NOSQL

Giulio Nazzicone

giulio.nazzicone@graduate.univaq.it

SQL & RDBMS

SQL (Structured Query Language) è il linguaggio di programmazione standard usato per comunicare con un database relazionale

E' usato per tutte le differenti operazioni che si possono fare su un database, ossia, gestire, archiviare e recuperare dati in database relazionali attraverso applicazioni e query o sulla stessa macchina o su computer in rete
Queste operazioni sono basate sull'algebra relazionale

Un server SQL consiste in un database relazionale che comprende un insieme di tabelle contenenti dati strutturati

Il database relazionale "matcha" i dati tramite caratteristiche comuni che si trovano nel set di dati e il gruppo risultante è definito come schema

Rispettano le proprietà ACID

NoSQL

NoSQL è il nome di un movimento che promuove sistemi software dove la persistenza dei dati è caratterizzata dal non utilizzare il modello relazionale

Il termine fu utilizzato per la prima volta nel 2009 da Eric Evans ma coniato nel 1998 da Carlo strozzi che lo uso' per una base di dati relazionale che non usava una interfaccia SQL

NOSQL

NOSQL

I dati possono essere archiviati su più nodi di elaborazione

Non richiede schemi di tabelle fissi

Evita operazioni di join

Scalabilità tipicamente orizzontale

NOSQL

NoSQL - un sostituto alle limitazioni di SQL

Enorme aumento della quantità di dati da organizzare

Problema di strutturazione dei dati

Approccio strutturato di RDBMS diminuisce di prestazioni all'aumentare dei dati

La scalabilità dei RDBMS non è sufficiente per le esigenze dei Big Data

NOSQL - UN SOSTITUTO ALLE LIMITAZIONI DI SQL

I database NOSQL consentono di raggiungere alte prestazioni e una elaborazione agile dei dati su scala molto grande

E' possibile raccogliere molti più dati e di vario genere con tempi di accesso e analisi di essi molto più veloce

Non è necessario sapere in anticipo esattamente che tipo di dato verrà raccolto nella base di dati

Vantaggi di NoSQL

Elastic scaling

Aumento del carico del database

Con i database relazionali si era costretti ad uno "scaling up", ossia all'acquisto di server più grandi e costosi anziché distribuire il database su più host

Database NOSQL progettati per espandere in modo trasparente e in orizzontale, andando a sfruttare nuovi nodi (anche con hardware di basso costo)

Ogni server può essere rimosso o aggiunto senza compromettere l'inattività delle applicazioni

VANTAGGI DI NOSQL

Bigger data handling capability

La capacità dei database relazionale è cresciuta in corrispondenza dell'aumento del volume dei dati, ma il limite del volume dei dati che un singolo RDBMS puo gestire è intollerabile per alcune imprese

Database NOSQL distribuiti consentono di distribuire i dati su migliaia di server con poca riduzione delle prestazioni superando il volume di dati gestiti da un singolo RDBMS

Vantaggi di NoSQL

Il mantenimento dei server NoSQL è più conveniente

Il mantenimento di sistemi RDBMS di fascia alta è costoso e può esser fatto solo con l'aiuto di DBA altamente qualificati (costi elevati)

I database NoSQL richiedono meno gestione.

Tra le caratteristiche che fanno decrescere i costi:

- riparazione automatica
- distribuzione dei dati più facile
- modelli di dati più semplice

Inoltre, minore ottimizzazione richiesta => minore gestione necessaria

VANTAGGI DI NOSQL

Minore costo di un server

I database NOSQL in genere utilizzano cluster di server a basso costo per la gestione dei volumi di dati che esplodono

I RDBMS tendono a fare affidamento su server proprietari e sistemi di storage costosi

Il costo di memorizzazione ed l'elaborazione per gigabyte nel caso di NOSQL può essere molto minore rispetto al costo nei RDBMS

Vantaggi di NoSQL

No Schema, No fixed data model

Dati inseriti in un database NOSQL senza prima definire uno schema

I dati possono essere modificati in qualsiasi momento, senza l'interruzione delle applicazioni

Immenso aumento della flessibilità per il business dell'applicazione

Al contrario, la gestione di un cambiamento è onerosa nel caso di RDBMS. Qui, anche piccole modifiche al modello di dati devono essere gestite con attenzione e possono richiedere tempi di inattività o livelli di servizio ridotti.

