ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

NOME CORSO

Fintech Software Developer

Unità Formativa (UF): Basi di dati SQL

Docente: Durando Giulio

Titolo argomento: Tipi di database NoSQL

Database NoSQL

L'acronimo NoSQL significa "**not only SQL**", e proprio in questo senso deve essere interpretato questo modello di banche dati: assolutamente non come una soluzione contrapposta, quanto piuttosto come un arricchimento e un'**utile integrazione dei tradizionali database SQL relazionali**. I database NoSQL, infatti, superano i limiti dei sistemi relazionali, utilizzando modelli di banche dati alternativi. Questo, tuttavia, non significa necessariamente che i sistemi SQL non vengano utilizzati: esistono infatti anche numerose varianti miste in cui convivono entrambe le soluzioni e che vengono anch'esse comprese nella definizione generale di NoSQL.

Fino alla fine degli anni 2000, i database SQL rappresentavano il non plus ultra del progresso applicativo. Altri approcci, come i database orientati ad oggetti, non hanno mai raggiunto una rilevanza paragonabile in ragione della loro complessità di utilizzo. Un'alternativa vera e propria si è concretizzata grazie allo sviluppo dei database NoSQL, che sono stati la risposta alle **limitazioni e ai problemi delle banche dati relazionali**. Queste ultime, infatti, spesso non sono in grado di soddisfare le esigenze dei moderni sviluppi applicativi. A differenza delle banche dati relazionali, i sistemi NoSQL utilizzano innovazioni, come i cloud server, per offrire un modello di dati alternativo in cui possono essere **archiviati ed elaborati moltissimi dati di diverso tipo**. Le strutture di dati così ottenute sono **efficienti** e **flessibili** e sono in grado di reagire rapidamente al mutamento delle esigenze.

I sistemi NoSQL vengono di frequente denominati sistemi ad archivi strutturati, definizione che ne esplicita la differenza sostanziale rispetto alle banche dati SQL relazionali: a differenza di queste ultime, i database NoSQL non utilizzano schemi tabellari rigidi all'interno dei quali i dati devono essere definiti prima dell'archiviazione. I database NoSQL, infatti, fanno ricorso a metodi più flessibili, che permettono di aggiungere senza difficoltà nuovi set di dati che si aggiornano di continuo nell'applicazione. Le soluzioni NoSQL sono pertanto adatte ad essere utilizzate per l'elaborazione di dati non strutturati o anche sconosciuti, cosa impensabile con una banca dati relazionale.

Funzionamento dei database NoSQL

A differenza delle banche dati relazionali basate su SQL, i database NoSQL non utilizzano per l'archiviazione dei dati tabelle tradizionali con righe e colonne. Anziché utilizzare le tabelle, **organizzano grandi quantità di dati con l'ausilio di tecniche flessibili**, ad esempio con documenti, grafici, coppie di valori e colonne. Perciò, i sistemi NoSQL sono indicati per le applicazioni che prevedono l'elaborazione di grandi quantità di dati e che, pertanto, richiedono **strutture flessibili**. Poiché i sistemi NoSQL utilizzano hardware cluster e cloud server, le **capacità vengono ripartite in modo uniforme** e il database lavora senza difficoltà anche in caso di un'elevata mole di dati. A differenza dei database relazionali, che all'aumentare della quantità di dati accusano rapidamente un calo di efficienza, i database NoSQL rappresentano una soluzione efficiente, flessibile e scalabile anche in caso di elevato volume di dati.

Una particolarità dei sistemi NoSQL è inoltre la **scalabilità orizzontale**. Le banche dati SQL relazionali sono scalate in direzione verticale e basano tutta la loro efficienza su un singolo server. Un incremento della loro capacità passa necessariamente attraverso l'investimento in un server più potente cosa che, nel tempo, non solo è correlata a costi più elevati, ma limita anche le possibilità di sviluppo applicativo. Le soluzioni NoSQL ripartiscono generalmente i dati su diversi server. In caso di aumento della quantità di dati, è sufficiente **aggiungere nuovi server**. Questo fa sì che i database NoSQL siano in grado di memorizzare ed elaborare senza difficoltà grandi volumi di dati, caratteristica che li rende idonei soprattutto per le applicazioni big data.

I quattro principali approcci NoSQL

I sistemi ad archivi strutturati rinunciano agli schemi rigidi dei database relazionali, diventando così soluzioni particolarmente indicate per le applicazioni big data. A livello mondiale sono già disponibili **numerosi sistemi NoSQL** di vario tipo, perlopiù open source, che presentano **strutture diverse** a seconda dello sviluppatore e dei requisiti. Sebbene non esistano regole uguali per tutti, i diversi approcci NoSQL possono essere suddivisi in **quattro categorie principali**.

Database orientati ai documenti

Nei database NoSQL orientati ai documenti, i dati vengono memorizzati direttamente in documenti con lunghezza variabile. Per essere memorizzati, i dati non devono necessariamente essere strutturati. Ad essi vengono associati dei cosiddetti **attributi** o **"tag"** in base ai quali è possibile effettuare ricerche all'interno dei contenuti del documento. I database NoSQL orientati ai documenti sono particolarmente adatti per i **sistemi di content management** e per i **blog**. Come formato dei dati viene attualmente utilizzato soprattutto il formato **JSON** (**J**ava**S**cript **O**bject **N**otation) che permette uno scambio veloce dei dati tra le applicazioni.

