UNIVERSITÀ DEGLI STUDI ROMA TRE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

Tesi Di Laurea

Questo è il titolo della tesi

Laureando
Marco Faretra
Matricola 460573

Relatore
Prof. Paolo Atzeni

Correlatore
Ing. Luigi Bellomarini

Anno Accademico 2016/2017

Questa è la dedica

Ringraziamenti

Grazie a tutti

Introduzione

Negli ultimi tempi, la mole di dati all'interno delle aziende cresce quotidianamente, per questo motivo molte compagnie desiderano mantenere i propri dati in knowledge graph, che per l'utilizzo e la gestione di tale strumento si necessita di un Knowledge Graph Management System (KGMS).

Fin dagli anni 70, l'importanza della conoscenza è stata evidente, e l'idea di salvare conoscenza e di elaborarla per trarre nuova conoscenza esisteva già da allora. Il collo di bottiglia era la tecnologia di quei tempi, gli hardware erano troppo lenti, la memoria principale troppo piccola; i DBMS erano troppo lenti e rigidi; Non era presente un web dove un sistema esperto poteva acquisire dati; il machine learning e le reti neurali furono ridicolizzate e non riuscite.

Con il passare degli anni, l'avvento tecnologico ha subito una crescita radicale, l'hardware si è evoluto, le tecnologie dei database sono migliorate notevolmente, è presente un web con dati aperti, le aziende possono partecipare sui social networks. La ricerca ha portato ad una comprensione migliore di molti aspetti nell'elaborazione della conoscenza e reasoning su grandi quantità di dati.

A causa di tutto ciò, migliaia di grandi e medie aziende, desiderano gestire i propri knowledge graph e cercano adeguati KGMS. Inizialmente soltanto le grandi aziende ad esempio Google (che utilizza il proprio knowledge graph per il proprio motore di ricerca), Amazon, Facebook, ecc... ne possedevano uno, ma con il passare degli anni molte aziende medio/basse desiderano avere un knowledge graph aziendale privato, che contiene molti dati in forma di fatti, come ad esempio conoscenza su clienti, prodotti, prezzi, concorrenti, piuttosto che conoscenza di tutto il mondo da Wikipedia o altre fonti simili.

Un KGMS completo deve svolgere compiti complessi di reasoning, ed allo stesso tempo ottenere performance efficienti e scalabili sui Big Data con una complessità computazionale accettabile. Inoltre necessita di interfacce con i database aziendali, il web e il machine learning. Il core di un KGMS deve fornire un linguaggio per rappresentare la conoscenza e il resoning.

Vadalog rappresenta un sistema KGMS, che offre un motore centrale di reasoning principale ed un linguaggio per la gestione e l'utilizzo.
Il linguaggio Vadalog appartiene alla Famiglia Datalog±, che estende Datalog con quantificatori esistenziali nelle teste delle regole, nonché da altre caratteristiche ed allo stesso tempo limita la sua sintassi in modo da ottenere decidibilità e tracciabilità dei dati.
Datalog [Wika] è un linguaggio di interrogazione per basi di dati che ha riscosso un notevole interesse dalla metà degli anni ottanta, è basato su regole di deduzione.
Il core logico di Vadalog è in grado di processare tale linguaggio ed è in grado di eseguire task di reasoning ontologici e risulta computazionalmente efficiente, tale da soddisfare i requisiti già citati (Big Data, Web, Machine Learning, ...), esso ha accesso ad un repository di regole. Per dare un esempio consente l'aggregazione attraverso la somma, il prodotto, il massimo, ecc... anche in presenza di ricorsioni.
Esso fornisce anche degli strumenti che permettono il data analytics, l'iniezione di codice procedurale, l'integrazione con diverse tipologie di input (ad esempio database relazionali, file csv, ecc...).

L'obiettivo della mia Tesi è stato l'ampliamento di Vadalog con nuove features.
Di seguito un accenno delle funzionalità di cui mi sono principalmente occupato, che verrà descritto in maniera più approfondita nei prossimi capitoli:

- Implementazione di nuovi tipi di dato.
- Riscritture per ottimizzare i tempi di calcolo (ad esempio "Push Selection Down").
- Creazione di benchmark per effettuare test sulle performance.
- Supporto di nuove funzionalità (ad esempio, supporto ai csv, supporto alle funzioni arbitrarie, ecc...).

