

UNIVERSITÀ
DEGLI STUDI
DI PADOVA



DIPARTIMENTO
DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Analisi e mappatura dei dati Medical Subject Headings (MeSH) in un database a grafo Neo4j

Relatore: Prof. Giorgio Maria Di Nunzio

Laureanda: Giorgia Bortoletti (1168157)

ANNO ACCADEMICO 2019-2020

Data di laurea 25/09/2020

Indice

1	Introduzione	7
2	Database a grafo e Neo4j	9
2.1	Database a grafo.....	10
2.2	Database a grafo e database relazionale	11
2.3	Neo4j	12
2.3.1	Cypher	13
3	Analisi dei dati MeSH	16
3.1	Dati MeSH.....	17
3.2	Requisiti strutturali.....	21
3.3	Glossario	24
4	Progettazione e realizzazione del Database	27
4.1	Progettazione Concettuale.....	27
4.1.1	Modello Concettuale.....	28
4.1.2	Ristrutturazione schema	28
4.1.3	Dizionario dei dati	29
4.2	Progettazione Logica	32
4.2.1	Modello Logico: Relazionale	33
4.2.2	Regole di Vincolo.....	33
4.3	Codice CYPHER	35
4.3.1	Convenzioni di denominazione	35
4.3.2	Struttura.....	36
4.3.3	Query.....	38
5	Conclusioni	40
6	Bibliografia	42

Elenco delle figure

Figure 1: Esempio di un modello a grafo	10
Figure 2: SQL vs Cypher.....	13
Figure 3: Esempio di un grafo in Neo4j.....	14
Figure 4: Descriptor “Alzheimer Disease” nel browser MeSH.....	17
Figure 5: Schema concettuale.....	28
Figure 6: Schema concettuale ristrutturato.....	28
Figure 7: Schema logico	33

Capitolo 1

1 Introduzione

Questo progetto di tesi consiste nel modellare la realtà dei dati dell'organizzazione medica MeSH¹ all'interno del DBMS (Database Management System) Neo4j² che permette la manipolazione di database a grafo.

La progettazione e la realizzazione si può suddividere in tre fasi principali:

- Studio e analisi dei **database a grafo** e del relativo ambiente di gestione **Neo4j**;
- Analisi dei dati medici **MeSH** e comprensione della loro struttura;
- Progettazione della **base di dati** che rappresenta al meglio questa realtà.

La prima parte consiste in uno studio individuale volto a comprendere e saper creare database a grafo. In questo Neo4j si pone come un ottimo strumento sia di gestione del database sia di apprendimento perché mette a disposizione video e manuali con lo scopo sia di fornire informazioni valide per qualsiasi database a grafo e sia nello specifico per la sua realizzazione nello stesso ambiente di sviluppo. Di supporto a Neo4j si trovano molte piattaforme capaci di tradurre un grafo disegnato dall'utente in codice di creazione per un database a grafo, ad esempio il sito web <http://www.apciones.com/arrows/>.

Presa confidenza con questi tools, la seconda parte prevede l'analisi dei dati medici MeSH. Medical Subject Headings (MeSH) è una terminologia organizzata gerarchicamente per l'indicizzazione e la catalogazione di informazioni biomediche. All'interno dei dati MeSH è possibile cercare un termine medico e avere come output i possibili sinonimi, le malattie derivate e anche i *Concept*³ che lo rappresentano biologicamente.

Infine, l'ultima parte consiste nel progettare un database a grafo Neo4j con la struttura dei dati MeSH appena analizzati. Questa fase si può dividere in altre tre sotto fasi: progettazione concettuale, logica e fisica. La progettazione concettuale si occupa di schematizzare la realtà di interesse attraverso un modello chiamato *labeled property graph model*⁴ che verrà trattato nel Capitolo 2.1. La progettazione logica, invece, consiste nella traduzione dello schema concettuale in un determinato modello logico di dati usato dal DBMS che si intende utilizzare. Infine, la progettazione fisica consiste nell'implementazione fisica attraverso il linguaggio Cypher, proprio del DBMS Neo4j.

¹ Per ulteriori approfondimenti, visitare il sito: <https://www.nlm.nih.gov/databases/download/mesh.html>

² Per ulteriori approfondimenti, visitare il sito: <https://neo4j.com/>

³ Molecola chimica farmacologica

⁴ Modello a grafo con proprietà ed etichette

Capitolo 2

2 Database a grafo e Neo4j

In questo capitolo verrà approfondito il concetto di *database a grafo* e citato il sistema di gestione Neo4j che permette la creazione e la modifica di questo tipo di database. Neo4j non è l'unico ambiente di gestione per database a grafo, esistono anche OrientDB, Oracle Spatial and Graph, ArangoDB e molti altri. Si è scelto di utilizzare Neo4j per il suo ambiente Open Source accessibile sia mediante download sia mediante registrazione online, e per la sua partnership con grandi aziende quali IBM e Amazon Web Services.

Nel capitolo saranno trattati:

- I concetti chiave per una struttura del tipo **database a grafo**;
- Le differenze tra due basi di dati: **database a grafo e database relazionale**;
- **Neo4j** dove verrà illustrato l'ambiente di sviluppo Neo4j e il relativo linguaggio *Cypher* usato per la scrittura, lettura e modifica fisica dei dati.

2.1 Database a grafo

Formalmente, un grafo è una struttura composta da punti (*vertici* o *nodi del grafo*) e da linee (*lati* o *spigoli del grafo*). Le linee mettono in relazioni i punti.

Un database a grafo sfrutta la struttura del grafo per assimilare ai punti i *nodi* e alle linee le *relazioni*.

Per la rappresentazione utilizziamo la forma più popolare di modello a grafo: il *labeled property graph model* di cui riportiamo un esempio nella Figura sottostante.

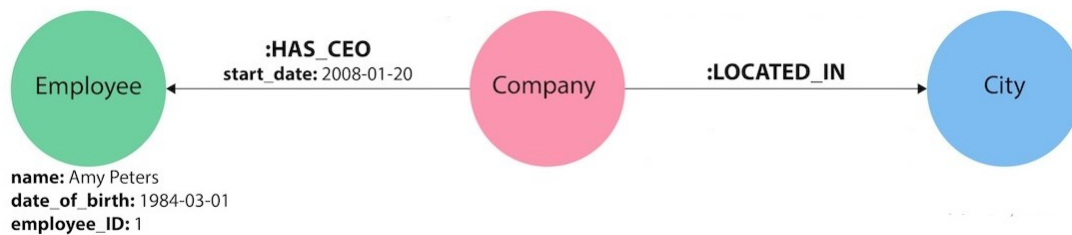


Figure 1: Esempio di un modello a grafo⁵

Un *labeled property graph model*⁶ si contraddistingue per rappresentare i dati mediante nodi e relazioni, con le relative proprietà ed etichette.

I **nodi** rappresentano gli oggetti e nel grafo sono i suoi punti (ad esempio: Employee, Company e City). Possono avere una o più attributi (coppie chiave-valore), chiamati **proprietà**. Un esempio di proprietà è il *name* nell'*Employee*. Inoltre, i nodi possono essere contrassegnati con delle **etichette** che rappresentano il loro ruolo nel contesto. Le etichette possono servire anche per agganciare metadata a gruppi di nodi.

Le **relazioni** invece sono le azioni compiute dagli oggetti e sono rappresentate mediante linee direzionali. Costituiscono una connessione tra due nodi (ad esempio: :LOCATED_IN tra Company e City) e hanno un nome, una direzione, un tipo, un nodo di partenza e un nodo di arrivo. Come i nodi, possono avere delle proprietà, si veda nella Figura, ad esempio, *start_date* in *HAS_CEO*. Solitamente hanno proprietà quantitative, come peso, costo, distanza, intervallo temporale e lunghezza.

