
Master research Internship

Bibliographic report

# Gestion de préférences de qualité des données dans les bases de données graphe

**Domain: Databases - Data Structures and Algorithms**

*Author:*
Etienne Scholly

*Supervisors:*
Virginie Thion
Olivier Pivert
SHAMAN / IRISA

ENSSAT
LANNION

**Abstract:** Les façons de stocker les données sont multiples. Un modèle de données qui émerge est celui des bases de données graphe, qui sont des bases de données reposant sur la théorie des graphes. Le stockage de données se fait sous forme de noeuds et d'arcs reliant les noeuds entre eux. L'interrogation repose, elle, sur des langages de requêtes spécifiques à ce type de données.
Quid des données que nous stockons dans nos bases de données ? Les données présentent souvent des problèmes de qualité ; elles peuvent être par exemple imprécises, incomplètes, incohérentes, obsolètes, etc. L'utilisateur, lorsqu'il interroge la base, doit être informé du niveau de qualité des réponses, et doit pouvoir également exprimer des conditions sur ce niveau de qualité.
Dans cette étude, nous allons nous intéresser au problème de gestion de la qualité des données dans les bases de données graphe.

**Keywords:** Bases de données, Modèle de données graphe, Qualité des données

# Contents

# 1 Introduction

Depuis son invention, l'informatique de manière générale ne cesse de se développer à très grande vitesse, et dans certains cas, il peut arriver que des standards, qui étaient autrefois la norme, se retrouvent dépassés par les dernières innovations. Avec l'essor d'internet et la nécessité de représenter et manipuler de nouveaux types d'objets complexes, ainsi que de répondre à de nouveaux besoins allant au delà des applications de gestion classiques [Ang12], les chercheurs se sont intéressés à de nouvelles façons de stocker et interroger les données. Parmi ces nouvelles structures se trouvent les bases de données graphe [AG08], qui sont de plus en plus en vogue en raison de leur adéquation à de nombreux domaines applicatifs. En effet, de nombreuses données sont représentées sous forme de graphe, par exemple des données cartographiques et bibliographiques, des données du web sémantique, des réseaux sociaux ou encore de nombreuses données scientifiques dans des sciences comme la chimie et la biologie [PSST16].

Parallèlement à cela, nous avons à notre disposition des quantités pharaoniques de données, dont la quantité ne cesse d'augmenter jour après jour. Cela peut s'expliquer en partie par le phénomène en pleine croissance de la publication de données dites "ouvertes". Cependant, beaucoup des données disponibles souffrent d'un manque de qualité [ZRM+15], et leur exploitation peut donc s'avérer risquée. Malheureusement, les données sont rarement porteuses de leur niveau de qualité, et beaucoup reste à faire dans le domaine de la gestion de la qualité de ces données [BBK+16]. C'est pourquoi la question de la qualité des données donne lieu à de nombreux travaux actuellement, le but étant de recueillir le maximum d'informations afin de pouvoir donner une indication sur la qualité des données manipulées et exploiter ces données en tenant compte de leur niveau de qualité.

Cependant, malgré le fort intérêt porté à ces deux sujets, et le fait que de nombreuses données disponibles peuvent être représentées sous forme de graphe, l'ajout d'un aspect qualité des données dans une base de données graphe n'est que peu développé.
Dans cette étude, nous nous intéressons aux travaux de la littérature portant sur la gestion de la qualité dans les bases de données graphe. Nous nous focalisons sur la façon d'ajouter des informations qualité aux bases de données graphe, ainsi que la façon d'exploiter ces informations dans les requêtes à patron. Les phases de mesure et d'amélioration de la qualité sont hors périmètre de l'étude. Elles pourront faire l'objet de travaux futurs.

Dans ce document, nous commençons par présenter dans la section 2 les bases de données graphe et les différentes facettes de celles-ci, en parlant notamment des différents modèles qui existent, puis nous parlons du système Neo4j qui sera utilisé dans le cadre du stage. Ensuite, nous voyons dans la section 3 tout ce qui concerne la qualité des données, en particulier en évoquant les principales dimensions qui permettent de traduire cette qualité. Par la suite, dans la section 4, nous tentons d'apporter des pistes pour la gestion de la qualité des données dans une base de données graphe. Enfin, nous dressons une conclusion de cette étude.

# 2 Bases de données graphe

Dans cette section, nous présentons les bases de données graphe de manière d'abord très générale, pour ensuite affiner notre vue et nous concentrer sur le modèle qui nous intéressera dans la suite du travail. Après avoir présenté les méthodes d'interrogation des bases de données graphe, nous terminons cette section par une présentation du système de gestion de bases de données graphe Neo4j, qui est le système utilisé pour notre travail.

## 2.1 Définition

Commençons par donner une définition d'une base de données graphe [BB13].