Vantaggi di NoSQL

Caching Integrato

Aumento delle prestazioni nella produzione dei dati anticipata

Al contrario nei database RDBMS deve essere fatto tramite una infrastruttura separata

COSA PERDIAMO CON NOSLQ

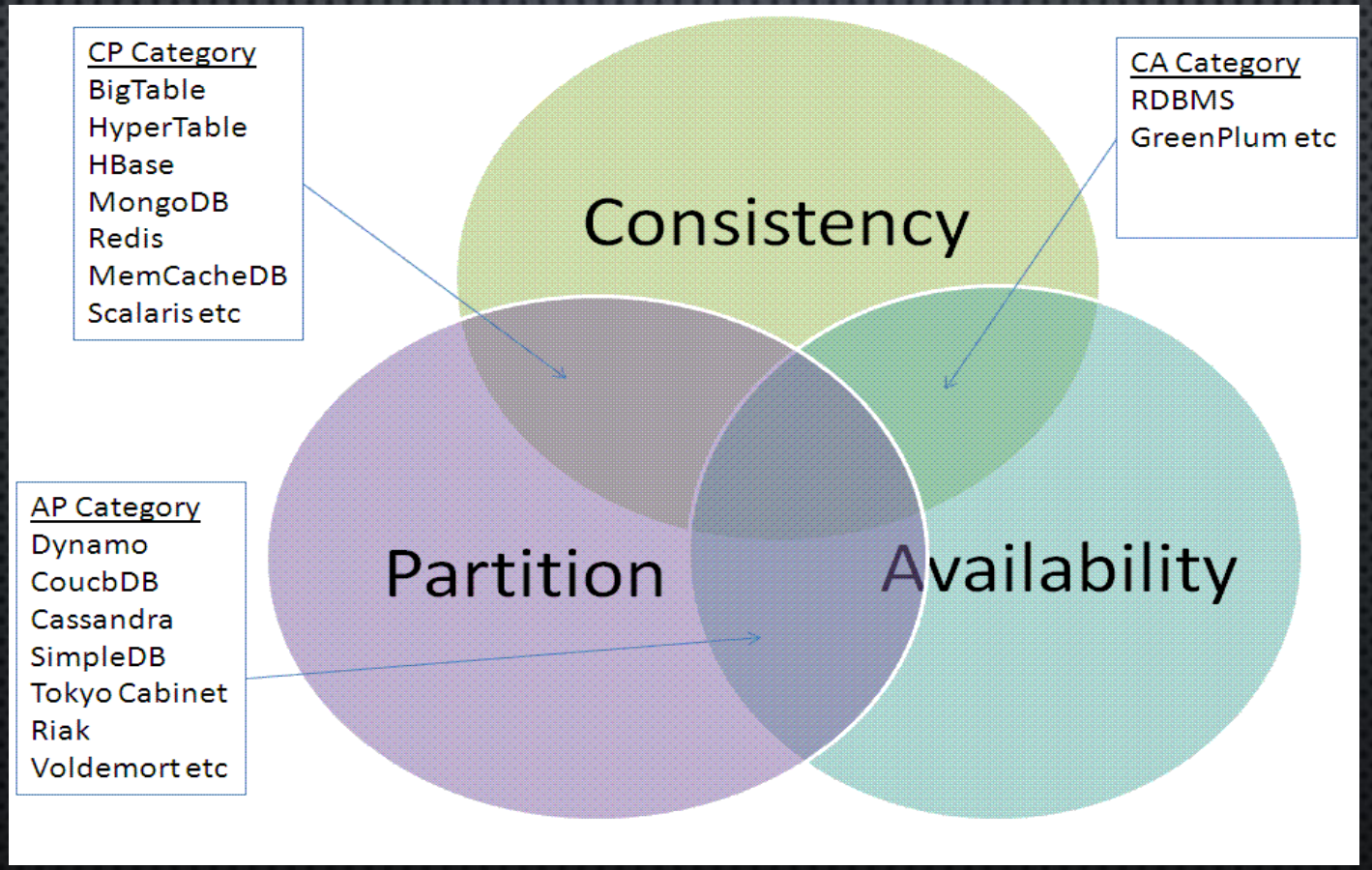
- JOIN
- GROUP BY / ORDER BY
- INDICI
- PROPRIETÀ ACID
- RELAZIONI COMPLESSE
- POTENZA ED ESPRESSIVITÀ DEL LINGUAGGIO SQL
- MATURITÀ
- ASSISTENZA

