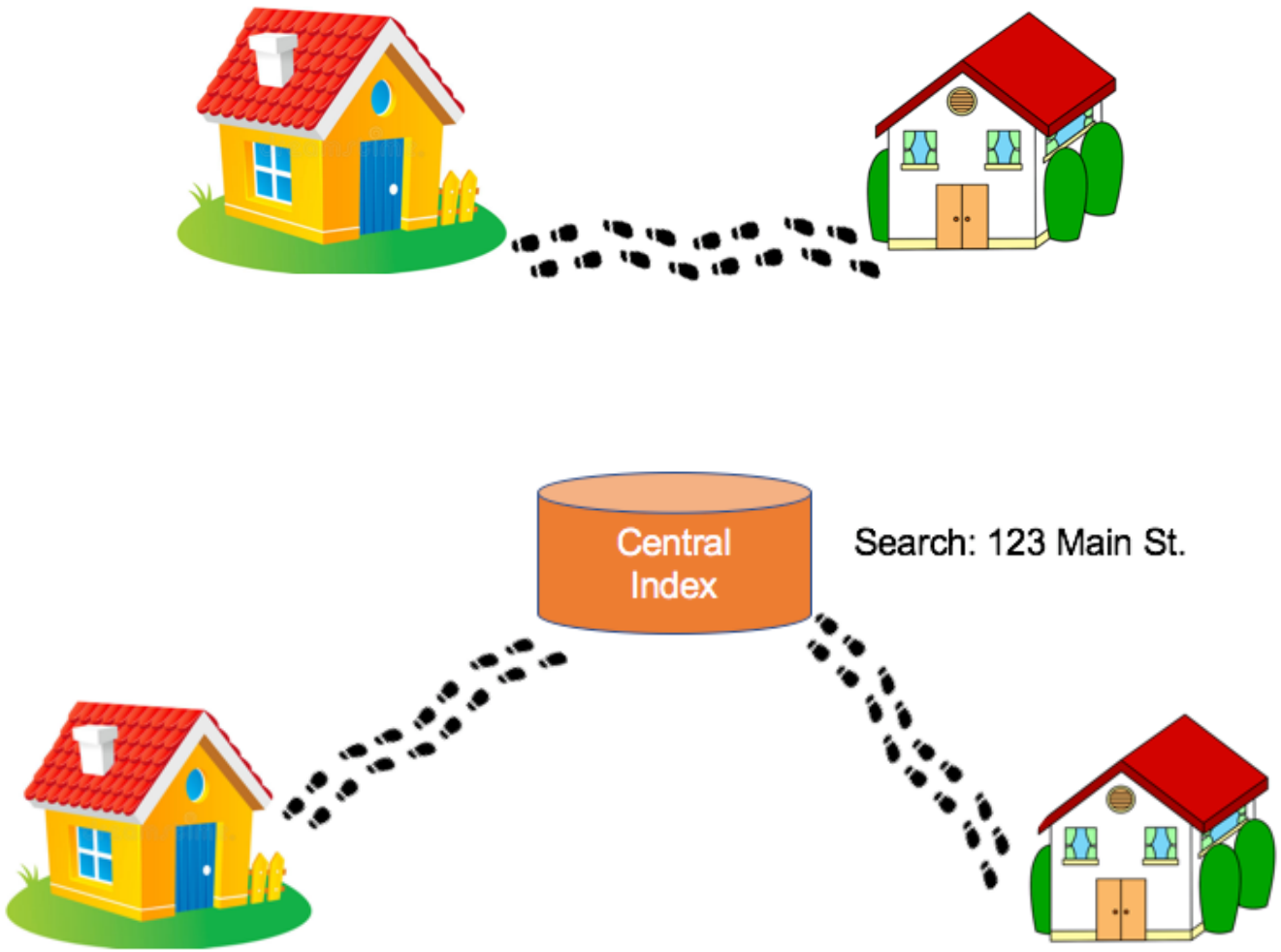
I database a grafo nativi non necessitano di indici, perché possono sfruttare i puntatori per implementare le relazioni fra i nodi.

I puntatori consentono un accesso diretto all'informazione, in quanto memorizzano la locazione fisica dei nodi.

I puntatori rendono possibile l'attraversamento di milioni di nodi in una manciata di secondi, a differenza della stessa operazione implementata in un modello relazionale tramite JOIN che può essere ordini di grandezza più lenta.

Al crescere delle dimensioni del grafo, il costo di passare da un nodo all'altro rimane costante (a differenza che in un RDBMS).