**Definition 1** *Soit $\mathcal{V}$ un ensemble de noeuds et $\Sigma$ un alphabet fini. Une base de données graphe $\mathcal{G}$ sur $\Sigma$ est un couple $(V, E)$, avec $V$ un ensemble fini de noeuds (i.e., $V \subseteq \mathcal{V}$) et $E \subseteq V \times \Sigma \times V$. Ainsi, chaque arc de $\mathcal{G}$ est un triplet $(v, a, v') \in V \times \Sigma \times V$, qui constitue un arc de label $a$ reliant $v$ à $v'$ dans $\mathcal{G}$.*

De manière moins théorique, dans les bases de données graphe, les données sont stockées sous forme de noeuds et d'arcs ; les noeuds sont reliés entre eux par des arcs. Ces arcs désignent la relation que les noeuds ont entre eux [AG08].

Lorsque nous manipulons des données pour lesquelles les interconnexions sont importantes, utiliser une base de données graphe nous donne un avantage sur la modélisation des données : elle est plus naturelle. En effet, les données structurées en graphe ont une meilleure visibilité pour l'utilisateur, et permettent une manipulation plus naturelle des données. En outre, les bases de données graphe ont l'avantage de pouvoir conserver toute l'information d'une entité dans un seul noeud, et de montrer via les arcs connectés au noeud les connexions aux autres entités de la base [AG08].

La figure 1 ci-dessous donne un exemple d'une base de données graphe, de modèle graphe à attributs permettant d'ajouter aux données des caractéristiques sous forme d'attributs (modèle qui sera présenté plus loin dans le rapport). Cette base de données graphe concerne des films (en vert) et les personnes (en bleu) qui ont interagit avec ces films. Les interactions sont désignées par les arcs reliant les noeuds entre eux. Par exemple, nous pouvons voir que Tom Hanks a joué dans le film *The Da Vinci Code*, que Ron Howard a réalisé ce même film, que Jessica Thompson en a fait une critique et que Jason Thompson suit Jessica Thompson. Tom Hanks a à la fois réalisé le film *That Thing You Do* et joué dans celui-ci, c'est pourquoi il y a deux arcs qui le relient au film, pour désigner ces deux relations.
De plus, les attributs sont visibles sous chaque noeud ; une personne a comme attribut son année de naissance et un film son année de sortie.

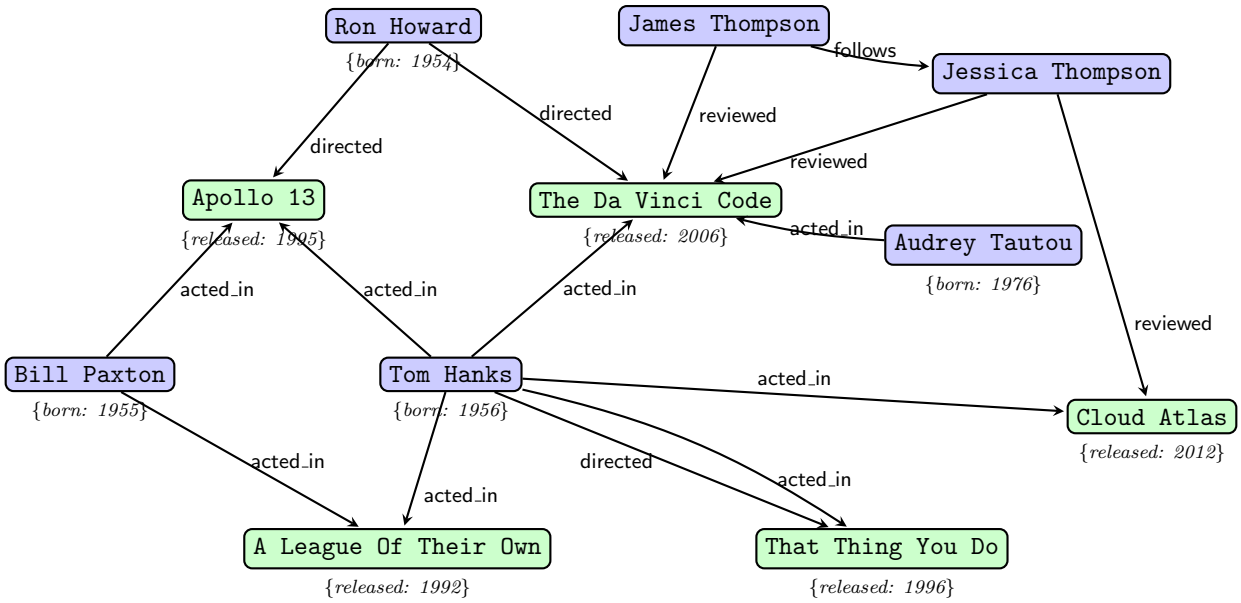


Figure 1: Un exemple de base de données graphe

## 2.2 Modèles

Comme évoqué dans le paragraphe précédent, une base de données graphe est définie par un ensemble de noeuds reliés entre eux par des arcs. Il existe plusieurs modèles de bases de données graphe, chacun ayant des caractéristiques propres [Ang12], et c'est justement ce que nous présentons ici.