La maggior parte delle persone trova questo modello intuitivo e di facile comprensione. Sebbene semplice, può essere usato per descrivere la stragrande maggioranza dei progetti che prevedono l'uso di un database a grafo. Alcuni utilizzi di questo tipo di database sono:

- I social network;
- I software per la gestione di grandi qualità di dati;
- Le ricerche basate su strutture a grafo;
- La gestione degli accessi con identificazione.

⁵ Tratto dal sito: <https://Neo4j.com/developer/graph-database/?ref=web-product-database/>


⁶ Tratto dal libro: Ian Robinson, Ian Robinson, Jim Webber e Emil Eifrem (2015) *Graph Databases*

2.2 Database a grafo e database relazionale

Se un database a grafo rappresenta i dati mediante l'uso di un grafo, un database relazionale utilizza invece delle tabelle. Infatti, i nodi del database a grafo sono nel database relazionale chiamate *entità* e i loro valori salvati in tabelle le cui colonne sono le proprietà della stessa *entità*. Le proprietà vengono chiamate *attributi*. Le relazioni vengono definite, invece, attraverso il collegamento di due o più colonne tra le tabelle.

Ad esempio, se si volesse rappresentare l'*entità* Persona con attributi (Nome, Cognome, CodiceFiscale) e l'*entità* Utente con attributi (CodiceFiscale, Username, Password) avrebbe questo aspetto:

Persona			Utente		
Nome	Cognome	CodiceFiscale	CodiceFiscale	Username	Password
Mario	Rossi	RSSMRA70A41F205Z	RSSMRA70A41F205Z	mario_ros	Secret!



In questo caso dovrebbe esserci continua coerenza tra il codice fiscale dell'Utente e della Persona. Le coerenze nel database relazionale vengono gestite mediante vincoli interni alla tabella, come i *vincoli di chiave primaria* che identificano univocamente ogni istanza di una entità, e vincoli tra le tabelle, come i *vincoli di chiave esterna* di cui vediamo un esempio nel codice fiscale della precedente rappresentazione.

La struttura del database relazionale deve essere, quindi, definita a priori. Questo comporta un vantaggio in termini di consistenza dei dati ma uno svantaggio in caso di aggiornamenti o cambi nella struttura futura. Se si decidesse di cambiare sostanziosamente la struttura dei dati, sarebbe necessario rivisitare tutte le entità con le rispettive relazioni.

Con i database a grafo questo invece è ottimizzato. La struttura non viene definita a priori e quindi ha prestazioni migliori in caso di aggiornamenti e non solo. Un altro aspetto, infatti, che differenzia i due tipi di database è il linguaggio con cui vengono scritte le interrogazioni sui dati. Nel database relazionale viene usato il linguaggio SQL invece in un database a grafo progettato su Neo4j viene utilizzato Cypher, che verrà approfondito nella prossima sezione.

2.3 Neo4j

Neo4j è indicato come uno dei principali gestori di database a grafo perché implementa in modo efficiente le proprietà dei modelli a grafo fino al livello di archiviazione. Ciò significa che i dati vengono archiviati esattamente come se venissero rappresentati su una lavagna e vengono letti attraverso dei puntatori.

Neo4j è stato sviluppato nel 2003 ma è stato reso disponibile al pubblico nel 2007. Il codice sorgente, scritto in Java e Scala, è disponibile gratuitamente su GitHub. Il software può essere scaricato mediante download oppure utilizzato tramite browser con Neo4j Sandbox⁷.

Neo4j è molto popolare tra gli sviluppatori per:

- **Cypher**, un linguaggio di query dichiarativo simile a SQL, ma ottimizzato per i grafi. Ora utilizzato da altri database come SAP HANA Graph e Redis graph tramite il progetto openCypher.
- **Attraversamento a tempo costante** in grafi con elevata profondità grazie all'efficace rappresentazione di nodi e relazioni. Consente di scalare fino a miliardi di nodi su hardware moderati.
- **Flessibilità** nella modifica dello schema di nodi e relazioni che permette un agevole manipolazione dei dati in caso di errori, aggiornamenti o implementazioni future.
- **Driver per i linguaggi di programmazione** più diffusi, tra cui Java, JavaScript, .NET, Python e molti altri.

⁷ Per ulteriori approfondimenti, visitare il sito: <https://sandbox.Neo4j.com/>


```
MATCH (b:`Movie` { `title`: "The Matrix" })
```

```
CREATE (a)-[:`ACTED_IN`]->(b)
```

oppure

```
CREATE (a:`Person` { `name`: "Keanu Reeves" })-[:`ACTED_IN`]->(b:`Movie` { `title`: "The Matrix" })
```

E la sua rappresentazione in Neo4j sarebbe:

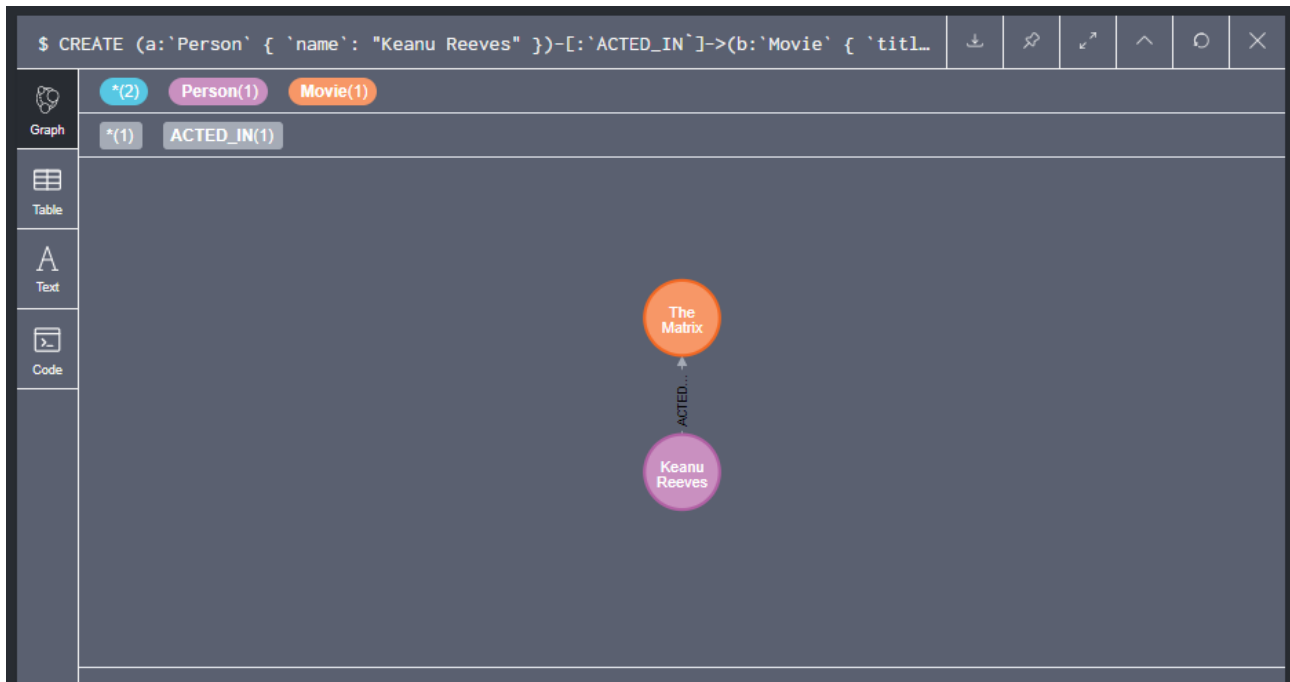


Figure 3: Esempio di un grafo in Neo4j

Capitolo 3

3 Analisi dei dati MeSH

In questo capitolo verrà spiegata, previa analisi, la struttura dei dati Medical Subject Headings (MeSH) aggiornati al giorno 01/01/2020. MeSH è una terminologia organizzata gerarchicamente per l'indicizzazione e la catalogazione di informazioni biomediche scritta e gestita dal National Library of Medicine (NLM) degli Stati Uniti.