- Integrazione di codice procedurale all'interno del linguaggio Vadalog.

L'attività di Tesi è stata svolta presso il Laboratorio Basi di Dati, dell'Università degli Studi Roma Tre, in collaborazione con L'Università di Oxford.

Indice

Elenco delle figure

Capitolo 1

Vadalog Engine

1.1 Proprietà di un KGMS

Un KGMS completo deve disporre di diversi requisiti, che elencheremo sulla base di tre categorie principali, descritte nei sottocapitoli 1.1.1, 1.1.2 ed 1.1.3

1.1.1 Linguaggio e sistema per il reasoning

Dovrebbe esistere un formalismo logico per esprimere fatti e regole, e di un engine per il reasoning che usa tale linguaggio che dovrebbe fornire le seguenti caratteristiche:

- Sintassi semplice e modulare: Deve essere semplice aggiungere e cancellare fatti e nuove regole. I fatti devono coincidere con le tuple sul database.
- Alta potenza espressiva: Datalog è un buon punto di riferimento per il potere espressivo delle regole. Con una negazione molto lieve cattura PTIME.
- Calcolo numerico e aggregazioni: Il linguaggio deve essere arricchito con funzioni di aggregazione per la gestione di valori numerici.
- Probabilistic Reasoning: Il linguaggio dovrebbe essere adatto ad incorporare metodi di reasoning probabilistico e il sistema dovrebbe propagare probabilità o valori di certezza durante il processo di reasoning.

- Bassa complessità: Il reasoning dovrebbe essere tracciabile nella complessità dei dati. Quando possibile il sistema dovrebbe essere in grado di riconoscere e trarre vantaggio da set di regole che possono essere elaborate in classi di complessità a basso livello di spazio.

- Rule repository, Rule management and ontology editor: È necessario fornire una libreria per l'archiviazione di regole e definizioni ricorrenti, ed un'interfaccia utente per la gestione delle regole.

- Orchestrazione dinamica: Per applicazioni più grandi, deve essere presente un nodo master per l'orchestrazione di flussi di dati complessi.

1.1.2 Accesso e gestione dei Big Data

- Accesso ai Big Data: Il sistema deve essere in grado di fornire un accesso efficace alle sorgenti e ai sistemi Big Data. L'integrazione di tali tecniche dovrebbe essere possibile se il volume dei dati lo rende necessario.

- Accesso a Database e Data Warehouse: Dovrebbe essere concesso un accesso a database relazionali, graph databases, data warehouse, RDF stores ed i maggiori NoSQL stores. I dati nei vari storage dovrebbero essere direttamente utilizzabili come fatti per il reasoning.

- Ontology-based Data Access (OBDA): Consente ad un sistema di compilare una query che è stata formulata in testa ad un'ontologia direttamente all'interno del database.

- Supporto multi-query: Laddove possibile e appropriato, i risultati parziali di query ripetute dovrebbero essere valutati una volta e ottimizzati a questo proposito.

- Pulizia dei dati, Scambio e Integrazione: L'integrazione, la modifica e la pulizia dei dati dovrebbero essere supportati direttamente (attraverso il linguaggio).

- Estrazione di dati web, Interazione e IOT: Un KGMS dovrebbe essere in grado di interagire con il web mediante estrazione dei dati web rilevanti (prezzi pubbliciz-

zati dai concorrenti) e integrandoli in database locali e scambiare dati con moduli e server web disponibili (API).

1.1.3 Inserimento di codice procedurale e di terze parti

- Codice procedurale: Il sistema dovrebbe disporre di metodi di incapsulamento per l'incorporazione di codice procedurale scritti in vari linguaggi di programmazione e offrire un'interfaccia logica ad esso.