**Graphe simple** Commençons par le modèle le plus simple. Comme son nom l'indique, nous parlons ici de la structure la plus basique : les noeuds sont reliés par des arcs, tout simplement. Le graphe est plat, les arcs et les noeuds ne sont pas porteur d'attribut.

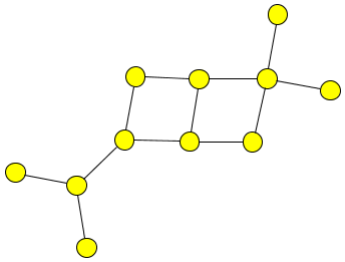

Figure 2: Un graphe simple [BEJ]

**Hypergraphe** Ceci constitue une extension du modèle de graphe simple, où les arcs ne relient plus deux noeuds entre eux mais plusieurs. Par conséquent, les arcs ne sont plus simplement des segments reliant deux noeuds mais des surfaces qui englobent plusieurs noeuds. Nous parlons dans ce cas d'hyperarcs.

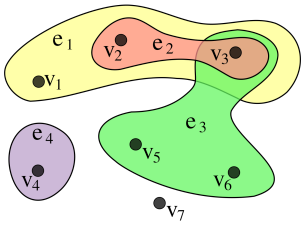


Figure 3: Un hypergraphe [Wik]

**Graphe imbriqué** C'est une autre extension du graphe simple : dans cette structure, les noeuds du graphe peuvent eux aussi être des graphes. Dans ce cas, nous les désignons alors comme des hypernoeuds.

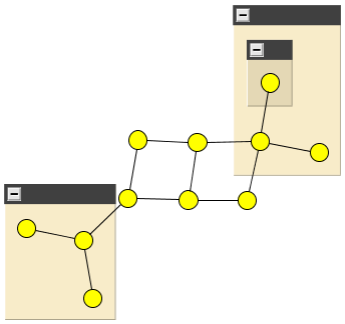

Figure 4: Un graphe imbriqué [BEJ]

**Graphe à attributs** À nouveau, ce modèle est une extension du graphe simple. Dans un graphe à attributs, les noeuds et les arcs peuvent contenir des attributs permettant de décrire leurs propriétés.

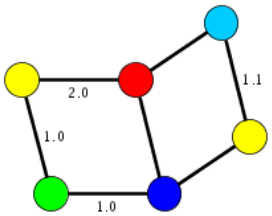


Figure 5: Un graphe à attributs [BEJ]

En plus des types de graphes présentés ci-dessus, nous avons aussi la possibilité d'avoir des arcs dirigés ou non, pour désigner un sens dans la relation entre deux entités. En reprenant l'exemple donné plus haut, un acteur joue dans un film, l'arc reliant la personne au film est donc dirigé, de la personne vers l'arc. Un arc non dirigé est une relation bilatérale entre deux personnes, par exemple deux époux, qui ont comme relation "mariés".

Dans la suite de ce document, nous nous limiterons à l'étude du modèle de graphe à attributs, qui est le modèle de données graphe le plus courant.

## 2.3 Interrogation

Comment interroge-t-on une base de données graphe ? La question est légitime puisqu'il n'est pas facile de deviner instinctivement ce qui va se passer. Les langages de requête sur les bases de données graphe sont basés sur la correspondance de modèles de graphe (*pattern matching*) [BB13]. A partir d'un patron, à savoir une forme de "morceau" de graphe, nous voulons trouver toutes les parties de la base de données graphe qui sont semblables au motif donné [AG08]. Une requête est un patron de graphe, c'est-à-dire un graphe dans lequel peuvent apparaître des variables et des conditions. Ceci constitue la syntaxe du langage de requête. Le résultat d'une requête est constitué de l'ensemble des sous-graphes de la base de données de départ qui correspondent au patron donné. Avec ceci, nous donnons un sens à notre requête, c'est la sémantique du langage de requête.

La figure 6 donne un exemple de requête sous forme de patron de graphe que nous souhaitons retrouver dans notre base de données. Cette requête consiste à chercher tous les acteurs qui ont joué avec Tom Hanks. Les noeuds ?`movie` et ?`coActor` sont des variables de noeuds.

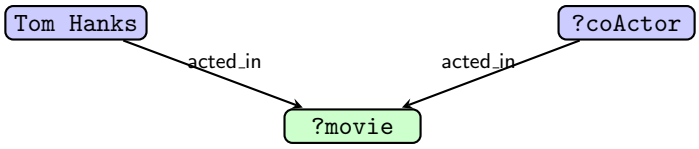


Figure 6: Un exemple de patron d'une requête

En interrogeant la base de données graphe de la figure 1 avec la requête ayant comme patron celui présenté figure 6, nous obtenons le résultat présenté figure 7. Nous obtenons trois résultats pour cette requête.