CAP TEOREM

E' IMPOSSIBILE PER UN SISTEMA INFORMATICO DISTRIBUITO GARANTIRE SIMULTANEAMENTE:

- **CONSISTENCY:** TUTTI I NODI VEDONO GLI STESSI DATI NELLO STESSO MOMENTO
- **AVAILABILITY:** LA GARANZIA CHE OGNI RICHIESTA RICEVA UNA RISPOSTA SIA POSITIVA CHE DI FALLIMENTO
- **PARTITION TOLERANCE:** OGNI NODO DEVE RISPONDERE ANCHE SE CI FOSSE PERDITA DI MESSAGGI TRA DUE NODI

CAP TEOREM


CP Category
BigTable
HyperTable
HBase
MongoDB
Redis
MemCacheDB
Scalaris etc
CA Category
RDBMS
GreenPlum etc
Consistency
AP Category
Dynamo
CoucbDB
Cassandra
SimpleDB
Tokyo Cabinet
Riak
Voldemort etc
Partition
Availability

BASE CONSISTENCY MODEL

BASICALLY **A**VAILABLE, **S**OFT STATE, **E**VENTUAL CONSISTENCE

- BASICALLY AVAILABLE: SE UN NODO FALLISCE, PARTE DEI DATI NON SONO DISPONIBILI, MA L'INTERO SISTEMA CONTINUA A FUNZIONARE
- SOFT STATE: GENERALMENTE STATELESS, SOLO DATI MEMORIZZATI IN CACHE
- EVENTUAL CONSISTENCE: IL SISTEMA POTREBBE DIVENIRE CONSISTENTE