- Pacchetti di terze parti per il machine learning, text mining, NLP, Data Analytics e Data Visualization: Il sistema dovrebbe essere dotato di accesso diretto a potenti package esistenti per il machine learning, text mining, data analytics e data visualization. Esistono diversi software di terze parti per questi scopi, un KGMS dovrebbe essere in grado di utilizzare una moltitudine di tali pacchetti tramite opportune interfacce logiche.

Capitolo 2

Il linguaggio Vadalog

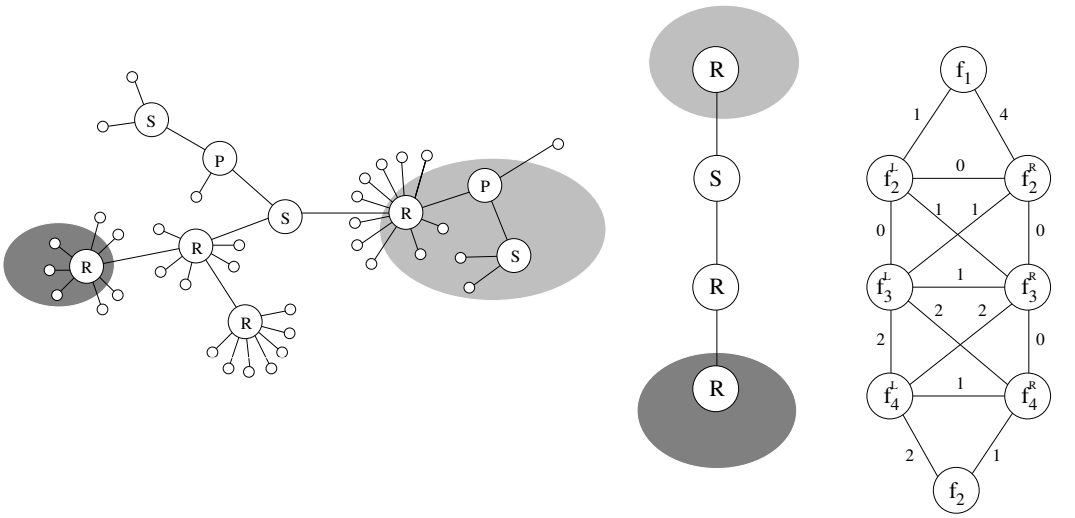
2.1 Questa è una Sezione

Prova di testo di capitolo. Vorrei citare qui tutta l'opera omnia di [oEE90, Wikb, Box97, AHPZ96].

2.1.1 Questa è una Sottosezione

Ancora del testo

Come si evince dalle Figure 2.1.a e 2.1.b non si capisce molto.



(a) (b)

Figura 2.1: SPQR-tree di un grafo. (a) L'albero di allocazione della faccia esterna. (b) Il cammino notevole di cui si parla tanto nella Sezione 2.1.

Capitolo 3

Algoritmi

Capitolo 4

Prove sperimentali

Capitolo 5

Related Work

Conclusioni e sviluppi futuri

La tesi è finita

Bibliografia

[AHPZ96] Eric Andonoff, Gilles Hubert, Annig Le Parc, and Gilles Zurfluh. Integrating versions in the omt models. In *ER '96: Proceedings of the 15th International Conference on Conceptual Modeling*, pages 472–487, London, UK, 1996. Springer-Verlag.

[Box97] D. Box. *Essential COM*. Addison Wesley Professional, 1997.

[JS96] Trevor H. Jones and Il-Yeol Song. Analysis of binary/ternary cardinality combinations in entity-relationship modeling. *Data Knowledge Engineering*, 19(1):39–64, 1996.

[Lar05] C. Larman. *Applicare UML e i pattern - Analisi e progettazione orientata agli oggetti - 3a Edizione*. Prentice Hall, 2005.

[oEE90] Institute of Electrical and Electronics Engineers. Ieee standard computer dictionary: A compilation of ieee standard computer glossaries, 1990.

[Wika] Wikipedia. `https://it.wikipedia.org/wiki/Datalog`.

[Wikb] Wikipedia. `http://en.wikipedia.org/wiki/Interoperability`.