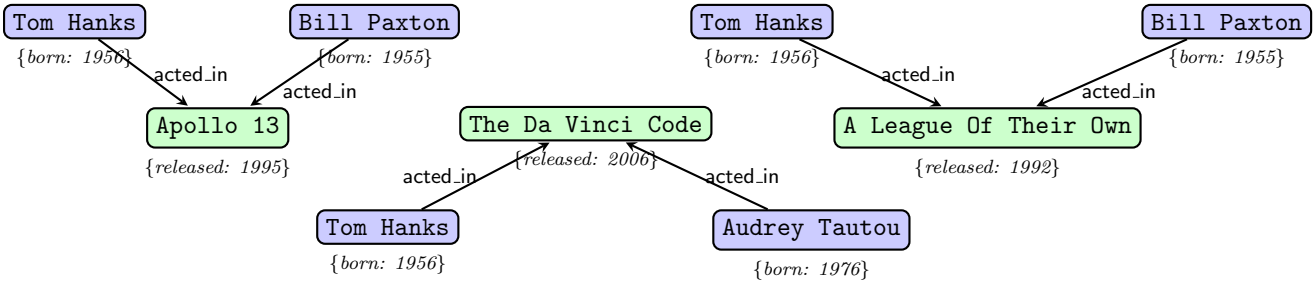


Figure 7: La réponse de la requête

## 2.4 Neo4j et Cypher

Neo4j est un système de gestion de bases de données orientées graphe [Incb]. Il utilise le modèle des graphes à attributs et propose un langage de requête appelé Cypher, qui lui est basé sur la correspondance de *pattern* comme évoqué dans la partie précédente [Inca].

Une requête Cypher se présente de la manière suivante :

`(n1:Type1)-[exp]->(n2:Type2)`
ou
`(n1:Type1)-[e:label]->(n2:Type2)`

où `n1` et `n2` sont des variables de noeuds, `e` est une variable d'arc, `label` est un label de $E$, `exp` est une expression régulière et `Type1` et `Type2` sont des types de noeuds. Une telle expression donne un chemin qui satisfait l'expression régulière *exp* (c'est "simple" dans la seconde forme $e$) allant d'un noeud du type `Type1` à un noeud du type `Type2`.

Tous les arguments passés en paramètre de la requête sont optionnels, donc la forme la plus simple d'une requête est `()-[]->()` désignant un chemin constitué de deux noeuds connectés par un arc quelconque. D'éventuelles conditions sur les attributs des noeuds et/ou des arcs sont fournies dans la clause WHERE, de manière assez similaire à ce qui se fait en SQL pour les bases de données relationnelles.

Le listing 1 donne la requête utilisée dans le paragraphe précédent mais avec la syntaxe du langage Cypher. La base de donnée utilisée dans Neo4j est la même que celle présentée précédemment. La figure 8 ci-dessous donne les réponses obtenues après avoir appliqué la requête à notre base de données graphe. Nous retrouvons à nouveau trois fois le patron donné pour la requête. Nous voyons bien le "noeud" modélisant Tom Hanks présent trois fois, associé à chaque fois à un film différent et à un acteur pour chaque film.

```
1 MATCH
2     (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(movies)<-[:ACTED_IN]-(↩
      coActors)
3 RETURN tom, movies, coActors
```

