NoSQL: Esame A.A. 2024-25

23 giugno 2025

1 DB a famiglie di colonne (3 punti)

(a) Spiegare brevemente l'uso di colonne senza valore (*valueless columns*) nelle basi di dati a famiglie di colonne.

(b) Si consideri un sistema di *social media* che deve tenere traccia dei "followers" di ogni utente. Spiegare come si possono usare le *valueless columns* a questo scopo, fornendo una struttura del DB colonnare utile, e illustrando brevemente i vantaggi rispetto a un DB relazionale.

2 CAP Theorem (4 punti)

Si consideri uno scenario in cui il sistema informativo di una banca gestisce i conti correnti attraverso due server principali:

- Server S1 (Milano): Gestisce le operazioni di scrittura (prelievi, depositi etc.).
- Server S2 (Napoli): Gestisce le operazioni di lettura (consultazione saldo, estratti conto etc.).

Il conto corrente del cliente Gaetano Scandurra ha un saldo di 1.500€; detto cliente effettua un prelievo di 200,00€ presso uno sportello automatico collegato al server S1. Contemporaneamente si verifica una partizione di rete tra Milano e Napoli che dura 10 minuti, impedendo la sincronizzazione tra i due server. Subito dopo la moglie del cliente, cointestataria del conto, controlla il saldo tramite applicazione mobile collegata al server S2.

(a) Spiegare brevemente e intuitivamente il CAP Theorem usando come esempio lo scenario descritto.

(b) Tra i due approcci principali AP e CP, quale è preferibile nello questo scenario dato? Giustificare brevemente la risposta.

3 DB Documentali (6 punti)

Un sistema informativo deve contenere dati relativi a categorie di prodotti e prodotti di un negozio online; ogni prodotto appartiene a una singola categoria, mentre ogni categoria può contenere molti prodotti. La categoria "Elettronica" ha nome visualizzato "Dispositivi Elettronici" e descrizione "Smartphone, tablet e accessori tecnologici"; la categoria "Abbigliamento" ha nome visualizzato "Moda e Stile" e descrizione "Vestiti per ogni occasione".

Nella categoria "Elettronica" ci sono i seguenti prodotti: "iPhone 15" con prezzo 899,99€ e 15 unità disponibili, "Samsung Galaxy S24" con prezzo 749,99€ e 8 unità disponibili. Nella categoria "Abbigliamento" ci sono: "Maglietta" con prezzo 19,99€ e 50 unità disponibili, "Jeans Slim Fit" con prezzo 79,99€ e 12 unità disponibili, "Scarpe Casual" con prezzo 129,99€ e 3 unità disponibili.

Si prevede inoltre che saranno eseguite le seguenti operazioni:

Q1: Dato l'identificativo di una categoria, mostrare il nome visualizzato, la descrizione, il numero totale di prodotti, il prezzo medio dei prodotti e il valore totale dell'inventario della categoria. (12.000 interrogazioni al giorno)

Q2: Dato l'identificativo di un prodotto, trovare nome, prezzo e disponibilità del prodotto. (15.000 interrogazioni al giorno)

W1: Inserire un nuovo prodotto con nome, prezzo e disponibilità in una categoria esistente. (50 inserimenti al giorno)
W2: Aggiornare la disponibilità di un prodotto esistente. (200 aggiornamenti al giorno)

Si illustri come memorizzare i dati di esempio, rappresentandoli in JSON[1], considerando le operazioni e la loro frequenza. Si spieghi brevemente l'uso di eventuali denormalizzazioni. Giustificare in modo conciso tutte le scelte effettuate e le risposte fornite.

4 Graph DBs (6 punti)

Si vogliono rappresentare squadre di calcio e relazioni tra di esse; in particolare quali squadre sono gemellate con altre; inoltre si rappresentano tifosi di dette squadre e le amicizie tra di loro. Pozzuoli FC è gemellata con AC Posillipo dal 2010; AC Posillipo è gemellata con Castellammare dal 2002; Castellammare è gemellata con Pozzuoli FC dal 1999. Gennaro Scandurra tifa per Pozzuoli FC ed è amico di Francesco Scannagatta il quale tifa Castellammare; Vincenzo Fusco è amico di Francesco Scannagatta e tifa anch'egli Castellammare.