ACID

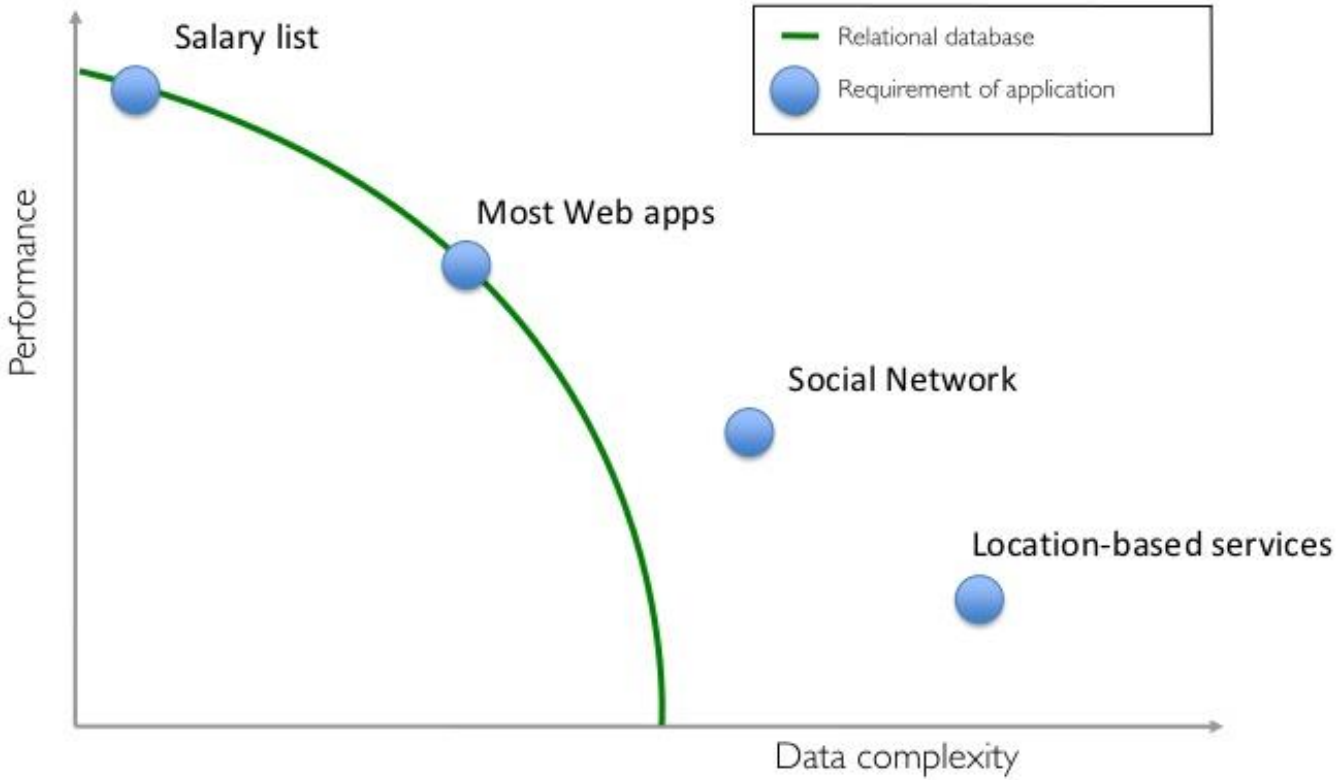
- ATOMICITY
- CONSISTENCY
- ISOLATION
- DURABILITY

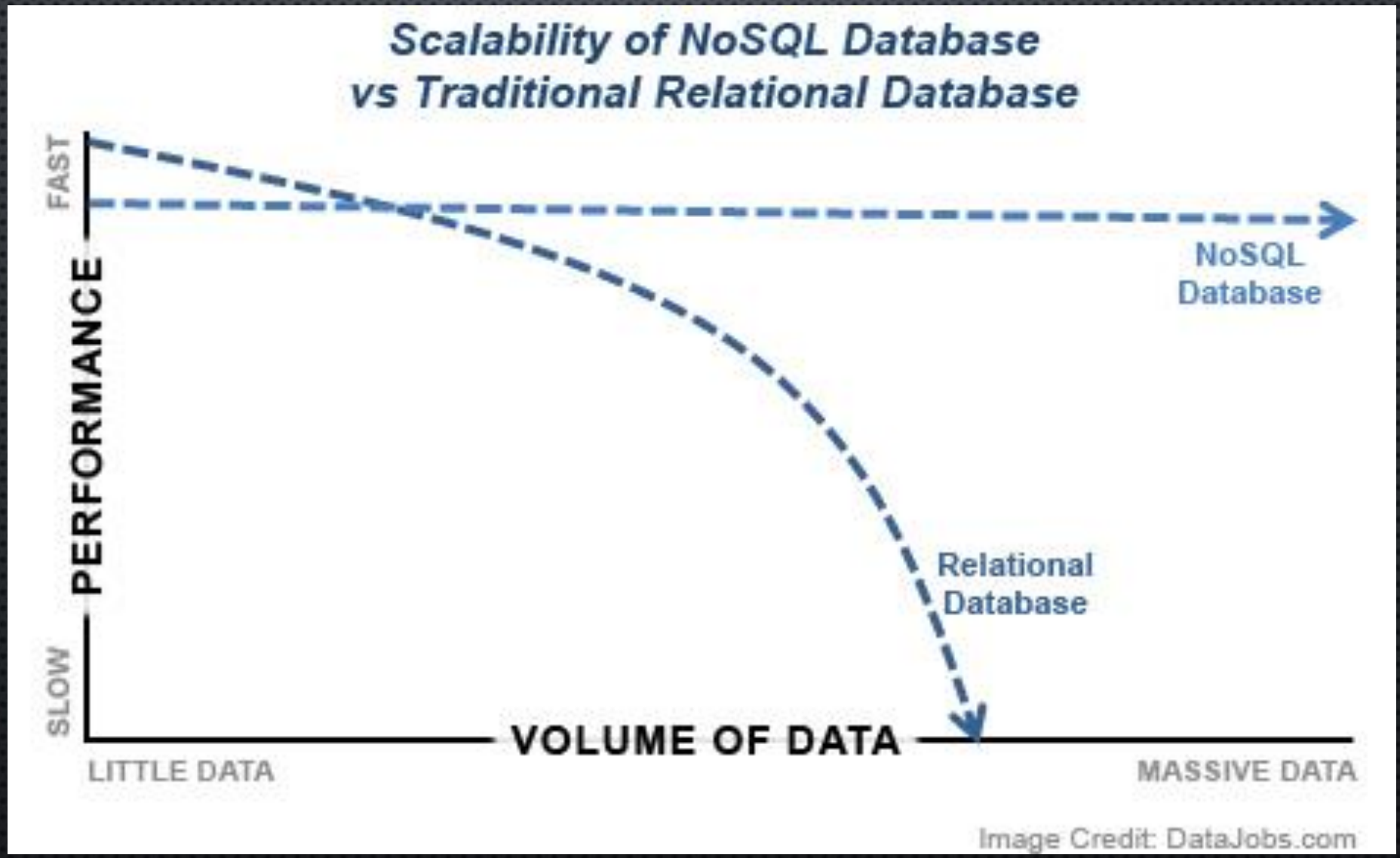
QUANDO UTILIZZARE UN DATABASE NOSQL?