Listing 1: Un exemple de requête Cypher

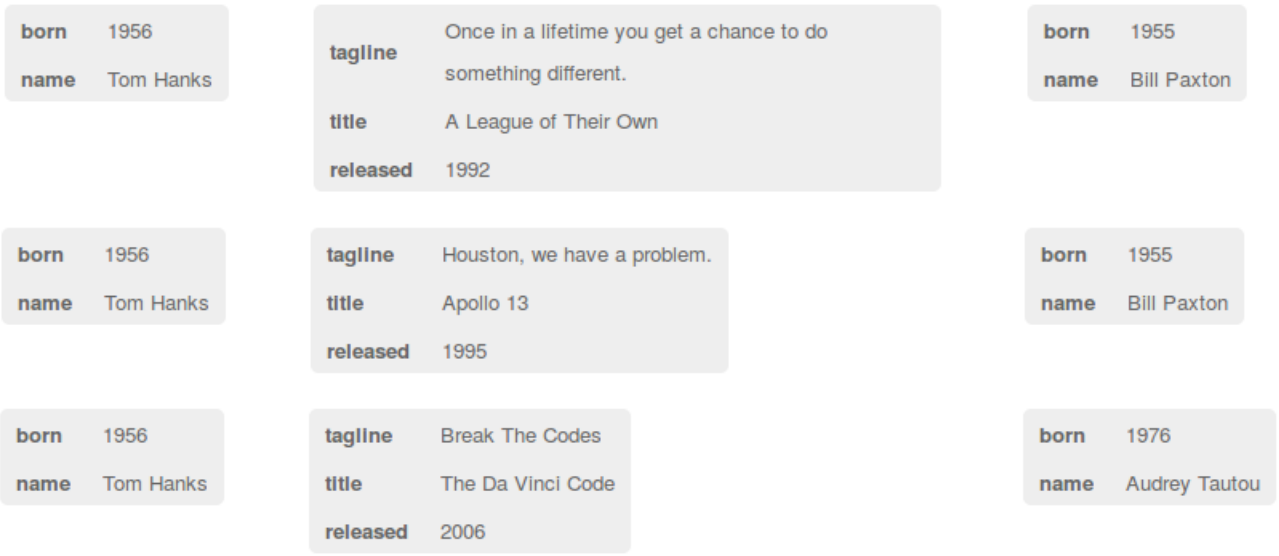


Figure 8: La réponse de la requête sur le graphe de données avec Neo4j

# 3 Qualité des données

Après avoir passé en revue le sujet des bases de données graphe en s'attardant plus en détail sur Neo4j et le modèle qu'il utilise, nous allons maintenant nous intéresser à la qualité des données de manière générale. Commençons par évoquer les dimensions de la qualité des données.

## 3.1 Dimensions

La qualité des données est désignée comme étant l'aptitude à l'emploi des données (en anglais, *fitness for use)*. Pour caractériser la qualité des données, les moyens sont multiples [ZRM+15].

Nous voyons dans ce paragraphe quelques dimensions de la qualité qu'il est possible de qualifier de "principales" puisque celles-ci font consensus dans la littérature scientifique [BS16a].

**Exactitude** C'est la première chose à vérifier lors de la vérification de la qualité des données : sont-elles exactes ? Il existe deux types d'exactitude : l'exactitude syntaxique et l'exactitude sémantique. L'exactitude syntaxique, facile à mettre en oeuvre, vérifie que la donnée suit bien le format requis, par exemple que le prix d'un article est bien un nombre. Cet aspect est très souvent "automatisé" dans les bases de données. L'exactitude sémantique, elle, est bien plus difficile à vérifier. Nous voulons savoir si notre information est correcte, mais dans beaucoup de cas, il est très difficile de le savoir sans avoir recours à une intervention humaine. Prenons l'exemple d'un numéro de téléphone associé à un client, le seul moyen de vérifier son exactitude, i.e. qu'il est associé à la bonne personne, est d'appeler ce numéro.

**Complétude** La complétude désigne le fait que le jeu de données est complet. Comme pour l'exactitude, il y a deux types de complétude. La complétude "horizontale" permet de vérifier qu'une entité possède une valeur pour toutes ses caractéristiques. Il est aisé d'imager cette dimension en prenant l'exemple d'une base de données relationnelle classique : un tuple est complet si tous ses attributs sont renseignés. D'autre part, nous avons la complétude "verticale", qui vérifie cette fois que toutes les entités à observer sont présentes dans les données. En reprenant l'analogie de la base de données relationnelles, une table Personne doit contenir toutes les personnes à étudier. Dans le cadre des bases de données graphe, une mesure de complétude "horizontale" consisterait à vérifer que tous les acteurs ont bien une année de naissance associée (en reprenant l'exemple de la base de données graphe de la section 2), et une mesure de complétude "verticale" vérifierait que tous les films de Tom Hanks sont représentés.

**Redondance** Il s'agit ici d'éviter les doublons, i.e. le fait d'avoir une même information en double (ou plus). Parfois, ces redondances peuvent être très difficiles à détecter, et peuvent aussi poser des problèmes d'exactitude, si par exemple nous avons deux fois le même individu avec une valeur d'attribut différente dans chaque version.

**Consistance** Cette dimension désigne la cohérence des données. Nous faisons référence ici au fait que les données respectent toutes les règles et propriétés nécessaires pour avoir des données potentiellement véridiques. Nous ne pouvons réellement savoir si une information est juste, mais nous pouvons par contre savoir si elle est fausse, et c'est le but des règles de vérification mises en place pour vérifier la cohérence des données. Par exemple, si nous considérons un annuaire téléphonique, les personnes vivant en Bretagne doivent avoir un numéro de téléphone fixe qui commence par 02.

**Lisibilité** Cette dimension fait référence à la compréhensibilité, à la clarté et à la simplicité des données. Si ces aspects sont bons, le lecteur peut alors comprendre aisément les données et peut en tirer facilement profit. Par exemple, nous pouvons évoquer les noms des attributs dans une base de données. Si tous les attributs sont nommés `attr_1`, `attr_2`, et ainsi de suite, la lisibilité est très mauvaise. En donnant un nom clair à chaque attribut (`nom`, `adresse`, par exemple), les données sont déjà bien plus lisibles.

**Confiance** Enfin, cette dimension nous donne une indication sur le degré de confiance que nous pouvons avoir par rapport aux données dont nous disposons. Pour cela nous pouvons regarder notamment la réputation des sources. Nous pouvons aussi considérer la crédibilité, qui est le degré auquel l'information est acceptée pour être correcte, vraie, réelle et crédible ; ainsi que la vérifiabilité qui fait référence au degré auquel l'utilisateur peut évaluer la justesse d'un ensemble de données [ZRM+15].