Database a grafo

Un database a grafo crea delle relazioni tra i dati utilizzando nodi e archi. Attraverso i **nodi** e i **collegamenti tra di essi** viene organizzata la rete di relazioni tra i dati. Nel caso di raccolte di dati con informazioni fortemente correlate, i database a grafo NoSQL raggiungono pertanto performance decisamente migliori rispetto alle banche dati SQL relazionali. Questi database vengono utilizzati soprattutto **nell'ambito dei social media**, ad esempio per rappresentare le relazioni tra follower su Twitter o Instagram.

Database key-value (chiave-valore)

Mentre le banche dati SQL relazionali utilizzano schemi e tabelle rigidi, i database chiave-valore archiviano i dati sotto forma di coppie chiave-valore. Ai singoli valori vengono associate chiavi specifiche, laddove il set di dati stesso funge da chiave (key) e rappresenta un valore (value). La chiave costituisce al contempo un indice con il quale è possibile eseguire ricerca all'interno del database. Le chiavi dei database chiave-valore NoSQL sono sempre univoche e sono assimilabili alle chiavi primarie delle banche dati relazionali.

Database orientati alle colonne

A differenza dei modelli di banche dati relazionali, i sistemi di database orientati alle colonne archiviano i **set di dati non in righe, bensì in colonne**. Questo rende possibili processi di lettura dati più brevi e un'elevata efficienza. Questo modello NoSQL viene utilizza soprattutto per il **data mining e i programmi di analisi**.

Vantaggi del NoSQL rispetto a SQL

A seconda della finalità dell'applicazione, il NoSQL può offrire determinati vantaggi rispetto ai database relazionali: laddove i sistemi SQL raggiungono rapidamente i loro limiti di capacità, come ad esempio nelle applicazioni di elaborazione di big data, i database NoSQL offrono modelli ad alta efficienza e scalabili, in grado di leggere ed elaborare enormi quantità di dati a grandissima velocità.

I database NoSQL abbandonano gli schemi rigidi dei sistemi SQL e puntano invece su modelli flessibili, particolarmente adatti ad elaborare grandi quantità di dati. Poiché **archiviano i dati su hardware cluster distribuiti**, sono meno esposti a guasti e decisamente più economici rispetto

all’installazione di un unico potente server, le cui capacità vengono puntualmente esaurite rendendo necessario passare a un sistema con capacità superiore.

Differenze tra database SQL e NoSQL

Database SQL	Database NoSQL
Un database per tutte le applicazioni	Diversi modelli di database, ad es. database orientati ai documenti, database a grafo, database chiave-valore e database a colonne
I singoli dati (ad es. “titolo del libro”) vengono archiviati in righe all’interno di una tabella e associati a determinati attributi (ad es. “autore”, “anno di pubblicazione”, ecc.). I set di dati vengono archiviati in tabelle separate e riassemblati dal sistema in caso di query di ricerca complesse.	I database NoSQL non utilizzano tabelle, ma a seconda del tipo fanno ricorso a documenti completi, chiavi-valori, grafi o colonne.
Il tipo e la struttura dei dati vengono definiti in anticipo. Per archiviare nuove informazioni è necessario modificare l’intero database (e, a questo scopo, passare alla modalità offline).	Flessibile. I nuovi set di dati possono essere aggiunti immediatamente. I dati strutturati, semi-strutturati e non strutturati possono essere archiviati insieme, non è necessaria alcuna conversione preliminare.
Scalabilità verticale. Un solo server deve assicurare le prestazioni dell’intero sistema di banca dati; questo determina un calo dell’efficienza in caso di grandi volumi di dati.	Scalabilità orizzontale. Ciascun amministratore può aggiungere nuovi commodity server e cloud server; il database NoSQL invia i dati automaticamente a tutti i server.
Open source (ad es. MySQL) oppure software a pagamento (Oracle Database)	Open source o software a pagamento
I database SQL offrono tutte le caratteristiche ACID.	Per poter mantenere la loro flessibilità e la scalabilità orizzontale, la maggior parte dei database NoSQL non supporta le transazioni ACID. Al loro posto viene utilizzato il modello BASE (Basically Available, Soft State, Eventually Consistant). Vale quindi il principio secondo cui la disponibilità è prioritaria rispetto alla coerenza.
In caso di elevato volume di dati, utilizzano degli indici. Per aumentare le prestazioni dei sistemi SQL, è necessario ottimizzare le query, gli indici e la struttura.	Poiché utilizzano cloud server e hardware cluster, i database NoSQL offrono prestazioni nettamente superiori.
Le richieste di archiviazione e richiamo dei dati vengono trasmesse con SQL (Structured Query Language).	I dati vengono archiviati e richiamati attraverso API basate su oggetti.