Nel capitolo sarà enunciata:

- La struttura dei **dati MeSH**;
- I **requisiti strutturali** che andranno a descrivere, in un linguaggio informale, tutte le entità che faranno parte del database con le relative proprietà;
- Un **glossario** che va a riassumere le principali entità con una breve descrizione, i sinonimi con cui ci si può riferire ad esse e i collegamenti diretti con le altre.

3.1 Dati MeSH

NLM mette a disposizione i dati MeSH in diverse modalità: permette di eseguire una ricerca localizzata o estesa a tutto il sistema¹⁰, permette di scaricare l'intero database di dati in diversi formati (XML, ASCII, MARC 21 o RDF)¹¹ e anche di scaricare la struttura di alcune tipologie di dati (*Descriptor*, *Qualifier* e *Supplemental*) proprie del database stesso.

Nella sezione di ricerca vengono mostrati, per ogni termine cercato, i dettagli, i sinonimi, la posizione all'interno del database, relativa ad una gerarchia di indici, i *Qualifiers* (ad esempio: il sangue) che sono a lui collegati e altri campi opzionali.

Provando a ricercare il termine "Alzheimer Disease" la pagina si presenta come segue:

The screenshot displays the MeSH browser interface for the term 'Alzheimer Disease'. The title is 'Alzheimer Disease MeSH Descriptor Data 2020'. Below the title are four tabs: 'Details' (selected), 'Qualifiers', 'MeSH Tree Structures', and 'Concepts'. The 'Details' tab shows the following information:

- MeSH Heading:** Alzheimer Disease
- Tree Number(s):** C10.228.140.380.100, C10.574.945.249, F03.615.400.100
- Unique ID:** D000544
- RDF Unique Identifier:** <http://id.nlm.nih.gov/mesh/D000544>
- Scope Note:** A degenerative disease of the BRAIN characterized by the insidious onset of DEMENTIA. Impairment of MEMORY, judgment, attention span, and problem solving skills are followed by severe APRAXIAS and a global loss of cognitive abilities. The condition primarily occurs after age 60, and is marked pathologically by severe cortical atrophy and the triad of SENILE PLAQUES; NEUROFIBRILLARY TANGLES; and NEUROFIL THREADS. (From Adams et al., Principles of Neurology, 6th ed, pp1049-57)
- Entry Version:** ALZHEIMER DIS
- Entry Term(s):** Acute Confusional Senile Dementia, Alzheimer Dementia, Alzheimer Disease, Early Onset, Alzheimer Disease, Late Onset, Alzheimer Sclerosis, Alzheimer Syndrome, Alzheimer Type Senile Dementia, Alzheimer's Disease, Alzheimer's Disease, Focal Onset, Alzheimer-Type Dementia (ATD), Dementia, Alzheimer Type, Dementia, Presenile, Dementia, Primary Senile Degenerative, Dementia, Senile, Early Onset Alzheimer Disease, Familial Alzheimer Disease (FAD), Focal Onset Alzheimer's Disease, Late Onset Alzheimer Disease, Presenile Alzheimer Dementia, Primary Senile Degenerative Dementia, Senile Dementia, Acute Confusional, Senile Dementia, Alzheimer Type
- See Also:** Amyloid beta-Peptides, Amyloid beta-Protein Precursor, Aphasia, Primary Progressive, Cerebral Amyloid Angiopathy, Kluver-Bucy Syndrome, Neurofilament Proteins, tau Proteins
- Publ MeSH Note:** 1998: see ALZHEIMER'S DISEASE 1984-1997, see DEMENTIA, PRESENILE 1967-1983, see PSYCHOSES, PRESENILE 1963-1966; for DEMENTIA, PRESENILE and DEMENTIA, PRIMARY DEGENERATIVE, SENILE see DEMENTIA 1981-1999; for DEMENTIA, SENILE see DEMENTIA 1966-1999
- History Note:** 1998(1963); for DEMENTIA, PRESENILE and DEMENTIA, PRIMARY DEGENERATIVE SENILE use DEMENTIA 1981-1999; for DEMENTIA SENILE use DEMENTIA 1966-1999
- Date Established:** 1984/01/01
- Date of Entry:** 1999/01/01
- Revision Date:** 2017/02/24

Figure 4: Descriptor "Alzheimer Disease" nel browser MeSH

In base alle tendine (Details, Qualifiers, MeSH Tree Structures, Concepts) possiamo scorrere tutte le informazioni e osservare nella sezione "Details" il codice UI univoco che identifica la malattia all'interno del database MeSH, i suoi *Entry Terms* (sinonimi) e anche le relative pubblicazioni al

¹⁰ Per ulteriori approfondimenti, visitare il sito: <https://meshb.nlm.nih.gov/search>

¹¹ Per ulteriori approfondimenti, visitare il sito: <https://www.nlm.nih.gov/databases/download/mesh.html>

riguardo. I sinonimi si rendono utili, ad esempio, per cercare tutte le cartelle cliniche inerenti alla malattia nei vari nominativi.

Nella tendina *MeSH Tree Structures* la malattia “Alzheimer Disease” compare sotto la categoria delle taupatie e delle demenze sia legate al cervello sia ai disturbi neurocognitivi. La *Tree Structure* quindi ci permette di inquadrare gerarchicamente la malattia rispetto alle altre.

Come *Qualifiers* invece ritroviamo il sangue, il metabolismo e la virologia, insieme a molti altri. I *Qualifiers* quindi indicano gli ambiti biochimici della malattia.

Oltre alla sezione di ricerca, NLM rende accessibili e scaricabile la struttura delle entità *Qualifier*, *Supplemental* e *Descriptor* utili a capire la suddivisione e l’organizzazione dei dati tra di loro.

Ad esempio, la struttura del *Supplemental* datata 01/01/2020 si mostra come segue:

```
<!-- Author: MeSH -->
<!-- Effective: 01/01/2020 -->

<!-- ===== -->
<!--      Revision Notes Section
-->
<!-- ===== -->

<!ENTITY % DescriptorRecordSet SYSTEM "nlmdescriptorrecordset_20200101.dtd">
%DescriptorRecordSet;

<!ENTITY % ChemicalReference "(SupplementalRecordUI, SupplementalRecordName)">

<!ELEMENT SupplementalRecordSet (SupplementalRecord*)>
<!ATTLIST                      SupplementalRecordSet                      LanguageCode
(cze|dut|eng|fin|fre|ger|ita|jpn|lav|por|scr|slv|spa) #REQUIRED>
<!ELEMENT SupplementalRecord (%ChemicalReference;,
    DateCreated,
    DateRevised?,
    Note?,
    Frequency?,
    PreviousIndexingList?,
    HeadingMappedToList+,
    IndexingInformationList*,
    PharmacologicalActionList?,
    SourceList*,
    ConceptList) >
<!ATTLIST SupplementalRecord SCRCClass (1 | 2 | 3 | 4) "1">

<!ELEMENT Note (#PCDATA)>
<!ELEMENT SourceList (Source+)>
<!ELEMENT Source (#PCDATA)>
```

```

<!ELEMENT Frequency (#PCDATA)>
<!ELEMENT HeadingMappedToList (HeadingMappedTo+)>
<!ELEMENT HeadingMappedTo (DescriptorReferredTo,QualifierReferredTo?) >
<!ELEMENT IndexingInformationList (IndexingInformation+)>
<!ELEMENT IndexingInformation (DescriptorReferredTo,QualifierReferredTo?)>
<!ELEMENT SupplementalRecordUI (#PCDATA) >
<!ELEMENT SupplementalRecordName (String) >