Les dimensions étant définies, il faut maintenant procéder à la mesure de ces dimensions. Il existe de nombreuses mesures, ou métriques, permettant de caractériser chaque dimension [ZRM+15] [BS16a]. Une métrique peut se placer à différents niveaux : sur un ou plusieurs attribut(s), sur un ou plusieurs tuple(s), sur une ou plusieurs colonne(s), voire même sur la base de donnée en entier. De plus, la mesure d'une dimension peut se faire de manière automatique ou non ; si elle n'est pas automatique, la mesure de la dimension requiert une intervention humaine. Enfin, la mesure d'une dimension peut-être objective, par exemple le nombre de tuples incomplets dans une base de données ; ou subjective, en interrogeant par exemple des utilisateurs sur la lisibilité d'un schéma.

## 3.2 Explications à travers un exemple

Pour illustrer chaque dimension, nous reprenons la base de données graphe présentée dans la section 2, avec cette fois-ci de nouvelles données volontairement dégradées pour illustrer les dimensions de la qualité des données [BS16a]. Les données qui posent problème sont de couleur rouge. La base de données est présentée figure 9.

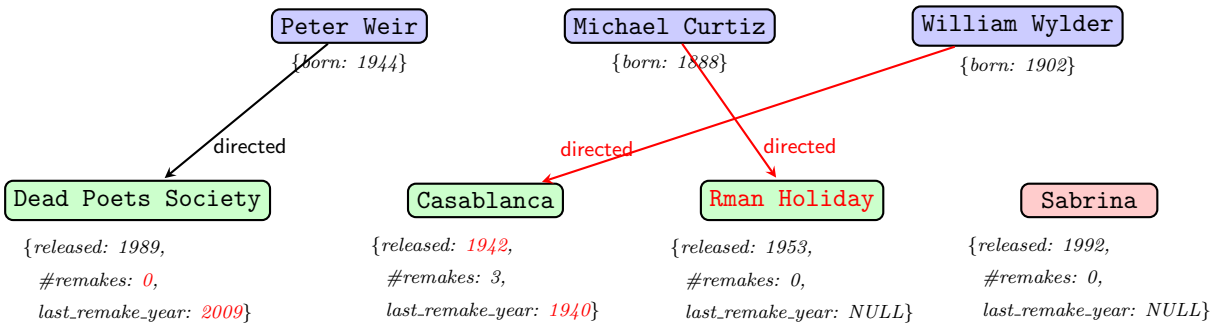


Figure 9: Une base de données graphe erronée

La première erreur que nous pouvons relever concerne le film *Rman Holiday*, qui en réalité n'existe pas : nous sommes en présence d'une erreur d'exactitude syntaxique, la donnée est inexacte. Le titre *Roman Holiday*, qui correspond au titre d'un film existant, est le plus proche du titre erroné, avec une distance d'édition de 1, qui correspond à l'ajout de la lettre "o".

Regardons maintenant le noeud décrivant le film *Sabrina*. Celui-ci n'est relié à aucun réalisateur par aucun arc, ce qui est problématique puisque le film a forcément eu un réalisateur : nos données sont donc incomplètes.

Pour le film *Dead Poets Society*, nous avons une indication qui dit que le film n'a pas eu de remake (la valeur de l'attribut #remakes vaut zéro), mais pourtant, nous avons une date dans l'attribut qui donne l'année du dernier remake. Nos données sont donc incohérentes, puisque nous ne pouvons avoir une année de dernier remake si aucun remake n'a été fait.

Pour le second film, *Casablanca*, nous avons comme information qu'il a donné lieu à 3 remakes, mais l'année du dernier remake est 1940, alors que le film est sorti en 1942. Nous avons donc là encore un problème de cohérence, puisqu'un remake n'a pas pu sortir avant le film.

Enfin, jetons un oeil aux liens qui connectent les deux derniers réalisateurs et leurs films. Il y a une erreur, mais il n'y a pas de problème apparent, tout est syntaxiquement correct. L'erreur est cette fois-ci une inexactitude sémantique : les deux arcs ont simplement été inversés. Nous voyons ici que cet aspect de la dimension d'exactitude est très difficile à vérifier puisque nous ne pouvons pas affirmer que l'information que nous avons est fausse... sans pouvoir affirmer qu'elle est vraie pour autant.

Dans le cas de la dernière erreur évoquée, nous pourrions utiliser des règles de consistance pour vérifier que nos données sont au moins plausibles à défaut d'être vraies, en comparant par exemple l'année de naissance des réalisateurs avec l'année de sortie des films. Dans notre cas ici, il n'est pas possible de tirer de conclusion avec cette comparaison puisque cela paraît plausible. Si nous avions eu Michael Curtiz comme réalisateur de *Sabrina*, nous aurions pu dire qu'il est peu probable de faire un film à 104 ans, et donc que nous avons une confiance assez peu élevée dans nos données.