Database di documenti

I database di documenti offrono una varietà di vantaggi, tra cui:

- Un modello di dati intuitivo con cui gli sviluppatori possono lavorare in modo facile e veloce.
- Uno schema flessibile che consente al modello di dati di evolversi in base alle esigenze dell'applicazione.
- La possibilità di ridimensionare orizzontalmente.

A causa di questi vantaggi, i database di documenti sono database generici che possono essere utilizzati in una varietà di casi d'uso e settori.

I database di documenti sono considerati database non relazionali (o **NoSQL**). Invece di archiviare i dati in righe e colonne fisse, i database dei documenti utilizzano documenti flessibili. I database di documenti sono l'alternativa più popolare ai database relazionali tabulari.

Struttura di un database di documenti

Gli elementi di un database di documenti sono

- **Documents**
- **Collections**

Documents

Un **document** è l'equivalente di un record in un database relazionale. Un documento in genere memorizza le informazioni su un oggetto e sui relativi metadati.
I documenti memorizzano i dati in coppie campo-valore. I valori possono essere una varietà di tipi e strutture, inclusi stringhe, numeri, date, matrici o oggetti. I documenti possono essere archiviati in formati come JSON, BSON e XML.
Di seguito è riportato un documento JSON che memorizza le informazioni su un utente di nome Tom dove in rosso sono evidenziati i campi e in verde i valori.


```
{
    "_id": 1,
    "first_name": "Tom",
    "email": "tom@example.com",
    "cell": "765-555-5555",
    "likes": [
      "fashion",
      "spas",
      "shopping"
    ],
    "businesses": [
      {
        "name": "Entertainment 1080",
        "partner": "Jean",
        "status": "Bankrupt",
        "date_founded": {
          "$date": "2012-05-19T04:00:00Z"
        }
      },
      {
```

```
      "name": "Swag for Tweens",
      "date_founded": {
        "$date": "2012-11-01T04:00:00Z"
      }
    }
  ]
}
```

Collections

Una **collection** è un gruppo di documenti. Le raccolte in genere archiviano documenti con contenuti simili.

Non tutti i documenti in una raccolta devono avere gli stessi campi, perché i database dei documenti hanno uno schema flessibile. Alcuni database di documenti forniscono la ***convalida dello schema*** , quindi lo schema può essere facoltativamente bloccato quando necessario.

Continuando con l'esempio sopra, il documento con informazioni su Tom potrebbe essere archiviato in una raccolta denominata users. È possibile aggiungere più documenti alla collection ***users*** per archiviare informazioni su altri utenti. Ad esempio, il documento sottostante che memorizza le informazioni su Donna potrebbe essere aggiunto alla collection ***users***.

```
{
    "_id": 2,
    "first_name": "Donna",
    "email": "donna@example.com",
    "spouse": "Joe",
    "likes": [
      "spas",
      "shopping",
      "live tweeting"
    ],
    "businesses": [
      {
        "name": "Castle Realty",
        "status": "Thriving",
        "date_founded": {
          "$date": "2013-11-21T04:00:00Z"
        }
      }
    ]
}
```

Come si può vedere, il documento per Donna non contiene gli stessi campi del documento per Tom. La collection ***users*** sfrutta uno schema flessibile per archiviare le informazioni esistenti per ciascun utente.

Operazioni CRUD

I database dei documenti in genere dispongono di un'API o di un linguaggio di query che consente agli sviluppatori di eseguire le operazioni CRUD (creazione, lettura, aggiornamento ed eliminazione).

- **Create** : i documenti possono essere creati nel database. Ogni documento ha un identificatore univoco.
- **Read** : I documenti possono essere letti dal database. L'API o il linguaggio di query consente agli sviluppatori di eseguire query sui documenti utilizzando i loro identificatori univoci o valori di campo. Gli indici possono essere aggiunti al database per aumentare le prestazioni di lettura.
- **Update** : i documenti esistenti possono essere aggiornati, in tutto o in parte.
- **Delete** : i documenti possono essere eliminati dal database.

Caratteristiche principali dei database di documenti

I database dei documenti hanno le seguenti caratteristiche principali:

- **Modello del documento:** i dati vengono archiviati nei documenti (a differenza di altri database che archiviano i dati in strutture come tabelle o grafici). I documenti vengono mappati agli oggetti nei linguaggi di programmazione più diffusi, il che consente agli sviluppatori di sviluppare rapidamente le proprie applicazioni.
- **Schema flessibile**: i database dei documenti hanno uno schema flessibile, il che significa che non tutti i documenti in una raccolta devono avere gli stessi campi. Si noti che alcuni database di documenti supportano la convalida dello schema , quindi lo schema può essere facoltativamente bloccato.
- **Distribuito e resiliente:** i database dei documenti sono distribuiti, il che consente il ridimensionamento orizzontale (in genere più economico del ridimensionamento verticale) e la distribuzione dei dati. I database dei documenti forniscono resilienza tramite la replica.
- **Interrogazione tramite un'API o un linguaggio di query:** i database dei documenti dispongono di un'API o di un linguaggio di query che consente agli sviluppatori di eseguire le operazioni CRUD sul database. Gli sviluppatori hanno la possibilità di eseguire query sui documenti in base a identificatori univoci o valori di campo.