```

I simboli usati hanno i seguenti significati:

- “#PCDATA” ovvero “parseable character data” in riferimento ad una proprietà indica che tale proprietà è in formato testuale, come ad esempio *Note*;
- “%” anticipa la sintassi dell’identificatore univoco di ciascun *SupplementalRecord* ovvero “(SupplementalRecordUI, SupplementalRecordName)” dove *SupplementalRecordUI* è di tipo #PCDATA e *SupplementalRecordName* di tipo String;
- “?” segue una proprietà che può essere presente al più una volta, ad esempio “DateRevised?” indica che il record può avere una o nessuna data di revisione;
- “*” segue una lista di entità che possono essere un numero da zero in su, ad esempio “SourceList*” indica che il record può essere collegato a zero o più *Source*;
- “+” segue una lista che deve avere una o più entità collegate al record in esame, ad esempio “HeadingMappedToList+” indica che il record deve essere collegato a uno o più entità *HeadingMappedTo*;
- “[]” tra due proprietà significa che ne deve essere scelta una delle due ma non entrambe.

Le date sono in formato *normal.date*, definito dal NLM con la dicitura “(Year, Month, Day)”.

Dall’analisi delle strutture fornite, si evince anche la struttura di altre entità, quali *Pharmalogical Action*, *Concept*, *Term*, *Substance* e *HeadingMappedTo* che abbiamo visto essere obbligatorio per il *Supplemental*.

Pharmacological Action viene definito come:

```

<!ELEMENT PharmacologicalAction (% DescriptorReferredTo,
PharmacologicalActionSubstanceList? >

<!ENTITY DescriptorReferredTo "(DescriptorUI, DescriptorName)">

<!ELEMENT PharmacologicalActionSubstanceList (Substance+)>

```

Substance viene definito come:

```

<!ELEMENT Substance (%SubstanceReference)>

<!ENTITY % SubstanceReference "(RecordUI, RecordName)">

```

Concept viene definito come:

```

<!ELEMENT Concept (%ConceptReference;,
    CASN1Name?,
    RegistryNumber?,
    ScopeNote?,
    TranslatorsEnglishScopeNote?,
    TranslatorsScopeNote?,
    RelatedRegistryNumberList?,
    ConceptRelationList?,
    TermList)>
<!ENTITY % ConceptReference "(ConceptUI,ConceptName)">

```

Term viene definito come:

```

<!ELEMENT Term (%TermReference;,
    DateCreated?,
    Abbreviation?,
    SortVersion?,
    EntryVersion?,
    ThesaurusIDlist?,
    TermNote?)>
<!ENTITY % TermReference "(TermUI, String)">

```

Infine, *HeadingMappedTo* è definito con un *DescriptorReferredTo*, ovvero un collegamento ad un *Descriptor*, e un *QualifierReferredTo*, ovvero un collegamento ad un *Qualifier*, opzionale.

Alle entità principali possono essere collegate una o più entità formate da un solo valore testuale di tipo “#PCDATA”, quali: *RelatedRegistryNumber*, *ThesaurusID*, *PreviousIndexing*, *TreeNumber* e *Source*.

E altre entità che hanno collegamenti opzionali con le entità principali, quali: *EntryCombination*, *AllowableQualifier* e *IndexingInformation*.

IndexingInformation ha la stessa struttura di *HeadingMappedTo*.

AllowableQualifier ha un *QualifierReferredTo*, ovvero un collegamento ad un *Qualifier Record*, e una proprietà “Abbreviation”.

EntryCombination, invece, è formata da una proprietà “ECIN” e una “ECOUT” entrambe con riferimenti a un *Descriptor* e un *Qualifier*.

Saranno spiegate e illustrate le relazioni tra le entità nel Capitolo 4.

3.2 Requisiti strutturali

Frase per Term

Term è un termine usato in un Concept.

È identificato da un UI testuale e da un nome in formato alfanumerico. Può avere in aggiunta la data di creazione, la versione di ordinamento e di iscrizione al sito, l'abbreviazione, delle note e una lista di riferimenti a dizionari con relativo anno.

Frase per Concept

Concept indica una molecola chimica farmacologica.

È identificato da un UI testuale e da un nome alfanumerico. È collegato a uno o più Term e può essere collegato ad una lista di numeri di registri e altri Concept. Può avere altri valori opzionali, quali: CASName (composizione chimica della molecola), numero di registro, uso e relativa traduzione inglese e in altre lingue.

Frase per Descriptor

Un Descriptor è un termine della rubrica principale MeSH.

È identificato da un UI testuale e un nome alfanumerico. Ha una data di creazione e può avere una data di revisione e di diffusione. Può avere dei riferimenti storici o online, anche all'interno di MeSH, e un numero di classificazione definito da NLM. È collegato a uno o più Concept ma può avere collegamenti anche a dei precedenti nominativi, dei sinonimi o altri Descriptor che lo precedono o lo seguono nella gerarchia ad albero di MeSH o che possono essere collegati a lui in qualche modo. Infine, può avere uno o più Pharmacological Action.

Frase per Qualifier

Qualifier indica l'ambito medico a cui appartiene un Descriptor.

È identificato da un UI testuale e un nome alfanumerico. Ha una data di creazione e può avere una data di revisione e di diffusione. Può avere delle note storiche o online. Può essere collegato ad altri Qualifier che lo precedono o lo seguono gerarchicamente. È, infine, sicuramente collegato a uno o più Concept.

Frase per Supplemental

Supplemental è un'entità di supporto.

È identificato da un UI testuale e un nome alfanumerico. Ha una data di creazione e può avere una data di revisione. Può avere delle note e una frequenza. Ha sicuramente uno o più collegamenti a

HeadingMappedTo e Concept. Può avere zero o più collegamenti con PreviousIndexing, IndexingInformation, PharmacologicalAction e Source.

Frase per Pharmacological Action

Una Pharmacological Action indica un'azione farmacologica ovvero un'alterazione provocata dalla somministrazione di una o più sostanze. È identificata da un Descriptor e può essere collegata ad una o più Substance.

Frase per Substance

Una Substance è una sostanza coinvolta in un'azione farmacologica.

È identificato da un UI testuale e un nome alfanumerico.

Frase per HeadingMappedTo

Partendo da un Supplemental, HeadingMappedTo permette di risalire a quale Descriptor, ed eventualmente Qualifier, Supplemental fa da supporto.

È identificato da un Descriptor e può avere un riferimento ad un Qualifier.

Frase per RelatedRegistryNumber

RelatedRegistryNumber indica il nome di registro legato ad un Concept. Ha solo un UI testuale.

Frase per ThesaurusID

ThesaurusID indica il dizionario con anno al quale un Term appartiene. Ha solo un UI testuale.

Frase per PreviousIndexing

PreviousIndexing indica un precedente nome di un Descriptor o di un Supplemental. Ha solo un UI testuale.

Frase per TreeNumber

TreeNumber è il numero o percorso numerico che posiziona un Descriptor o un Qualifier all'interno del database MeSH. Ha solo un UI testuale.

Frase per Source

Source è la sorgente di un Supplemental. Ha solo un UI testuale.

Frase per EntryCombination

EntryCombination indica un sinonimo di un certo Descriptor. È formata da un Entry in entrata (ECIN) e una Entry in uscita (ECOUT). ECIN e ECOUT sono entrambe collegate ad un Descriptor ma ECIN è sicuramente collegata ad un Qualifier invece ECOUT lo è opzionalmente.