“Considerare sempre se è il caso di utilizzare un database RDBMS per salvare una grande quantità di dati non strutturati”

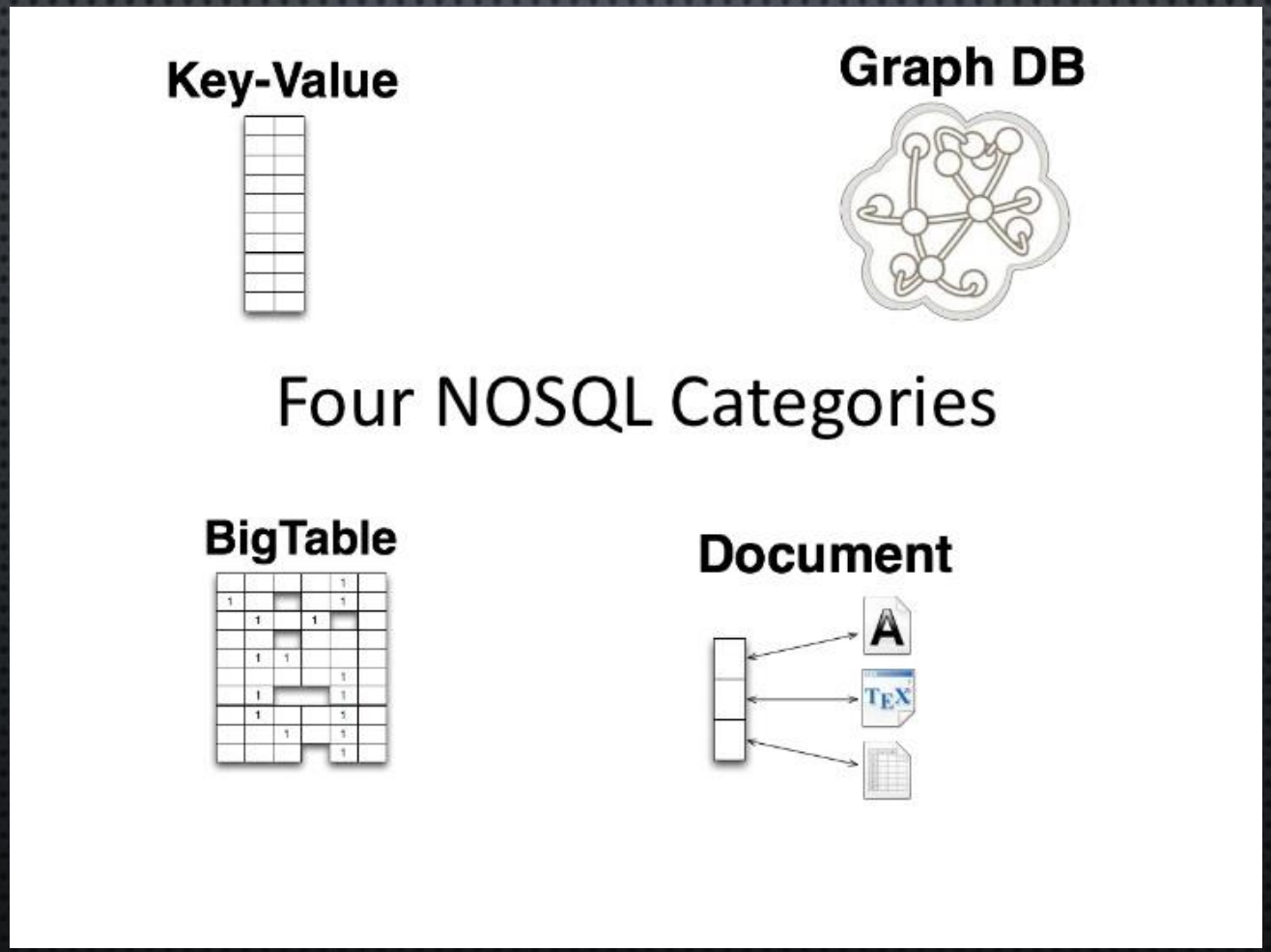
- Salvare un grande quantità di dati non stutturati
  - Log, dati statistici
- Caching di dati per I database lenti
- Denormalizzazione
- Sistemi real time