Cosa rende i database dei documenti diversi dai database relazionali?

Tre fattori chiave differenziano i database dei documenti dai database relazionali:

1. **L'intuitività del modello di dati: i** documenti si associano agli oggetti nel codice, quindi è molto più naturale lavorarci. Non è necessario scomporre i dati tra tabelle, eseguire join costosi o integrare un livello ORM (Object Relational Mapping) separato. I dati a cui si accede insieme vengono archiviati insieme, quindi gli sviluppatori hanno meno codice da scrivere e gli utenti finali ottengono prestazioni più elevate.
2. **L'ubiquità dei documenti JSON:** JSON è diventato uno standard consolidato per lo scambio e l'archiviazione dei dati. I documenti JSON sono leggeri, indipendenti dalla lingua e leggibili dall'uomo. I documenti sono un superset di tutti gli altri modelli di dati, quindi gli sviluppatori possono strutturare i dati nel modo richiesto dalle loro applicazioni: oggetti avanzati, coppie chiave-valore, tabelle, dati geospaziali e di serie temporali o nodi e bordi di un grafico.
3. **La flessibilità dello schema: lo schema** di un documento è dinamico e autodescrittivo, quindi gli sviluppatori non devono prima predefinirlo nel database. I campi possono variare da documento a documento. Gli sviluppatori possono modificare la struttura in qualsiasi momento, evitando migrazioni di schemi dirompenti. Alcuni database di documenti offrono la ***convalida dello schema*** in modo da poter applicare facoltativamente regole che regolano le strutture dei documenti.

Database di documenti più popolari

- **CouchDB :** Apache CouchDB è un database NoSQL open source, orientato ai documenti. Fornisce un archivio chiave-valore, in cui i dati sono salvati all'interno di documenti JSON
- **Lotus Notes :** Lotus Notes è il client applicativo ed e-mail di Domino, un software collaborativo client/server prodotto dalla divisione Lotus, di IBM
- **MongoDB :** MongoDB è un database NoSQL, orientato ai documenti in stile J**SON** con schema dinamico (MongoDB chiama il formato **BSON**), rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce.
- **Apache Solr:** è una piattaforma di ricerca open source del progetto **Apache Lucene**. La sua caratteristica principale è la presenza di ricerca full text, hit highlighting, faceted search,

raggruppamento dinamico, integrazione con le basi di dati, gestione di documenti "ricchi" (come documenti word e pdf).

- **OrientDB :** è un database documentale in cui le relazioni sono gestite principalmente come in un database a grafo con connessioni dirette tra i singoli dati. OrientDB supporta modalità senza schema, con schema oppure miste.

Graph database (database a grafo)

Il **graph database** o database a grafo è un modello di database in grado di gestire informazioni altamente interconnesse. Quest'ultimo fornisce la risposta ai problemi del classico database relazionale, il quale raggiunge rapidamente i suoi limiti nel caso di banche dati molto grandi e complesse, e si colloca così fra i database alternativi moderni, lontani dal tradizionale approccio relazionale e designati con il termine di **NoSQL ("Not only SQL").**

Cos'è un database a grafo

Un database a grafo (in inglese *graph database*), come già il nome stesso rivela, è **mappato su grafici**. Con essi, le complesse informazioni di rete e le relative relazioni sono visualizzate chiaramente e memorizzate come un insieme di dati ampio e coerente. I grafici sono composti da **nodi**, entità, oggetti di dati identificati e identificabili in modo univoco, e dai **bordi**, denominati anche ***edges***. Questi ultimi rappresentano le relazioni di questi oggetti tra loro. Entrambi i componenti sono **rappresentati** visivamente **come punti e linee**. I bordi presentano rispettivamente un punto di partenza e uno di arrivo, mentre ogni nodo mostra sempre un certo numero di relazioni, in entrata, in uscita o non orientate, con altri nodi.

Modelli correnti per la creazione di questi database a grafo sono il **labeled-property graph** e il **Resource Description Framework (RDF)**: con il primo vengono attribuite determinate proprietà (in inglese ***properties***) ai nodi e ai bordi. Nel Resource Description Framework (RDF) invece la modellazione del grafico è regolamentata da **triple e quadruple**: le triple sono costituite da tre elementi secondo lo schema **nodo-bordo-nodo**. Le quadruple completano le triple **con ulteriori informazioni contestuali**, rendendo facile raggruppare queste ultime in gruppi.

Funzionamento delle query in un database a grafo

Nell'applicazione di un database grafico vengono utilizzate query completamente diverse. Ciò è dovuto soprattutto al fatto che non esiste un linguaggio uniforme di interrogazione. A differenza dei modelli tradizionali, i database grafici si basano inoltre su **speciali algoritmi** per svolgere la loro funzione essenziale: semplificare e snellire complesse query di dati.