Frase per AllowableQualifier

AllowableQualifier specifica l'abbreviazione di un certo Qualifier in merito ad un Descriptor. È sicuramente collegato ad un Qualifier e ha una abbreviazione.

Frase per IndexingInformation

IndexingInformation inquadra genericamente un Supplemental in riferimento ad un Descriptor ed eventualmente anche un Qualifier. Ha la stessa struttura di HeadingMappedTo.

3.3 Glossario

TERMINE	DESCRIZIONE	SINONIMI	COLLEGAMENTI
ThesaurusID	Indica a quale dizionario, con relativo anno, si riferisce un certo Term.		Term
Term	Termine usato in un Concept.		ThesaurusID, Concept
Concept	Molecola chimica farmacologica.		Term, RelatedRegistryNumber, Supplemental, Qualifier, Descriptor, Concept
Descriptor	Termine principale della rubrica MeSH.		Concept, PharmacologicalAction, PreviousIndexing, EntryCombination, Descriptor, TreeNumber, AllowableQualifier, HeadingMappedTo, IndexingInformation
Qualifier	Ambito medico a cui appartiene un Descriptor.		Concept, IndexingInformation, HeadingMappedTo, AllowableQualifier, TreeNumber
Supplemental	Entità di supporto.	Chemical	PreviousIndexing, HeadingMappedTo, IndexingInformation, Source, PharmacologicalAction, Concept
Pharmacological Action	Azione farmacologica che coinvolge più sostanze.		Substance, Descriptor, Supplemental
Substance	Sostanza coinvolta in un'azione farmacologica.		PharmacologicalAction
Heading Mapped To	Permette di risalire a quale Descriptor ed eventualmente Qualifier è stato mappato un determinato Supplemental.		Supplemental, Descriptor, Qualifier
Previous Indexing	Nominativo storico precedente all'attuale.		Descriptor, Supplemental
Indexing Information	Descriptor ed eventualmente Qualifier a cui un Supplemental è collegato.		Supplemental, Descriptor, Qualifier

Related Registry Number	Numero di registro a cui è collegato un Concept.		Concept
Tree Number	Posizione di un Descriptor o di un Qualifier nell'albero generato da NLM.		Descriptor, Qualifier
Allowable Qualifier	Abbreviazione di un certo Qualifier in un dato Descriptor.		Descriptor, Qualifier
Entry Combination	Sinonimo di un certo Descriptor.		ECIN, ECOUT
ECIN	Nome di partenza di un Descriptor.		Descriptor, Qualifier
ECOUT	Nome di arrivo di un Descriptor.		Descriptor, Qualifier
Source	Sorgente di un certo Supplemental.		Supplemental

Capitolo 4

4 Progettazione e realizzazione del Database

4.1 Progettazione Concettuale

La progettazione o modellazione Concettuale è la prima fase di schematizzazione e formalizzazione di una realtà di dati e delle relazioni tra di essi.

Ha le seguenti caratteristiche:

- È **indipendente** rispetto al DBMS che verrà utilizzato;
- È in grado di rappresentare i dati in un modello formale, ad **alto livello**.

È per definizione indipendente dalla concorrenza e dalla memorizzazione dei dati gestita dal DBMS e dal sistema utilizzato.

Nel capitolo verrà descritto:

- Il **modello concettuale** rappresentativo dei dati MeSH ed espresso attraverso il modello *labeled property graph* spiegato al Capitolo 2;
- Un'eventuale **ristrutturazione** dello schema concettuale;
- Un **dizionario dei dati** dove vengono chiarite le entità e le associazioni tra di esse, con i relativi attributi e identificatori.