Side note: RDBMS performance




Scalability of NoSQL Database
vs Traditional Relational Database
FAST
PERFORMANCE
SLOW
NoSQL
Database
Relational
Database
VOLUME OF DATA
LITTLE DATA
MASSIVE DATA
Image Credit: DataJobs.com

NOSQL DATABASE MANAGEMENT SYSTEM



KEY-VALUE

- FORMA PIÙ SEMPLICE E SPINA DORSALE DI DATABASE NOSQL
- MATCHING DI UNA CHIAVE CON UN VALORE COME UN DIZIONARIO
- FOCUS PRINCIPALE: **SCALABILITÀ**

KEY = STRINGA

VALUE = UNA STRINGA RAPPRESENTATE UN VIDEO, UN FILE XML, UN DOCUMENTO DI TESTO, ETC

KEY NAMESPACE PER EVITARE COLLISIONI

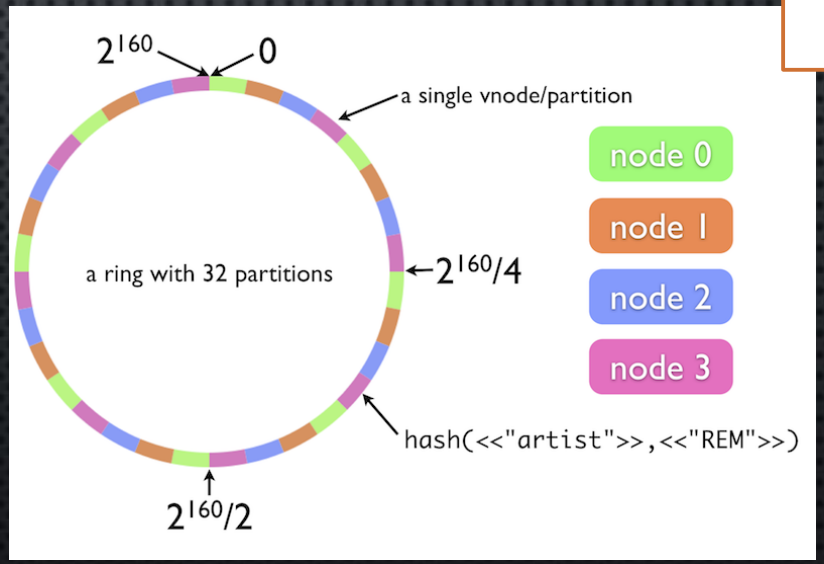
KEY-VALUE PRO&CON

- FACILITÀ DI UTILIZZO
- ALTE PRESTAZIONI
- NESSUN INDICE DA MANTENERE
- ALTA SCALABILITÀ

- NON È POSSIBILE FARE QUERY COMPLESSE
- NO TRANSAZIONI
- DIFFICOLTA DI MODELLAZIONE DI STRUTTURE DATI
- DEVONO STARE IN MEMORIA