# 4 Qualité des données dans les bases de donnés graphe

Dans les sections précédentes, nous avons présenté les bases de données graphes, puis la qualité des données. Nous voulons maintenant combiner ces deux domaines pour parler de la qualité des données dans les bases de données graphe ; pour cela, cette section commence par parler de la représentation des informations qualité dans les bases de données graphe.

## 4.1 Représentation des informations qualité dans les bases de données graphe

Actuellement, il n'existe pas de contribution scientifique concernant la gestion de la qualité dans les données de type graphe à attributs, qui est le modèle de bases de données graphe qui nous intéresse. Les travaux qui s'approchent le plus de notre problème sont ceux portant sur la gestion de la qualité dans les données RDF.

Dans l'optique de donner à l'utilisateur des informations qualité sur les données qu'il manipule, et ce de manière cohérente, une norme du W3C sur l'ajout de ces informations qualité dans les données RDF a été définie [W3C]. L'objectif est de pouvoir associer des informations qualité aux données en respectant un vocabulaire prédéfini. Ainsi, l'utilisateur peut se faire son propre jugement avec les informations qu'il a. Pour un ensemble de données provenant d'une source et répondant à un certain standard, nous ajoutons aussi une annotation qualité, qui représente le retour d'information et le certificat de qualité du jeu de données étudié. En marge de l'annotation qualité peut aussi être

fournie la valeur d'une mesure de la qualité, qui donne à l'utilisateur une information qualitative ou quantitative à propos de l'ensemble de données.

En plus du travail effectué sur l'ajout d'informations qualité dans les données RDF, des travaux ont aussi été menés sur l'ajout de l'aspect qualité des données dans des bases de données XML [BS16b]. L'idée est de rajouter les informations qualité à la base de données XML en créant un second arbre ayant une structure identique à l'arbre de données initial. Ce second arbre contient les informations qualité et chaque noeud est relié au noeud de l'arbre initial avec un lien spécial. Ainsi, pour chaque noeud dans l'arbre qualité, il est possible d'ajouter une mesure pour diverses dimensions de la qualité pour la donnée présente au noeud équivalent dans l'arbre de données.

### 4.2 Les requêtes *quality-aware*

En voulant travailler avec une base de données qui contient des erreurs, nous avons deux solutions. La première est de procéder à un nettoyage de cette base de données, qui peut parfois s'avérer risqué, étant donné la difficulté de trouver les erreurs dans certains cas, comme énoncé dans la section 3. L'autre solution est de "vivre" avec ces erreurs le mieux possible, en adaptant le langage de requête associé à la base de données, d'une part en informant l'utilisateur de ces problèmes en lui fournissant les informations qualité à notre disposition, d'autre part en prenant en compte les préférences qualité de l'utilisateur dans ses requêtes. Nous nous inscrivons dans cette optique.

L'un des moyens permettant de vivre avec les erreurs est d'offrir un langage de requête permettant à l'utilisateur d'exprimer des préférences sur la qualité des données qu'il veut interroger. Il n'existe actuellement pas de travaux sur le requêtage de bases de données graphe prenant en compte des informations qualité. Les travaux les plus proches concernent le requêtage à préférences, plus particulièrement sur des bases de données graphe.

Les motivations pour introduire des préférences dans les requêtes de base de données sont multiples [HKP11]. La première idée est d'offrir des langages de requête plus expressifs qui peuvent être plus fidèles à ce que l'utilisateur souhaite réellement exprimer (par exemple, les acteurs "jeunes"). Deuxièmement, l'introduction de préférences dans les requêtes fournit une base pour classer les articles récupérés, ce qui est particulièrement utile dans le cas d'un grand nombre d'éléments répondant à la requête. Enfin, une requête "classique" peut également avoir un ensemble vide de réponses, tandis qu'une requête floue peut potentiellement trouver des éléments participant à la réponse.

Au lieu d'avoir simplement des conditions qui s'avèrent vraies ou fausses, l'utilisateur voudrait pouvoir exprimer ses préférences de façon flexible, que le système de requêtage doit pouvoir prendre en compte. Les préférences floues peuvent être de deux types [PST15]. Dans un premier temps, une préférence floue peut être portée sur les noeuds, où l'idée est d'exprimer des conditions flexibles concernant les attributs associés aux noeuds et / ou arcs du graphe, en demandant par exemple des acteurs "jeunes". Une préférence floue peut aussi être exprimée sur des aspects structurels du graphe, en exprimant des conditions sur la longueur ou la force de chemins, par exemple.

Dans ces travaux, les auteurs ne prennent pas en compte la qualité des données du graphe.