Tra i più importanti algoritmi figurano la **ricerca in profondità** e la **ricerca in ampiezza**: nella ricerca in profondità si ispeziona di volta in volta il nodo del livello inferiore, mentre la ricerca in ampiezza procede livello per livello. Gli algoritmi consentono di trovare modelli (*graph patterns*) e nodi limitrofi diretti e indiretti. Altri algoritmi permettono di calcolare il percorso più breve tra due nodi e di identificare cricche (sottoinsieme di nodi) e hotspot (informazioni altamente interconnesse). Uno dei punti di forza del database a grafo consiste nel fatto che le **relazioni sono memorizzate nel database stesso** e non devono pertanto essere calcolate prima con la query. Ne risulta una velocità di elevata performance anche in query complesse.

Differenza nei confronti di database relazionali e altri database NoSQL

I database relazionali si sono affermati come standard per i database sin dalla loro origine nel 1970. A differenza dei database grafici, essi lavorano **sulla base di tabelle** che gestiscono in singole righe le relazioni di set di dati nominati tuple. Nelle colonne invece è possibile raffigurare proprietà con diversi valori di attributo. Oltre alla struttura e all'architettura anche il loro funzionamento differisce sostanzialmente dalla rappresentazione tramite grafico. Per essere in grado di visualizzare e salvare le relazioni nel caso di informazioni altamente collegate in rete, diverse tabelle devono essere interconnesse in modo complesso e contabilizzate. Questo può spesso richiedere molto tempo e denaro per grandi quantità di dati.

Mentre i database che si basano su tabelle utilizzano esclusivamente il **linguaggio di interrogazione SQL** ("Structured Query Language"), i database NoSQL più moderni si stanno allontanando sempre di più da questo linguaggio di interrogazione e dal concetto relazionale associato, approccio seguito anche dai database grafici come membri del gruppo NoSQL.

A cosa servono i database a grafo?

I database a grafo possono essere applicati a diversi settori e scopi. Essi consentono di analizzare informazioni collegate in rete e di comprendere, valutare e sfruttare processi e connessioni.

Un esempio significativo dell'impiego di database a grafo è l'analisi delle relazioni degli utenti nei social network o del comportamento d'acquisto degli utenti nello shopping online. Sulla base di dati e relazioni diversi, possono quindi essere fornite in un secondo momento raccomandazioni abbastanza mirate **su prodotti o amicizie** ed essere create reti di prodotti o persone. Le aziende traggono vantaggio dalla possibilità di creare **profili clienti completi** sulla base di informazioni provenienti da query di ricerca, cronologia dei clic e altri fattori. Nella gestione della catena di distribuzione vengono utilizzati i database grafici al fine di seguire lo sviluppo di tutti i processi dal design alla vendita. Infine i database sono impiegati per le **analisi del rischio**, l'**accertamento delle frodi** e la **ricerca degli errori**.

Vantaggi e svantaggi del database a grafo

La potenza di un database può essere misurata principalmente sulla base di quattro fattori: **integrità, performance, efficienza** e **scalabilità**. La query di dati dovrebbe essere più veloce e più semplice: così può essere sintetizzato approssimativamente lo scopo principale dei database grafici. Laddove, ad esempio, i database relazionali raggiungono i loro limiti di prestazioni, il modello basato sui grafici funziona in modo molto agile. Infatti complessità e quantità dei dati non influiscono negativamente sul processo di query in questo modello.

Inoltre con il modello del database a grafo è possibile **memorizzare fatti reali in modo naturale**. La struttura utilizzata è molto simile al pensiero umano e pertanto rende i collegamenti particolarmente vividi. Tuttavia, anche i database grafici non sono soluzioni olistiche. Raggiungono per esempio i loro **limiti** col tema **scalabilità**. Essendo progettati soprattutto per un'architettura a server singolo la crescita rappresenta una sfida (matematica). Inoltre non esiste ancora un linguaggio di query uniforme.

Vantaggi e svantaggi dei database grafici in sintesi:

Vantaggi	Svantaggi
La velocità della query dipende solo dal numero di relazioni concrete e non dalla quantità di dati	Scarsa scalabilità, in quanto progettati per un'architettura a server singolo
Risultati in tempo reale	*Linguaggio di query non uniforme*
Presentazione chiara e comprensibile delle relazioni	
Strutture flessibili e agili	

Fondamentalmente i database grafici non dovrebbero essere considerati un sostituto assoluto e migliore dei database tradizionali. Quindi le strutture relazionali continuano a essere utili modelli standard, che garantiscono un'elevata integrità e stabilità dei dati e consentono una scalabilità flessibile.

Database grafici più popolari

- **Neo4j**: Neo4j è il database grafico più popolare ed è progettato come modello open source.
- **Amazon Neptun**: questo database grafico è disponibile attraverso il cloud pubblico di Amazon Web Services ed è stato pubblicato nel 2018 come database ad alte prestazioni.
- **SAP Hana Graph**: con SAP Hana lo sviluppatore SAP ha creato una piattaforma, che si basa su un sistema di gestione di database relazionali ed è completata dal modello SAP Hana Graph integrato e orientato ai grafici.
- **OrientDB**: questo database grafico è considerato uno dei modelli più veloci attualmente disponibil

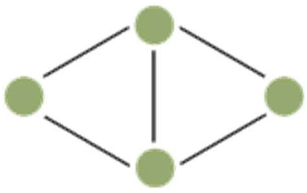
Un grafico è costituito da nodi, linee e proprietà che rappresentano le relazioni all'interno dei dati. Un database di grafici memorizza i grafici e fornisce funzionalità integrate per i grafici di query.