4.1.1 Modello Concettuale

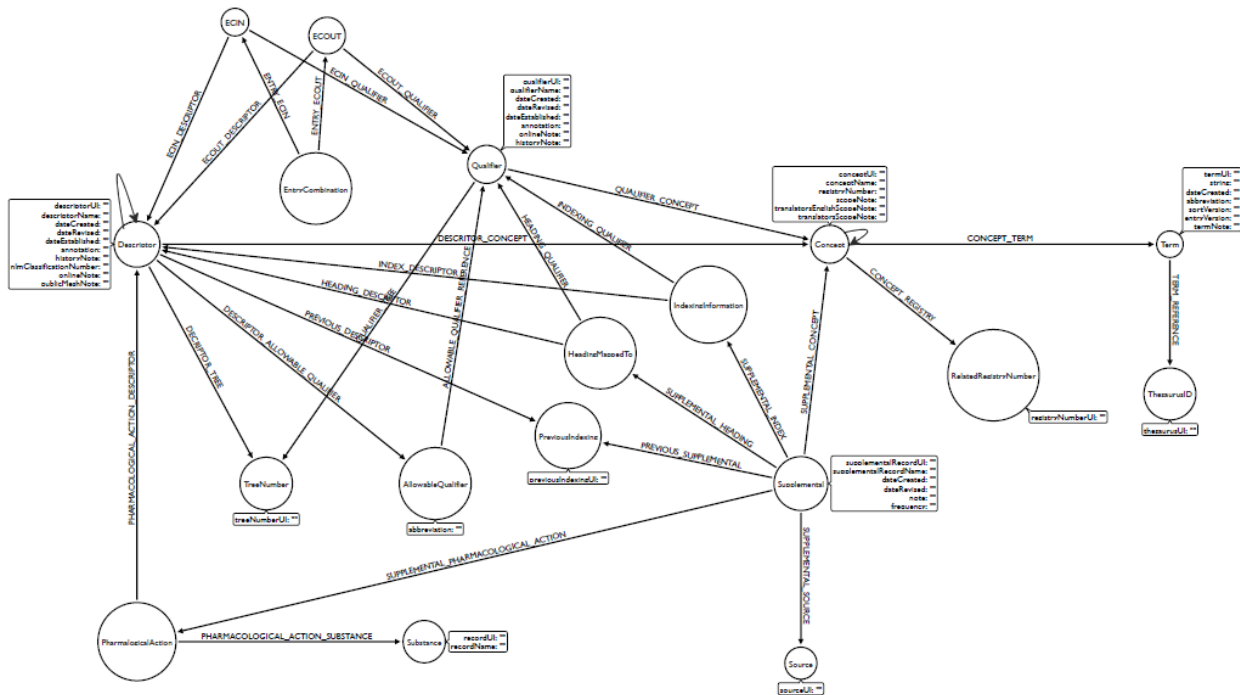


Figure 5: Schema concettuale

4.1.2 Ristrutturazione schema

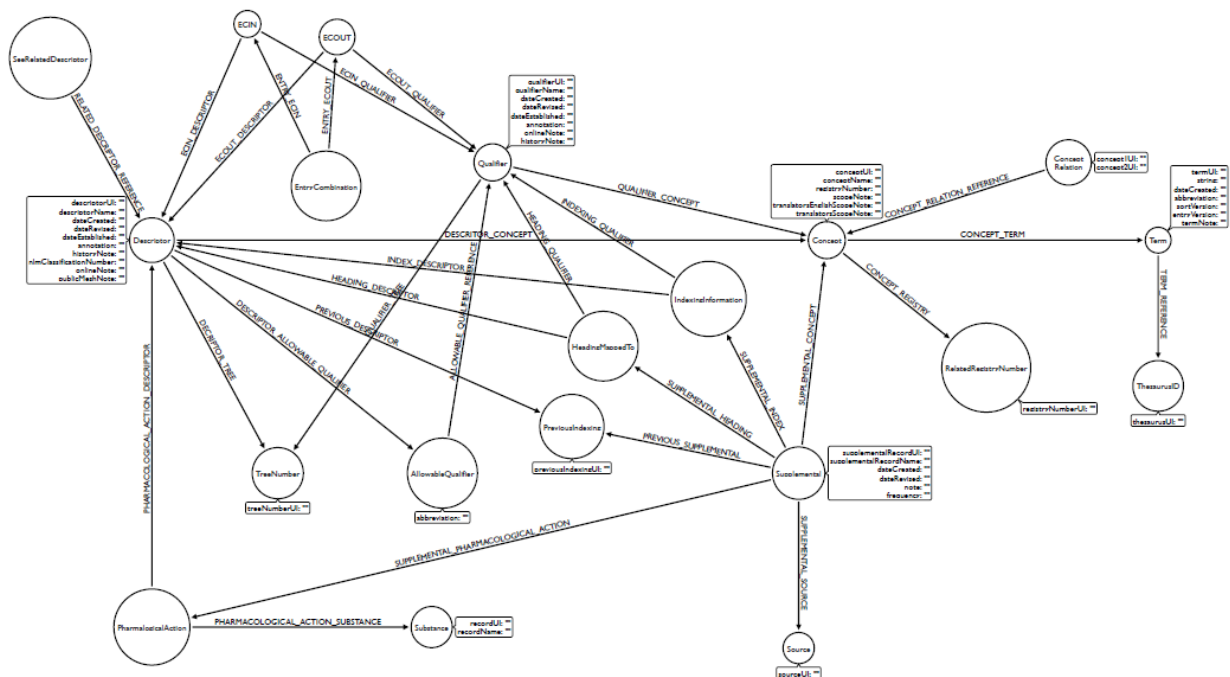


Figure 6: Schema concettuale ristrutturato

4.1.3 Dizionario dei dati

Entità

TERMINE	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORI
ThesaurusID	Indica a quale dizionario, con relativo anno, si riferisce un certo Term.	thesaurusUI	thesaurusUI
Term	Termine usato in un Concept.	termUI, string, dateCreated, abbreviation, sortVersion, entryVersion, termNote	termUI, string
Concept	Molecola chimica farmacologica.	conceptUI, conceptName, casn1Name, registryNumber, scopeNote, translatorsEnglishScopeNote, translatorsScopeNote	conceptUI, conceptName
Descriptor	Termine principale della rubrica MeSH.	descriptorUI, descriptorName, dateCreated, dateRevised, dateEstablished, annotation, historyNote, nlmClassificationNumber, onlineNote, publicMeSHNote, considerAlso	descriptorUI, descriptorName
Qualifier	Ambito medico a cui appartiene un Descriptor.	qualifierUI, qualifierName, dateCreated, dateRevised, dateEstablished, annotation, historyNote, onlineNote	qualifierUI, qualifierName
Supplemental	Entità di supporto.	supplementalRecordUI, supplementalRecordName, dateCreated, dateRevised,	supplementalRecordUI, supplementalRecordName

		note, frequency	
Pharmacological Action	Azione farmacologica che coinvolge più sostanze.	descriptorUI, descriptorName	descriptorUI, descriptorName
Substance	Sostanza coinvolta in un'azione farmacologica.	recordUI, recordName	recordUI, recordName
Heading Mapped To	Permette di risalire a quale Descriptor ed eventualmente Qualifier è stato mappato un determinato Supplemental.	descriptorUI, descriptorName, qualifierUI, qualifierName	descriptorUI, descriptorName
Previous Indexing	Nominativo storico precedente all'attuale.	previousIndexingUI	previousIndexingUI
Indexing Information	Descriptor ed eventualmente Qualifier a cui un Supplemental è collegato.	descriptorUI, descriptorName, qualifierUI, qualifierName	descriptorUI, descriptorName
Related Registry Number	Numero di registro a cui è collegato un Concept.	relatedRegistryNumberUI	relatedRegistryNumberUI
Tree Number	Posizione di un Descriptor o di un Qualifier nell'albero generato da NLM.	treeNumberUI	treeNumberUI
Allowable Qualifier	Abbreviazione di un certo Qualifier in un dato Descriptor.	qualifierUI, qualifierName, abbreviation	qualifierUI, qualifierName
Entry Combination	Sinonimo di un certo Descriptor.	ecinDescriptorUI, ecinDescriptorName, ecinQualifierUI, ecinQualifierName ecoutDescriptorUI, ecoutDescriptorName	ecinDescriptorUI, ecinDescriptorName, ecinQualifierUI, ecinQualifierName ecoutDescriptorUI, ecoutDescriptorName
Ecin	Nome di partenza di un Descriptor.	descriptorUI, descriptorName, qualifierUI, qualifierName	descriptorUI, descriptorName, qualifierUI, qualifierName
Ecout	Nome di arrivo di un Descriptor.	descriptorUI, descriptorName, qualifierUI, qualifierName	descriptorUI, descriptorName

Source	Sorgente di un certo Supplemental.	sourceUI	sourceUI
Concept Relation	Indica una relazione tra due Concept.	concept1UI, concept2UI	concept1UI, concept2UI
See Related Descriptor	Indica la relazione tra due Descriptor.	descriptor1UI, descriptor1Name, descriptor2UI, descriptor2Name	descriptor1UI, descriptor1Name, descriptor2UI, descriptor2Name,

Associazioni

Associazione	Attributi	Entità collegate
TERM_REFERENCE		Term (0, N), ThesaurusID (0, N)
CONCEPT_TERM		Concept (1, N), Term (0, N)
CONCEPT_REGISTRY		Concept (0, N), RelatedRegistryNumber (0, N)
SUPPLEMENTAL_CONCEPT		Supplemental (1, N), Concept (0, N)
SUPPLEMENTAL_SOURCE		Supplemental (0, N), Source (0, N)
QUALIFIER_CONCEPT		Qualifier (1, N), Concept (0, N)
DESCRIPTOR_CONCEPT		Descriptor (1, N), Concept (0, N)
SUPPLEMENTAL_INDEX		Supplemental (0, N), IndexingInformation (0, N)
SUPPLEMENTAL_HEADING		Supplemental (1, N), HeadingMappedTo (0, N)
PREVIOUS_SUPPLEMENTAL		PreviousIndexing (0, N), Supplemental (0, N)
PREVIOUS_DESCRIPTOR		PreviousIndexing (0, N), Descriptor (0, N)
DESCRIPTOR_ALLOWABLE_QUALIFIER		Descriptor (0, N), AllowableQualifier (0, N)
DESCRIPTOR_TREE		Descriptor (0, N), TreeNumber (0, N)
QUALIFIER_TREE		Qualifier (0, N), TreeNumber (0, N)
SUPPLEMENTAL_PHARMACOLOGICAL_ACTION		Supplemental (0, N), PharmacologicalAction (0, N)
PHARMACOLOGICAL_ACTION_SUBSTANCE		PharmacologicalAction (0, N), Substance (0, N)

4.2 Progettazione Logica

La progettazione o modellazione logica è la fase che segue la progettazione concettuale e che permette di tradurre lo schema concettuale in un linguaggio sempre più vicino al DBMS utilizzato.

Nei Database a grafo le relazioni non sono sottoposte a vincoli di chiave esterna come nei Database relazionali bensì sono composte dagli UI dei nodi che vi partecipano. Gli UI dei nodi sono UI univoci all'interno dell'intero Database, assegnati ad ogni nodo indipendentemente dall'etichetta dello stesso.

Per dare una visione più completa della realtà si sceglie, dunque, di mostrare uno schema logico di tipo relazionale. Questo tipo di schema mostra quali sono le proprietà di ogni nodo che interessano la relazione facendo attenzione alla cardinalità.