# 5 Conclusion

Dans ce rapport, nous avons présenté les bases de données graphe, qui permettent de stocker des données sous formes de noeuds reliés entre eux par des arcs, ceux-ci donnant la relation qu'ont les noeuds connectés. Il existe plusieurs modèles de bases de données graphe et celui qui nous intéresse est le modèle des graphes à attributs. Une base de données graphe s'interroge avec un langage de requête basé sur la recherche de patron dans notre structure. Neo4j et Cypher sont respectivement un système de gestion de bases de données graphe et un langage de requête pour des bases de données graphe, que nous utilisons pour ce travail.

Nous avons aussi présenté la qualité des données, en parlant particulièrement des dimensions qui permettent d'avoir une idée du niveau de qualité des données manipulées. Ces dimensions peuvent ensuite être mesurées en définissant une métrique pour avoir une valeur à exploiter. A travers cette section nous avons vu un exemple sur une base de données graphe qui présente des erreurs, afin d'illustrer les dimensions de la qualité.

Après cela, nous avons parlé de l'ajout d'informations qualité dans une base de données graphe. Le sujet n'est pas développé pour le modèle des graphes à attributs, c'est pourquoi nous avons évoqué les domaines les plus proches, à savoir la qualité des données RDF dans les bases de données graphe afin de voir comment ajouter ces informations qualité dans une base de données graphe à attributs. Ensuite, nous avons vu les langages de requête à préférences dans le cas des bases de données graphe floues, qui expriment un degré d'appartenance d'un élément à un ensemble flou. Ces travaux nous permettent d'avoir une idée de comment adapter un langage de requêtes à préférences pour une base de données graphe portant des informations qualité.

Dans le cadre du problème qui nous intéresse, il reste donc un certain nombre de verrous scientifiques à lever. En premier lieu, il faut creuser sur la modélisation des informations qualité dans les bases de données graphe, avec comme principale question : comment pouvons-nous ajouter ces informations dans la base ? Pour cela, nous allons devoir étendre le modèle de la définition 1 pour attacher aux bases de données graphe des informations sur la qualité des données stockées.
L'autre grand problème est l'interrogation de la base de données graphe porteuse d'informations qualité. Pour cela, il faut travailler sur l'ajout de l'aspect *quality-aware* au langage de requêtes à préférences qui peut interroger de telles bases de données graphe. Il est donc nécessaire de définir la syntaxe et la sémantique d'une extension du langage de requête Cypher permettant cela.

Par la suite, en ayant étendu la notion de bases de données graphe pour y ajouter les informations qualité, puis en ayant travaillé sur un langage à préférences pour prendre en compte les préférences qualités, notre objectif sera d'implanter tout cela à l'aide de Neo4j, Cypher et le langage de requêtes à préférences FUDGE.

## References


[AG08] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1, 2008.

[Ang12] Renzo Angles. A comparison of current graph database models. In *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on*, pages 171–177. IEEE, 2012.

[BB13] Pablo Barceló Baeza. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, pages 175–188. ACM, 2013.

[BBK+16] Delphine Barrau, Nathalie Barthélémy, Zoubida Kedad, Brigitte Laboisse, Sylvaine Nugier, and Virginie Thion. Gestion de la qualité des données ouvertes liées - État des lieux et perspectives. *Revue des Nouvelles Technologies de l'Information*, 2016.

[BEJ] Ulrik Brandes, Markus Eiglsperger, and Lerner Jürgen. GraphML primer. `http://graphml.graphdrawing.org/primer/graphml-primer.html`.

[BS16a] Carlo Batini and Monica Scannapieco. Data quality dimensions. In *Data and Information Quality*, pages 21–51. Springer, 2016.

[BS16b] Carlo Batini and Monica Scannapieco. Models for information quality. In *Data and Information Quality*, pages 137–154. Springer, 2016.

[HKP11] Allel Hadjali, Souhila Kaci, and Henri Prade. Database preference queries—a possibilistic logic approach with symbolic priorities. *Annals of Mathematics and Artificial Intelligence*, 63(3-4):357–383, 2011.

[Inca] Neo Technology Inc. Cypher. `https://neo4j.com/docs/developer-manual/current/cypher/`.

[Incb] Neo Technology Inc. Neo4j: The world's leading graph database. `https://neo4j.com/`.

[PSST16] Olivier Pivert, Olfa Slama, Grégory Smits, and Virginie Thion. SUGAR: A graph database fuzzy querying system. In *Research Challenges in Information Science (RCIS), 2016 IEEE Tenth International Conference on*, pages 1–2. IEEE, 2016.

[PST15] Olivier Pivert, Grégory Smits, and Virginie Thion. Expression and efficient processing of fuzzy queries in a graph database context. In *Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on*, pages 1–8. IEEE, 2015.

[W3C] W3C. Data quality vocabulary. `https://www.w3.org/TR/2015/WD-vocab-dqv-20150625/`.

[Wik] Wikipedia. Hypergraphe. `https://fr.wikipedia.org/wiki/Hypergraphe`.

[ZRM+15] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2015.