Le linee in genere hanno una direzione che va da un oggetto all'altro o più oggetti. Nodi e linee formano una rete di punti dati chiamata "grafico".

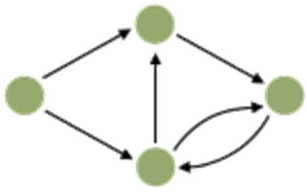
Nella matematica discreta, un grafo è definito come un insieme di vertici e spigoli. In informatica, è considerato un tipo di dati astratto che è davvero bravo a rappresentare connessioni o relazioni, a differenza delle strutture dati tabulari dei sistemi di database relazionali, che ironicamente sono molto limitate nell'esprimere relazioni.

Nozioni di base sul database grafico

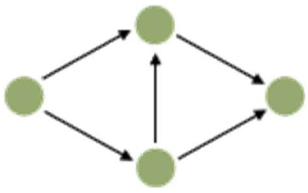
I grafici possono essere di natura diversa. I grafici possono essere non orientati, diretti o formare un cosiddetto grafico aciclico diretto (DAG).



Non orientato



Orientato

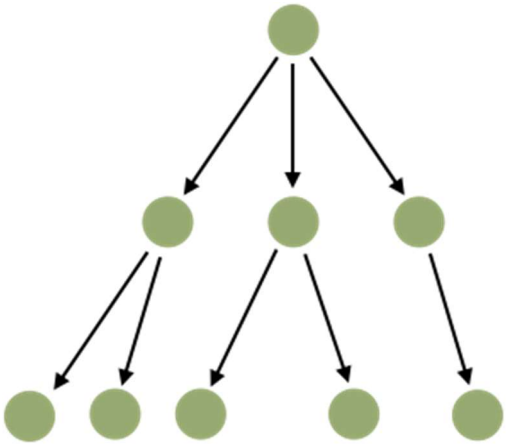


Aciclico diretto

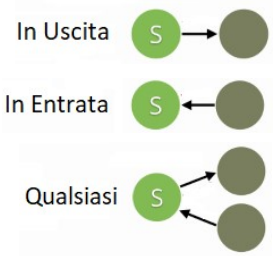
Non orientato – i bordi connettono coppie di nodi senza avere una nozione di direzione

Orientati: i bordi hanno una direzione ad essi associata

Grafico aciclico diretto (DAG): i bordi hanno una direzione e non ci sono loop. Un esempio per un DAG è una topologia ad albero.



Gli archi memorizzati hanno sempre una direzione da un vertice ad un altro. Visti da un certo vertice, gli archi in entrata sono chiamati archi in entrata e in uscita in uscita. Durante le query, la direzione memorizzata può essere ignorata dalla query effettiva quando si decide di seguire qualsiasi direzione.



Modello di query in un database grafico

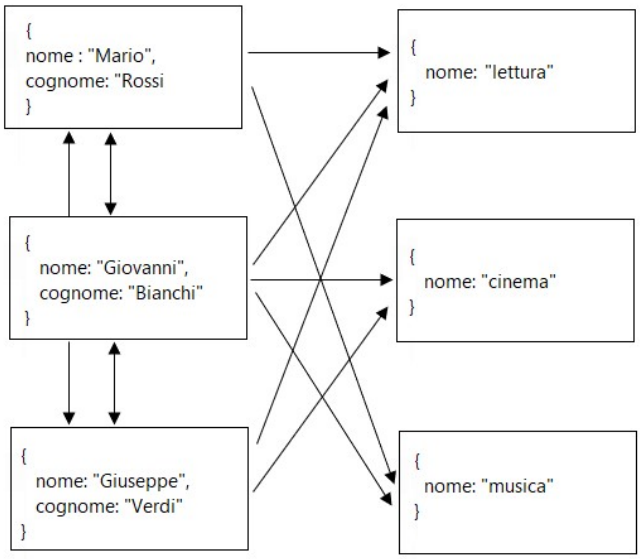
I database di grafici offrono algoritmi specializzati per analizzare le relazioni dei dati

L'algoritmo più semplice è un cosiddetto **graph traversal (**attraversamento del grafico). Un attraversamento del grafico inizia ad attraversare il grafico a partire da un vertice iniziale definito e termina a una profondità definita con il vertice finale.

Quando si applicano filtri durante l'attraversamento di un grafico sulle proprietà di un vertice o di un bordo, viene utilizzato l'algoritmo **pattern matching algorithm** (corrispondenza del modello).

E' possibile anche analizzare la distanza più breve tra due vertici o nodi dati. Questo modello di query è chiamato **shortest path** (percorso più breve).

Un modo semplice per immaginare un grafico è pensare a un social network. In un social network, una persona ha amici e qualcosa che è anche comune è che potrebbero avere altri amici oltre alla persona stessa. Potresti anche essere amico di quegli "altri" amici.