Nativo VS non nativo



Basi di dati a grafo - Modelli dei dati

Oltre alla modalità di memorizzazione fisica dell'informazione, le basi di dati a grafo possono adottare diversi modelli dei dati, ad esempio:

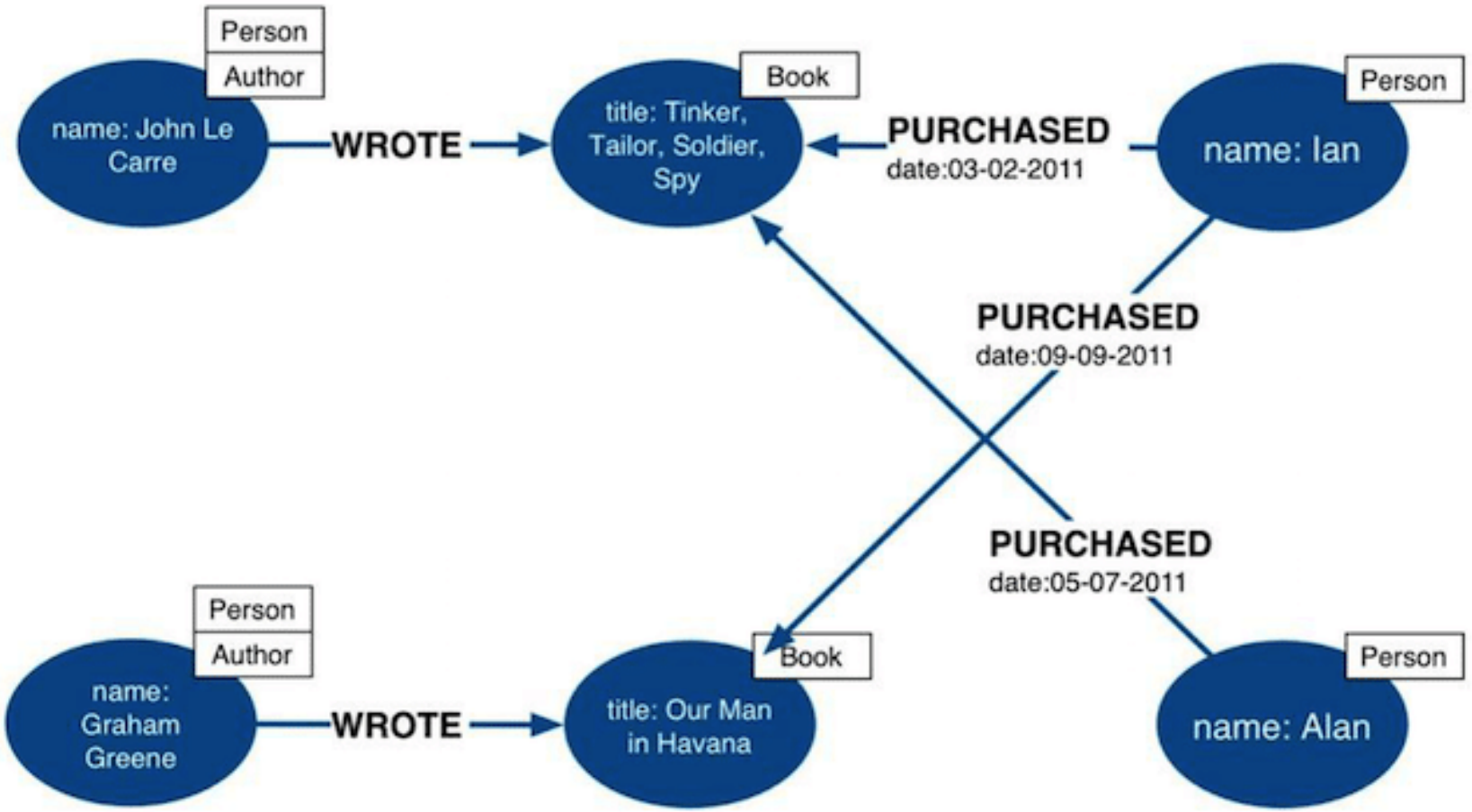
- Property graph
- Ipergrafi
- Triplette

Basi di dati a grafo - Property graph

Un property graph ha le seguenti caratteristiche:

- È costituito da nodi e relazioni
- I nodi contengono proprietà (coppie chiave-valore)
- I nodi possono avere una o più etichette (es., per codificare gerarchie)
- Ciascuna relazione ha un nome, è rappresentata da un arco diretto, e ha esattamente una sorgente e una destinazione
- Anche le relazioni possono contenere proprietà