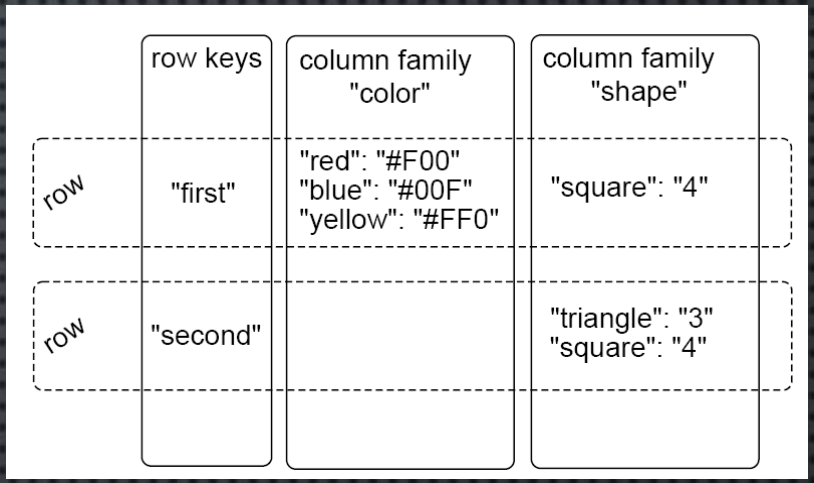
KEY VALUE QUANDO USARLI?

- STOCCAGGIO PREZZI
- STATISTICHE
- REAL-TIME DATA COLLECTION
- COMUNICAZIONE REAL-TIME
- SESSION STORAGE
- CACHE DI ALTRI DB



Implementazioni:
Riak
Redis
Voldemort
Dynamo

COLUMNAR



- FORMA EVOLUTA DEI K/V
- SIMILI AD UN RELAZIONALE MA NON HANNO UNA STRUTTURA PREDEFINITA
- CONSISTONO IN UNA COPPIA CHIAVE E ZERO O PIU VALORI
- OGNI RIGA PUÒ AVERE PIÙ COLONNE CON DIVERSA NATURA DI DATI
- POSSONO ESSERCI COLONNE VUOTE IN UNA RIGA
- OGNI COLONNA PUÒ OSPITARE UN ENORME QUANTITATIVO DI DATI
- L'AGGIUNTA DI UNA COLONNA NON COSTA MOLTO

COLUMNAR PRO&CON

- SOLUZIONE AI 'BIG DATA'
- ALTA DISPONIBILITÀ
- GARBAGE COLLECTION PER I DATI SCADUTI
- SCANSIONE MOLTO RAPIDA

- COSTA NELL'INSERIMENTO
- RICHIEDE QUERY PRE-PIANIFICATE

COLUMNAR QUANDO USARLI?

- SEARCH ENGINE
- HISTORICAL DATA ANALISYS
- DATA WAREHOUSING
- BUSINESS INTELLIGENCE

Implementazioni:
HBase
BigTable
Cassandra
Vertica

DOCUMENT

- FORMA EVOLUTA DI COLUMNAR
- PERMETTONO UN NESTING PIÙ PROFONDO
- SI POSSONO CREARE STUTTURE MOLTO COMPLESSE (JSON)
- I DOCUMENTI POSSONO ESSERE VERSIONATI
- SCEGLIERE IL TIPO DI DATABASE IN BASE ALLE ESIGENZE
  - INDEXING, QUERING, REPLICATION, CONSISTENCY

DUCUMENT PRO&CON

- DATI VARIABILI
- PARADIGMA OBJECT ORIENTED
- CONCORRENZA
- LAVORA BENE CON DATI DENORMALIZZATI

- DIFFICILE FARE QUERY COMPLESSE
- NO JOINS

DOCUMENT QUANDO USARLI?