Questa relazione tra te, i tuoi amici e i loro amici fa parte di ciò che forma il tuo social network. Queste connessioni possono essere facilmente tradotte in un grafico e infatti potrebbe essere molto utile per strutturare un social network come un grafico. Tu e i tuoi amici potreste essere rappresentati come vertici individuali (nodi) e quindi le cose che vi legano o descrivono la vostra relazione sarebbero le linee che collegano i nodi.

Quindi il vantaggio più semplice sarebbe la linea che ti collegava a un amico. Tuttavia, cosa accadrebbe se questa connessione facesse un ulteriore passo avanti e descrivesse più cose sulla tua relazione? Potresti includere dettagli che sono comuni tra di voi, come il fatto che entrambi amino la musica e poi quando volevi trovare amici che si unissero a te per andare ad un concerto potresti facilmente richiedere tali informazioni. Ciò consentirebbe cose come suggerire nuovi amici, trovare eventi in base a te e ai tuoi amici che corrispondono agli interessi o persino registrare date importanti come la data in cui sei diventato amico o altri eventi della vita condivisi.

I dettagli che compongono le cose che ti piacciono, le cose che piacciono ai tuoi amici e poi le cose che condividi in comune potrebbero essere considerate proprietà di te e delle tue amicizie. Questo concetto di modellazione dei dati con etichette descrittive è il modo in cui i dati vengono modellati in un grafico delle proprietà. I grafici delle proprietà utilizzano etichette semantiche pertinenti per modellare i dati e le relative connessioni. Ciò significa che i dati possono essere strutturati in un modo facilmente comprensibile per un essere umano. Poiché i dati sono modellati utilizzando termini rilevanti, possono anche essere interrogati in modo facile da leggere.

Database Key Value (Chiave Valore)

Il database chiave-valore si basa su una tabella con solo due colonne: una contiene il valore, l'altra contiene una chiave, un identificatore univoco. Un valore può assumere forme diverse: sono quindi possibili valori molto semplici come stringhe o numeri interi, ma anche oggetti complessi possono comparire come valori nel database. Ad esempio anche un documento può assumere la funzione o il ruolo del valore, nel qual caso si parla dunque di un database orientato ai documenti. Anche i riferimenti ai file possono essere inseriti nel database. È possibile Inoltre implementare le tuple, ovvero le raccolte di valori.

I valori di un database non devono essere uniformi. È possibile ospitare oggetti diversi nella stessa colonna. Lo stesso vale per gli identificatori univoci. Nella maggior parte dei casi la chiave seguirà probabilmente un determinato schema, ma in linea di principio ciò non è necessario. Anche in questo caso stringhe e numeri interi possono essere realizzati in base a un criterio qualsiasi.

Vantaggi e usi del Key Value Database

I vantaggi dei database chiave-valore sono **la performance elevata e la scalabilità flessibile**. Entrambi derivano dalla struttura semplice del modello. Dato che il Key Value Store non richiede o non pretende uno schema omogeneo, è possibile apportare modifiche al database continuamente. È quindi possibile introdurre un nuovo campo durante lo svolgimento contemporaneo di attività in altre voci.

Questo modello di database consente una velocità elevata grazie alla sua **semplice combinazione di chiave e valore**: se si desidera recuperare informazioni, si accede direttamente al valore tramite la chiave specifica. I dati sono direttamente disponibili. Ma questo è allo stesso tempo anche uno degli svantaggi del Key Value Store, in quanto in realtà non è previsto un altro metodo di accesso. In particolare i database relazionali consentono invece query complesse. Il contenuto di questi database può essere cercato in vari modi. Al contrario, in un Key Value Store è previsto solo l'accesso tramite la chiave. In genere si deve rinunciare ad altri indicatori e a funzioni di consultazione.

Le aree di applicazione dei Key Value Store sono determinate sia dai vantaggi che dalle limitazioni dei database. Ogni qualvolta sono richiesti tempi di accesso rapidi con grandi quantità di dati, vengono utilizzati i database chiave-valore. Esempi di simili applicazioni sono **i carrelli degli acquisti online o i dati di sessione**, dove le informazioni sono chiaramente assegnate e devono essere disponibili nel più breve tempo possibile. Per i siti di grandi dimensioni e popolari, questi database devono costantemente creare nuove voci ed eliminare quelle obsolete. I database chiave-valore sono perfettamente adatti a questo scopo.

Principali database chiave-valore

Oggi ci sono diversi sistemi di gestione di basi di dati (DBMS) che si basano su un database chiave-valore.

- **Amazon DynamoDB**: il DBMS proprietario fa parte di Amazon Web Services (AWS) e può essere utilizzato anche come database orientato ai documenti.
- **Berkeley DB**: questo DBMS è sviluppato da Oracle e offre interfacce per i vari linguaggi di programmazione.
- **Redis**: il progetto open-source Redis è uno dei DBMS più diffusi ed è stato utilizzato all'inizio da Instagram e GitHub.
- **Riak**: il DBMS Riak è disponibile in una versione open-source, come soluzione aziendale o sotto forma di memoria cloud.
- **Voldemort**: il DBMS distribuito viene utilizzato e promosso tra gli altri da LinkedIn.