Property Graph - Esempio

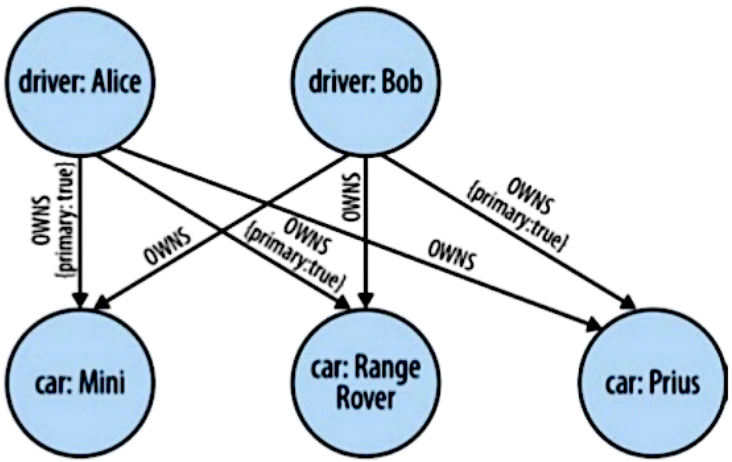


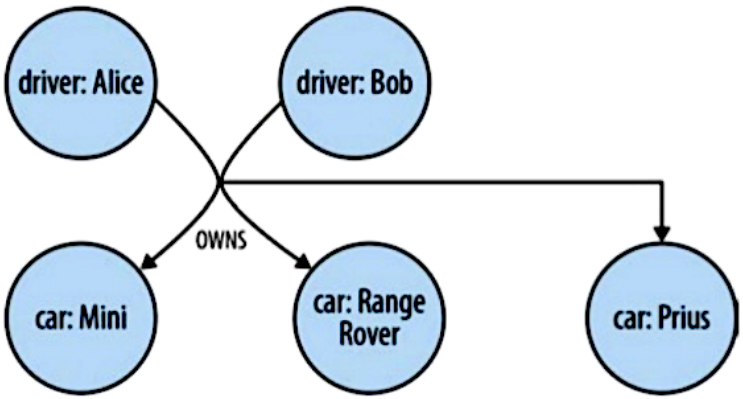
Basi di dati a grafo - Ipergrafi

Un ipergrafo è una generalizzazione del concetto di grafo in cui una relazione (detta iperarco) può connettere più nodi fra loro.

Gli ipergrafi sono particolarmente utili per gestire scenari in cui vi sono diverse relazioni molti-a-molti.

È sempre possibile rappresentare l'informazione codificata in un ipegrafo tramite un property graph con archi classici.





Basi di dati a grafo - Triplette

Il modello a triplette è storicamente legato all'ambito del *web semantico*, il cui obiettivo è di rendere Internet machine-readable.

L'informazione è codificata per mezzo di triplette, ciascuna delle quali può essere considerata una struttura del tipo soggetto-predicato-oggetto

Attraverso le tripletteè possibile catturare fatti come "Ginger balla con Fred" e "A Fred piace il gelato".

Standard RDF (Resource Description Framework) per il formato delle triplette, e linguaggio SPARQL per l'interrogazione e la modifica.

Basi di dati a grafo - Neo4j

Neo4j è una base di dati a grafo nativa, scritta nel linguaggio Java. Nato nel 2010, è in continuo sviluppo.

Supporta le proprietà acide delle transazioni.

Permette la scrittura di query attraverso il
Cypher query language.

La *community edition* è open-source e liberamente scaricabile, così come la versione desktop. È anche presente una *enterprise edition* a pagamento che offre maggiori prestazioni.

⇝ Sandbox: `https://neo4j.com/sandbox/?ref=neo4j-home`

Altri esempi di basi di dati a grafo

Apache Giraph sfrutta Hadoop MapReduce per processare i grafi. Inizialmente sviluppato da Yahoo ed in seguito donato all'Apache Foundation.

AllegroGraph è un sistema proprietario che sfrutta il modello a triplette, progettato per gestire dati in formato RDF; supporta le proprietà acide.

ArangoDB è un DBMS multi-modello open-source. Supporta tre diversi modelli dei dati: chiave/valore, document-oriented, e a grafo.

Bibliografia

Stavros Harizopoulos, Daniel Abadi, Peter Boncz (2009), Column-Oriented Database Systems, VLDB 2009 Tutorial

Ian Robinson, Jim Webber & Emil Eifrem, Graph Databases Second Edition, O'Reilly Media, Inc.

Big Data Management and NoSQL Databases Lecture 7. Column-family stores, Doc. RNDr. Irena Holubova, Ph.D.