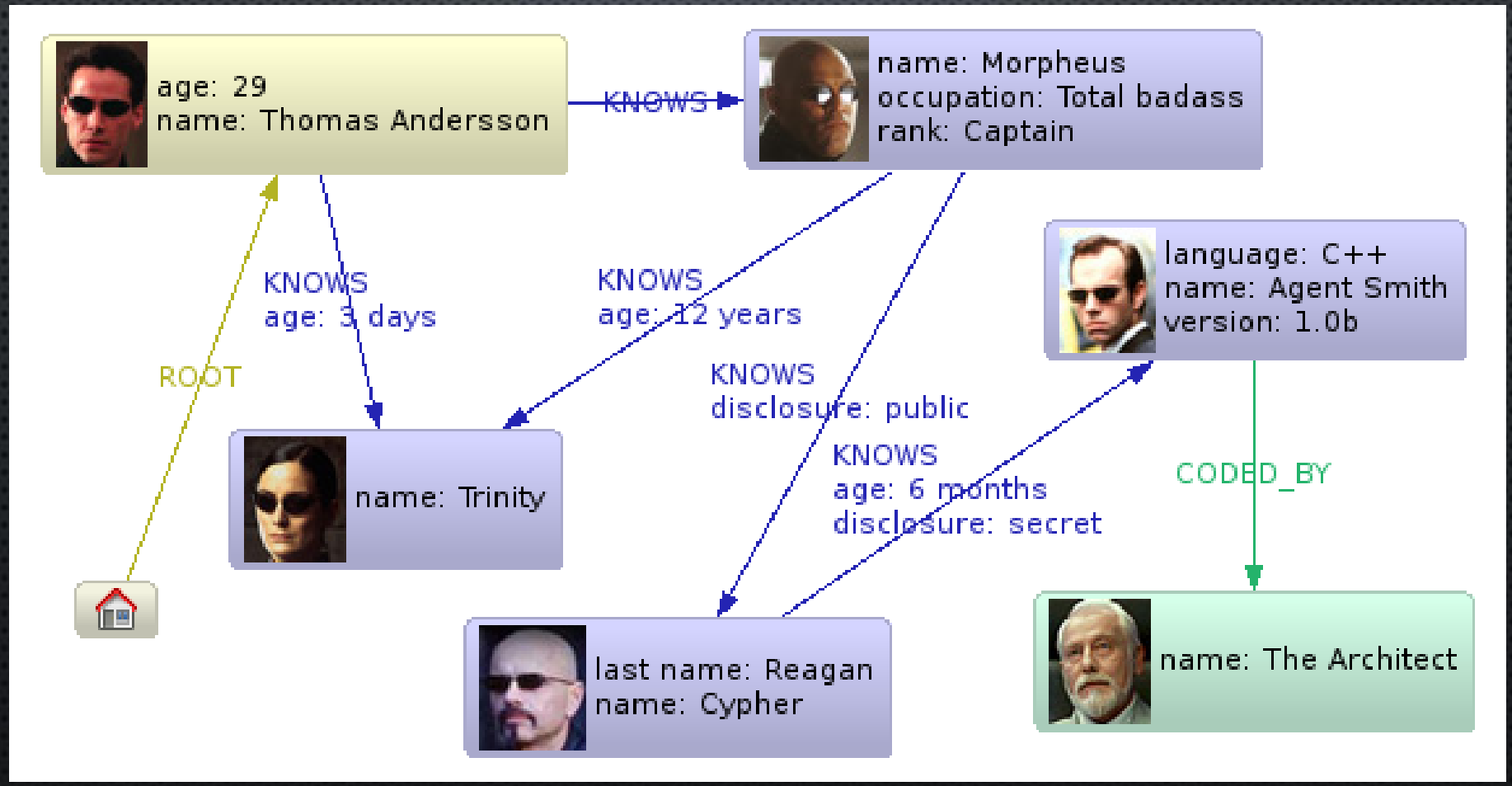
Implementazioni:
MongoDB
CouchDB
RavenDB

- QUANDO È CHIARA A PRIORI LA STRUTTURA DEI DATI CHE ANDREMO A REALIZZARE
- QUANDO ABBIAMO BISOGNO DEL PARADIGMA OBJECT-ORIENTED
- QUANDO È IMPORTANTE IL CONTROLLO DI VERSIONE

GRAPH

- PENSATO PER MODELLARE STRUTTURE DI DATI E LA LORO INTERCONNETTIVITÀ
- ISPIRATO ALLA TEORIA DEI GRAFI $G = (V,E)$
- IL MODELLO DEI DATI È UN GRAFO CON PROPRIETÀ
  - LE ENTITÀ SONO I NODI
  - LE RELAZIONI SONO GLI ARCHI TRA I NODI
- MOLTO POTENTE NEL TRATTAMENTO DI DATI IPERCONNESSI
  - MODELLARE UN GRAFO CON RDBMS È FATTIBILE MA MOLTO ONEROSO

GRAPH


age: 29
name: Thomas Andersson
KNOWS
name: Morpheus
occupation: Total badass
rank: Captain
KNOWS
age: 3 days
KNOWS
age: 12 years
language: C++
name: Agent Smith
version: 1.0b
ROOT
KNOWS
disclosure: public
name: Trinity
KNOWS
age: 6 months
disclosure: secret
CODED_BY
last name: Reagan
name: Cypher
name: The Architect