Database a colonne

Un archivio dati a colonne consente di organizzare i dati in righe e colonne. Nella sua forma più semplice, un archivio dati a colonne può risultare molto simile a un database relazionale, almeno a livello concettuale. L'efficacia di un database a colonne sta nell'approccio denormalizzato per strutturare i dati di tipo sparse, che hanno origine da un approccio all'archiviazione dei dati orientato alle colonne. Utilizza tabelle, righe e colonne, ma a differenza di un database relazionale, i nomi e il formato delle colonne possono variare da riga a riga nella stessa tabella. Un archivio a colonne larghe (**wide-columns**) può essere interpretato come un archivio chiave-valore bidimensionale.

È possibile considerare un archivio dati a colonne come contenente dati tabulari con righe e colonne, ma le colonne sono suddivise in gruppi, noti come famiglie di colonne. Ogni famiglia di colonne contiene un set di colonne logicamente correlate tra loro e, in genere, recuperate o modificate come un'unità. Altri dati di cui si accede separatamente possono essere archiviati in famiglie di colonne separate. All'interno di una famiglia di colonne, è possibile aggiungere nuove colonne in modo dinamico e le righe possono essere di tipo sparse, vale a dire una riga non deve necessariamente avere un valore per ogni colonna.

Nel diagramma seguente viene illustrato un esempio con due famiglie di colonne, Identity e Contact Info. I dati per una singola entità contengono la stessa chiave di riga in ogni famiglia di colonne. Questa struttura, in cui le righe per un determinato oggetto in una famiglia di colonne possono variare in modo dinamico, rappresenta un vantaggio importante dell'approccio a colonne, rendendo questa forma di archivio dati molto adatta per l'archiviazione di dati con schemi variabili.

CustomerID	Column Family: Identity
001	**First name:** Mu Bae **Last name:** Min
002	**First name:** Francisco **Last name:** Vila Nova **Suffix:** Jr.
003	**First name:** Lena **Last name:** Adamcyz **Title:** Dr.

CustomerID	Column Family: Contact Info
001	**Phone number:** 555-0100 **Email:** someone@example.com
002	**Email:** vilanova@contoso.com
003	**Phone number:** 555-0120

A differenza di un archivio chiave/valore o un di database a documento, la maggior parte dei database a colonne archivia fisicamente i dati in ordine di chiave, anziché calcolando un hash. La chiave di riga viene considerata l'indice primario e consente l'accesso basato su chiave tramite una chiave specifica o un intervallo di chiavi. Alcune implementazioni consentono di creare indici secondari su colonne specifiche in una famiglia di colonne. Gli indici secondari consentono di recuperare i dati in base al valore delle colonne, anziché della chiave di riga.

Su un disco, tutte le colonne all'interno di una famiglia di colonne vengono archiviate nello stesso file, con un numero di righe specifico in ogni file. Con set di dati di grandi dimensioni, questo approccio consente un miglioramento delle prestazioni riducendo la quantità di dati che devono essere letti dal disco quando vengono eseguite query solo su un numero ridotto di colonne.

Le operazioni di lettura e scrittura per una riga sono in genere atomiche in una singola famiglia di colonne, anche se alcune implementazioni offrono l'atomicità sull'intera riga, con estensione su più famiglie di colonne.

A differenza dei database relazionali, i database a colonne archiviano i propri dati per colonne, anziché per righe. Queste colonne vengono raccolte per formare sottogruppi.

Le chiavi e i nomi delle colonne di questo tipo di database non sono fissi. Le colonne all'interno della stessa famiglia di colonne, o cluster di colonne, possono avere un diverso numero di righe e possono contenere diversi tipi di dati e nomi.

Questi database vengono utilizzati più spesso quando è necessario un modello di dati di grandi dimensioni. Sono molto utili per i data warehouse o quando sono necessarie prestazioni elevate o la gestione di query intensive.

Principali database wide-column

- **Apache Cassandra**: è un sistema di gestione di database NoSQL gratuito e open source, distribuito, con archivio a colonne ampie, progettato per gestire grandi quantità di dati su molti server di base. Cassandra offre supporto per cluster che si estendono su più datacenter, con replica asincrona senza master che consente operazioni a bassa latenza per tutti i client.
- **ScyllaDB,** ScyllaDB è un datastore NoSQL distribuito open source a colonne larghe. È stato progettato per essere compatibile con Apache Cassandra, ottenendo velocità effettiva significativamente più elevate e latenze inferiori.
- **Apache HBase,** HBase è una base di dati distribuita open source modellata su BigTable di Google e scritta in Java.
- **Google BigTable** BigTable è un sistema proprietario di database compresso ad alte prestazioni, sviluppato da Google. È progettato per poter supportare carichi a livello del petabyte attraverso centinaia o migliaia di macchine e di rendere facile l'aggiunta di nuove macchine al sistema, utilizzando le nuove risorse senza che sia necessaria alcuna riconfigurazione.
- **Microsoft Azure Cosmos DB** Azure Cosmos DB è un database NoSQL serverless e completamente gestito per applicazioni ad alte prestazioni di qualsiasi dimensione o scala.