NoSQL: Esame A.A. 2024-25, 12 settembre 2025

1 DB Distribuiti e Consistenza (6 punti)

(a) Fornire una dimostrazione intuitiva del *CAP Theorem*, aiutandosi eventualmente con una figura.

(b) Fornire un esempio di applicazione, con un esempio di dati e transazioni in presenza di una partizione di rete, in cui l'approccio CP è preferibile a quello AP.

2 DB Documentali (6 punti)

Un sistema informativo deve contenere dati relativi a clienti di un'azienda che vende prodotti e reclami da essi presentati[1]; ogni reclamo è presentato da un unico cliente; ogni cliente ovviamente può presentare piú reclami. Il cliente Francesco Scannapieco abita a Napoli e ha numero di telefono 081-2292929; il cliente Gaetano Scandurra ha 33 anni, abita a San Gennaro Vesuviano e ha indirizzo di posta elettronica `gsca@scandurra.com`. Il cliente Francesco Scannapieco ha presentato un reclamo il 27-08-2023, per il prodotto con codice identificativo P123 con motivo "Pacco mai arrivato", e un altro il 21-09-2023 con motivo "Numero clienti risulta occupato". Il cliente Gaetano Scandurra ha presentato un reclamo il 12-12-2022, per il prodotto P321, con motivo "Prodotto difettoso".

Si stima che la base di dati dovrà contenere dati come quelli di cui sopra, con circa 100.000 clienti e 20.000 reclami. Si prevede inoltre che saranno eseguite le sequenti operazioni con relative frequenze stimate:

Q1: Dato l'identificativo di un reclamo, trovare la motivazione del reclamo nonché l'identificativo e il nome del cliente che ha presentato il medesimo reclamo. *(8.000 interrogazioni al giorno)*

Q2: Dato l'identificativo di un cliente, trovare l'identificativo di tutti i reclami presentati dal cliente in questione. *(5.000 interrogazioni al giorno)*

W1: Dati nome, numero di telefono, città di residenza ed eventualmente l'età di un cliente, inserire i dati di tale cliente. *(10 inserimenti al giorno)*

Si illustri come memorizzare i dati di esempio, rappresentandoli in JSON[2], 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.

3 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 AC Posillipo ~~Castellammare~~; Antimo Scannapieco è amico di Vincenzo Fusco e tifa Castellammare.

(a) Rappresentare i dati sopra descritti in un *property graph* (fornire solo una rappresentazione grafica del grafo, senza il codice Cypher per crearlo).

[1] In questa specifica usiamo dati parziali di esempio piuttosto che dati realistici, come è ovvio nel contesto di questo esercizio. Nel nostro esempio non rappresentiamo i prodotti con documenti, ma ne utilizziamo solo i codici identificativi.

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

CE

dato

(b) Scrivere in Cypher una query che conta quanti giocatori ha ciascuna squadra che ha come tifosi: *(i)* Francesco Scannagatta, e *(ii)* un amico di un amico di Francesco Scannagatta (senza contare Francesco Scannagatta medesimo).

4 RDF (5 punti)

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

studenti

(a) Si formuli una interrogazione SPARQL che trova tutti i corsi di laurea, e per ciascuno di essi tutti i ~~corsi~~ relativi.

(b) Si formuli una interrogazione SPARQL che trova tutti gli studenti che hanno sostenuto almeno un esame di un corso insegnato dal Prof. Mario Rossi con voto superiore a 26.

5 Datalog (7 punti)

Sia dato un database D espresso con i seguenti fatti Datalog.

```
coniugi(enzo,concetta).
genitore(enzo,pasquale).
genitore(concetta,carmela).
coniugi(carmela,ciro).
```

Il predicato `genitore` definisce il fatto che due persone sono genitore e figlio; per esempio `genitore(enzo,francesca)` significa che Enzo è genitore di Francesca. `coniugi(enzo,concetta)` significa che Enzo e Concetta sono coniugi.

(a) Scrivere un programma Datalog, definendo ove opportuno dei nuovi predicati IDB, che abbia le seguenti proprietà:

- Il predicato `coniugi` è simmetrico (ciò va espresso con una regola Datalog per il predicato `coniugi`).
- Il predicato binario `nu_ge` esprime che una persona è genero o nuora di un'altra, cioè che l'una è coniuge del figlio (o figlia) dell'altra.

(b) Si assuma di voler definire un predicato unario (cioè con un argomento) `1figlio` che definisce le persone che hanno esattamente un figlio. Spiegare se è possibile definire in Datalog un tale predicato; in caso di risposta affermativa, definire il predicato con opportune regole; altrimenti, spiegare brevemente perché la query non è formulabile.

No, vincoli di cardinalità inammissibili per omomorfismo e monotonicità

A 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 .
```