(a) Rappresentare i dati sopra descritti in un *property graph* (fornire solo una rappresentazione grafica del grafo, senza il codice Cypher per crearlo).
(b) Scrivere in Cypher una query che trova i nomi dei tifosi che *(i)* sono amici di amici di Francesco Scannagatta (senza contare Francesco stesso), e *(ii)* tifano per squadre gemellate con AC Posillipo.

5 RDF (5 punti)

Si consideri il grafo RDF definito nella Sezione 7 (si ricordi che `a` è abbreviazione di `rdf:type`).

(a) Si illustri brevemente l'uso e l'utilità dei prefissi standard del W3C.
(b) Si formuli una interrogazione SPARQL che trova tutte le persone coi loro nomi e, se ce l'hanno, il corso di laurea a cui sono iscritte.
(c) Si formuli una interrogazione SPARQL che trova tutti gli studenti che hanno sostenuto almeno un esame con voto superiore alla media generale di tutti gli esami (sostenuti da tutti gli studenti).

6 Datalog (6 punti)

Supponiamo che in un database di un'agenzia di viaggi ci sia una tabella con lo schema

`flight(from, to, airline)`

dove una tupla (c_1, c_2, ℓ) significa che è possibile volare (con una singola tratta) dalla città c_1 alla città c_2 con la compagnia aerea ℓ.

Considerare le seguenti due interrogazioni:

1. Restituire tutte le coppie di città (X, Y) tali che sia possibile viaggiare da X a Y usando voli di non piú di una compagnia aerea (quindi di una sola compagnia).
2. Restituire tutte le coppie di città (X, Y) tali che sia possibile viaggiare da X a Y usando voli di non piú di due compagnie aeree.

Per ciascuna interrogazione, spiegare se essa è esprimibile in Datalog o meno. Se lo è, scrivere l'interrogazione in Datalog, altrimenti spiegare perché non è esprimibile.

[1] Qui non è necessario rappresentare i dati come in MongoDB, con identificativi e tipi di dato specifici; si usi una semplice sintassi JSON.

7 Grafo in RDF

Prefissi:

```
@prefix uni: <http://esempio.org/universita#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

Ontologia:

```
# Classi con gerarchia

foaf:Person a rdfs:Class ;
    rdfs:label "Persona" .

uni:Studente a rdfs:Class ;
    rdfs:subClassOf foaf:Person ;
    rdfs:label "Studente" .

uni:Docente a rdfs:Class ;
    rdfs:subClassOf foaf:Person ;
    rdfs:label "Docente" .

uni:Corso a rdfs:Class ;
    rdfs:label "Corso" .

uni:Esame a rdfs:Class ;
    rdfs:label "Esame" .
```

Grafo:

```
# Persone generiche (non studenti né docenti)

uni:segretario a foaf:Person ;
    foaf:name "Gennaro Locascio" .

# Docenti
uni:prof_rossi a uni:Docente ;
    foaf:name "Mario Rossi" .

# Studenti
uni:anna a uni:Studente ;
    foaf:name "Anna Verdi" .

uni:luca a uni:Studente ;
    foaf:name "Luca Bianchi" .


# Corsi
uni:informatica a uni:Corso ;
    foaf:name "Informatica" ;
    uni:crediti 6 .
```

```
uni:matematica a uni:Corso ;
    foaf:name "Matematica" ;
    uni:crediti 9 .

# Esami (istanze di esame con voto)

uni:esame1 a uni:Esame ;
    uni:voto 28 ;
    uni:esamePerCorso uni:informatica .

uni:esame2 a uni:Esame ;
    uni:voto 30 ;
    uni:esamePerCorso uni:matematica .

uni:esame3 a uni:Esame ;
    uni:voto 25 ;
    uni:esamePerCorso uni:informatica .

# Relazioni

uni:prof_rossi uni:insegna uni:informatica, uni:matematica .
uni:anna uni:iscrittoA uni:matematica .
uni:luca uni:iscrittoA uni:informatica .

uni:anna uni:haEsame uni:esame1, uni:esame2 .
uni:luca uni:haEsame uni:esame3 .
```