Nel capitolo sarà, quindi, riportato:

- Lo **schema logico** prodotto per la realtà di interesse;
- Un insieme di **regole di vincolo** per lo schema logico altrimenti non esprimibili solo con lo stesso.

4.2.1 Modello Logico: Relazionale

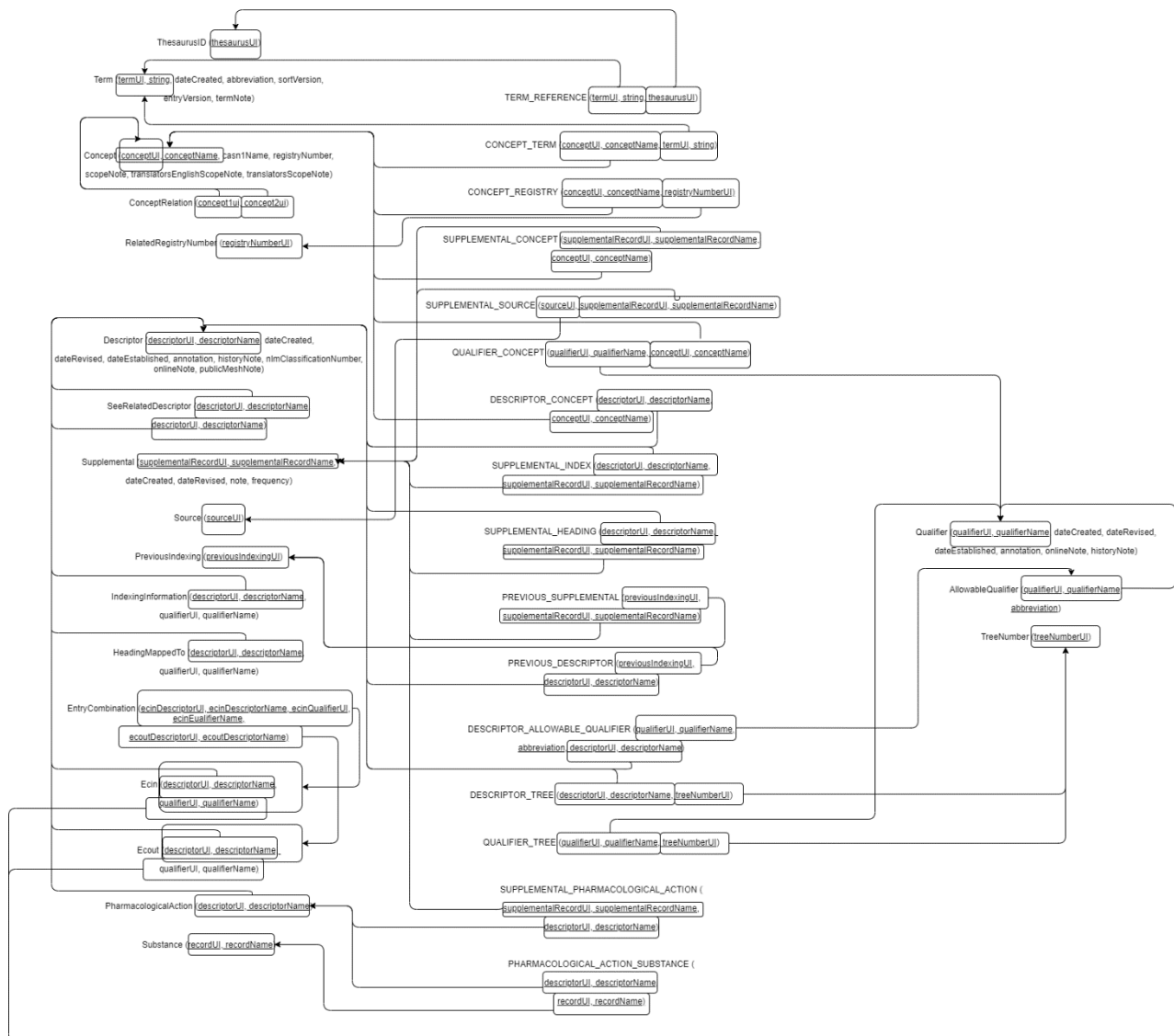


Figure 7: Schema logico

4.2.2 Regole di Vincolo

Le regole di vincolo dello schema logico evidenziano le cardinalità obbligatorie nelle relazioni presenti nel modello logico.

- **RV1:** Un Descriptor deve essere collegato ad almeno un Concept.
- **RV2:** Un Concept deve essere collegato ad almeno un Term.
- **RV3:** Un Qualifier deve essere collegato ad almeno un Concept.
- **RV4:** Un Supplemental deve essere collegato ad almeno un Concept.
- **RV5:** Un Supplemental deve essere collegato ad almeno un HeadingMappedTo.
- **RV6:** Un HeadingMappedTo deve essere collegato ad uno e un solo Descriptor.
- **RV7:** Un PharmacologicalAction deve essere collegato ad uno e un solo Descriptor.
- **RV8:** Un AllowableQualifier deve essere collegato ad uno e un solo Qualifier.

- **RV9:** Ecin deve essere collegato ad uno e un solo Descriptor.
- **RV10:** Ecin deve essere collegato ad uno e un solo Qualifier.
- **RV11:** Ecout deve essere collegato ad uno e un solo Descriptor.
- **RV12:** SeeRelatedDescriptor deve essere collegato a due e due soli Descriptor.
- **RV13:** ConceptRelation deve essere collegato a due e due soli Concept.

4.3 Codice CYPHER

Il linguaggio Cypher proprio di Neo4j, come spiegato nel Capitolo 2, permette la scrittura di operazioni volte a manipolare i dati e la struttura del database creato.

In questo capitolo verranno riportate le istruzioni Cypher per:

- La creazione della **struttura** del database per i dati MeSH;
- L'interrogazione mediante **query** sui dati già scritti all'interno del database.

Sarà spiegata anche la **convenzione** usata per la **denominazione** nelle strutture.

4.3.1 Convenzioni di denominazione

La convenzione utilizzata per i nomi dei nodi, delle proprietà e delle relazioni è quella suggerita dalla stessa organizzazione Neo4J nella sezione "2.9. Naming rules and recommendations" della guida "The Neo4j Getting Started Guide v4.1" al link: <https://neo4j.com/docs/getting-started/current/graphdb-concepts/#graphdb-naming-rules-and-recommendations> .

4.3.2 Struttura

CREATE

```
(`0` :ThesaurusID {thesaurusUI:""}) ,

(`1` :Concept
{conceptUI:"",conceptName:"",registryNumber:"",scopeNote:"",translatorsEnglishScopeNote:"",tr
anslatorsScopeNote:""}) ,

(`2` :RelatedRegistryNumber {registryNumberUI:""}) ,

(`3` :Term
{termUI:"",string:"",dateCreated:"",abbreviation:"",sortVersion:"",entryVersion:"",termNote:""}) ,

(`4` :Descriptor
{descriptorUI:"",descriptorName:"",dateCreated:"",dateRevised:"",dateEstablished:"",annotation:
"",historyNote:"",nlmClassificationNumber:"",onlineNote:"",publicMeshNote:""}) ,

(`5` :Qualifier
{qualifierUI:"",qualifierName:"",dateCreated:"",dateRevised:"",dateEstablished:"",annotation:"",o
nlineNote:"",historyNote:""}) ,

(`6` :Supplemental
{supplementalRecordUI:"",supplementalRecordName:"",dateCreated:"",dateRevised:"",note:"",fre
quency:""}) ,

(`7` :PharmalogicalAction ) ,

(`8` :Substance {recordUI:"",recordName:""}) ,

(`9` :HeadingMappedTo ) ,

(`10` :`Concept Relation` {concept1UI:"",concept2UI:""}) ,

(`11` :AllowableQualifier {abbreviation:""}) ,

(`12` :EntryCombination ) ,

(`13` :SeeRelatedDescriptor ) ,

(`14` :PreviousIndexing {previousIndexingUI:""}) ,

(`15` :TreeNumber {treeNumberUI:""}) ,

(`16` :IndexingInformation ) ,

(`17` :Source {sourceUI:""}) ,

(`18` :ECIN ) ,

(`19` :ECOUT ) ,

(`1`)-[:`CONCEPT_TERM` ]->(`3`),

(`4`)-[:`DESCRIPTOR_CONCEPT` ]->(`1`),
```

```
(`5`)-[:`QUALIFIER_CONCEPT` ]->(`1`),  
(`6`)-[:`SUPPLEMENTAL_CONCEPT` ]->(`1`),  
(`7`)-[:`PHARMACOLOGICAL_ACTION_SUBSTANCE` ]->(`8`),  
(`6`)-[:`SUPPLEMENTAL_HEADING` ]->(`9`),  
(`9`)-[:`HEADING_DESCRIPTOR` ]->(`4`),  
(`3`)-[:`TERM_REFERENCE` ]->(`0`),  
(`1`)-[:`CONCEPT_REGISTRY` ]->(`2`),  
(`10`)-[:`CONCEPT_RELATION_REFERENCE` ]->(`1`),  
(`7`)-[:`PHARMACOLOGICAL_ACTION_DESCRIPTOR` ]->(`4`),  
(`4`)-[:`DESCRIPTOR_ALLOWABLE_QUALIFIER` ]->(`11`),  
(`11`)-[:`ALLOWABLE_QUALIFIER_REFERENCE` ]->(`5`),  
(`13`)-[:`RELATED_DESCRIPTOR_REFERENCE` ]->(`4`),  
(`4`)-[:`PREVIOUS_DESCRIPTOR` ]->(`14`),  
(`4`)-[:`DESCRIPTOR_TREE` ]->(`15`),  
(`5`)-[:`QUALIFIER_TREE` ]->(`15`),  
(`6`)-[:`PREVIOUS_SUPPLEMENTAL` ]->(`14`),  
(`6`)-[:`SUPPLEMENTAL_INDEX` ]->(`16`),  
(`16`)-[:`INDEX_DESCRIPTOR` ]->(`4`),  
(`9`)-[:`HEADING_QUALIFIER` ]->(`5`),  
(`16`)-[:`INDEXING_QUALIFIER` ]->(`5`),  
(`6`)-[:`SUPPLEMENTAL_PHARMACOLOGICAL_ACTION` ]->(`7`),  
(`6`)-[:`SUPPLEMENTAL_SOURCE` ]->(`17`),  
(`12`)-[:`ENTRY_ECIN` ]->(`18`),  
(`18`)-[:`ECIN_DESCRIPTOR` ]->(`4`),  
(`18`)-[:`ECIN_QUALIFIER` ]->(`5`),  
(`12`)-[:`ENTRY_ECOUT` ]->(`19`),  
(`19`)-[:`ECOUT_DESCRIPTOR` ]->(`4`),  
(`19`)-[:`ECOUT_QUALIFIER` ]->(`5`)
```

4.3.3 Query

-- Lista dei possibili nomi e sinonimi di un Descriptor.

```
MATCH (in: Ecin {descriptorName: 'Alzheimer disease'})<-[:ENTRY_ECIN]-(EntryCombination)<-[:ENTRY_ECOUT]-(out:Ecout)
```

```
RETURN out
```

-- Lista delle sostanze coinvolte in un'azione farmacologica.

```
MATCH (p: PharmacologicalAction {descriptorName: 'Alzheimer disease'})<-[:PHARMACOLOGICAL_ACTION_SUBSTANCE]-(s: Substance)
```

```
RETURN s
```

-- Origine di un termine.

```
MATCH (t: Term {String: 'Alzheimer disease'})<-[:TERM_REFERENCE]-(o: ThesaurusID)
```

```
RETURN o
```

-- Il dizionario ThesaurusID con più termini.

```
MATCH (o: ThesaurusID)<-[:TERM_REFERENCE]-(t: Term)
```

```
CALL {
```

```
    CALL {
```

```
        MATCH (o: ThesaurusID)<-[:TERM_REFERENCE]-(t: Term)
```

```
        RETURN o, t, count(*) as cnt
```

```
    }
```

```
RETURN max(cnt) as m
```

```
}
```

```
RETURN o, t, count(*) = m
```

-- Ordinare i Descriptor in base al TreeNumber.

```
MATCH (d: Descriptor) -[:DESCRIPTOR_TREE] -> (t: TreeNumber)
```

```
RETURN d.descriptorName, t.treeNumberUI as num
```

```
ORDER BY num DESC
```

-- Lista dei Descriptor e dei loro nomi precedenti.

```
MATCH (d: Descriptor)-[:PREVIOUS_DESCRIPTOR]->(p: PreviousIndexing)
```

```
RETURN d.descriptorName, p.previousIndexingUI
```


Capitolo 5

5 Conclusioni

In conclusione l'argomento principale della tesi ha riguardato la progettazione di un database a grafo Neo4j a seguito di un'analisi dei dati MeSH.

I database a grafo sono oggi molto utilizzati nei social network soprattutto per la loro flessibilità. Neo4j è uno di questi. Nello specifico, Neo4j si pone anche l'obiettivo di aiutare l'utente ad apprendere il funzionamento del suo sistema grazie ad un set di video tutorial¹² creati da loro e accessibili gratuitamente. La prima fase per la stesura della tesi è stata studiare il sistema Neo4j proprio grazie a questi video e altro materiale inerente.

La fase successiva è stata l'analisi dei dati MeSH volta alla comprensione della struttura degli stessi. L'analisi è iniziata con il download¹³ dei DTD (Document Type Definition) e dei dati presenti nel sistema. È stata, quindi, analizzata la struttura dei dati con l'ausilio di alcuni esempi reali fino alla loro formalizzazione in uno schema concettuale. È iniziata così la fase finale.

La fase finale ha riguardato la progettazione del database iniziando con la stesura dello schema concettuale, passando poi alla sua ristrutturazione e formalizzazione in uno schema logico. Infine, sono state scritte le istruzioni per la creazione del database in Neo4j e alcune possibili query.

Questo progetto lascia spazio a molteplici implementazioni future. Una di queste potrebbe essere la creazione di un software che preleva i dati MeSH in formato XML, ASCII, MARC 21 o RDF¹⁴, inserisca i dati direttamente in un database a grafo Neo4j con la possibilità, inoltre, di gestirli a video nella visualizzazione, modifica, eliminazione e interrogazione.

Tutto il codice sorgente del mio lavoro è disponibile sul mio account GitHub al link:

<https://github.com/bortoletti-giorgia/MeSH-data-in-Neo4j> .

¹² Per ulteriori approfondimenti, visitare il sito:

<https://www.youtube.com/playlist?list=PL9HI4pk2FsvWM9GWaguRhICQ-pa-ERd4U>

¹³ Per ulteriori approfondimenti, visitare il sito: <https://www.nlm.nih.gov/databases/download/mesh.html>

¹⁴ Ibidem

6 Bibliografia

- [1] Neo4j. (2020). *Neo4j Tutorial: Developer Guides*. Tratto da <https://neo4j.com/developer/get-started/>
- [2] NLM. (2020). *Download MeSH Data*. Tratto da <https://www.nlm.nih.gov/databases/download/mesh.html>
- [3] NLM. (2020). *MeSH Browser*. Tratto da <https://meshb.nlm.nih.gov/search>
- [4] Robinson, I., Webber, J., & Eifrem, E. (2015). *Graph Databases*.