Imperfect RDF Databases : From Modelling to Querying

Amna Abidi

THESE

Pour l'obtention du Grade de

DOCTEUR DE L'ECOLE NATIONALE SUPERIEURE DE MECANIQUE ET D'AEROTECHNIQUE DE POITIERS et DE L'INSTITUT SUPERIEUR DE GESTION DE TUNIS

(Diplôme National - Arrêté du 25 Mai 2016)

Ecole doctorale: Sciences et Ingénierie des Systèmes, Mathématiques, Informatique

Secteur de recherche: INFORMATIQUE ET APPLICATIONS

Présentée par

Amna ABIDI

Imperfect RDF Databases: From Modelling to Querying

Directeurs de thèse: **Allel HADJALI** et **Boutheina BEN YAGHLANE**

****

Soutenue le 11 Juin 2019

devant la commission d'examen

****

- JURY -

Rapporteurs:				
Président de jury	Djamal	BENSLIMANE	Professeur	Univ. Lyon 1, France
	Hajer	BAAZAOUI	MC-HDR	Univ. de la Manouba, Tunisie
Membres de jury:				
	Juliette	DIBIE BARTHELEMY	Professeur	AgroParisTech, France
	Salah	BEN ABDALLAH	Professeur	Univ. de Tunis, Tunisie
	Allel	HADJALI	Professeur	ENSMA, France
	Boutheina	BEN YAGHLANE	Professeur	Univ. Carthage, Tunisie
	Mohamed Anis	BACH TOBJI	MC-HDR	Univ. de la Manouba, Tunisie

Abstract

The ever-increasing interest of RDF (Resource Description Framework) data on the Web has led to several and important research efforts to enrich traditional RDF data formalism for the exploitation and analysis purpose. The work of this thesis is a part of the continuation of those efforts by addressing the issue of RDF data management in presence of imperfection (untruthfulness, uncertainty, etc.). As a first part of our dissertation, we particularly tackled the trusted RDF data model. Hence, we proposed to extend the skyline queries over trust RDF data, which consists in extracting the most interesting trusted resources according to user-defined criteria. To this end, we introduced appropriate semantics of the Trust-skyline, the set of the most interesting resources in a trust RDF dataset. Then, we studied via statistical methods the impact of the trust measure on the Trust-skyline set.

Second, we integrated in the structure of RDF data (i.e., subject-property-object triple) a fourth element expressing a possibility measure to reflect the user opinion about the truth of a statement. To deal with possibility requirements, appropriate framework related to language is introduced, namely Pi-SPARQL, that extends SPARQL to be possibility-aware query language.

Finally, we studied a new skyline operator variant to extract possibilistic RDF resources that are possibly dominated by no other resources in the sense of Pareto optimality. We introduced a dominance relation and a skyline model adapted to the aforementioned kind of data. For experiments, we introduced a new algorithm which outperforms the naive methods. As a first step, it summarizes the region of data explored in earlier iterations. Indeed, new candidate skyline points are compared to this summary rather than the rest of the skyline candidates. In addition, as reducing the number of dominance checks is crucial, another optimization is added by reducing the complexity of the dominance function using the properties of the new variant skyline defined on possibilistic RDF data.

Keywords: Semantic web, Ressource Description Framework, SPARQL, Trust, Uncertainty, Possibility theory, Preference queries, Skyline queries, Querying, User-centered system design

Résumé

L'intérêt sans cesse croissant des données RDF (Resource Description Framework) disponibles sur le Web a conduit à l'émergence de multiple et importants efforts de recherche pour enrichir le formalisme traditionnel des données RDF à des fins d'exploitation et d'analyse. Le travail de cette thèse s'inscrit dans la continuation de ces efforts en abordant la problématique de la gestion des données RDF en présence d'imperfections (manque de confiance/validité, incertitude, etc.). Dans la première partie de ce mémoire, nous nous sommes intéressés aux données RDF pondérées par un degré de confiance (Trust-RDF). En effet, nous avons proposé d'appliquer l'opérateur skyline sur ces données dans le but d'extraire les ressources les plus confiantes selon des critères définis par l'utilisateur. Pour cela, nous avons introduit une sémantique appropriée du Trust-skyline, l'ensemble des ressources les plus intéressantes extraites d'un ensemble de données RDF pondérées par un degré de confiance. Ensuite, nous avons discuté via des méthodes statistiques l'impact des mesures de confiance sur le Trust-skyline.

Dans une deuxième contribution, nous avons intégré à la structure des données RDF (au niveau du triplet sujet-prédicat-objet) un quatrième élément, exprimant une mesure de possibilité, pour traduire l'expression de la véracité d'une déclaration. Pour gérer cette mesure de possibilité, un cadre langagier appropriée est étudié, à savoir Pi-SPARQL, qui étend le langage SPARQL aux requêtes permettant de traiter des distributions de possibilités.

Enfin, nous avons étudié une variante d'opérateur skyline pour extraire les ressources RDF possibilistes qui ne sont éventuellement dominées par aucune autre ressource dans le sens de l'optimalité de Pareto. Nous avons introduit une relation de dominance et un modèle skyline adaptés pour le contexte de données RDF possibilistes. Pour valider notre approche, et à des fins d'expérimentations, nous avons proposé un nouvel algorithme plus performant que la méthode naïve. Comme première étape, il résume les données précédemment explorées. Ainsi, les nouveaux candidats du skyline ne seront comparés qu'avec ce résumé plutôt qu'avec tout l'ensemble des points. De plus, et comme la réduction du nombre des comparaisons est cruciale, nous avons proposé une autre optimisation en réduisant la complexité de la fonction de dominance en utilisant les propriétés de la nouvelle variante du skyline introduite sur des données RDF possibilistes.

Mots-clés: Web sémantique, Ressource Description Framework, SPARQL, Bases de Degré de confiance, Incertitude, Théorie des possibilités, Requêtes à préférences, Opérateur Skyline, données–Interrogation, Conception centrée sur l'utilisateur

Guide de lecture

Le Manuscrit est composé de deux parties contenant deux chapitres chacune, les deux premiers chapitres présentent le background nécessaire à la compréhension des contributions présentées dans les autres chapitres. Les deux derniers chapitres présentent les contributions de ce travail.

Chapitre 1: RDF Formalism

Ce chapitre rappelle les principaux concepts et formalismes des données RDF (Resource Description Framework) ainsi que les bases du langage de requêtes SPARQL. Il présente ensuite les différents modèles de données RDF incertaines (Trust RDF, Probabilistic RDF, etc).

Chapitre 2: Background on Possibility Theory and Skyline Queries

Ce chapitre est aussi lié à la première partie présentant les préliminaires. Il présente les notions de la théorie des possibilités. La théorie des possibilités est comparée aux autres théories d'incertitude: la théorie des probabilités et la théorie de l'évidence.

Il présente ensuite les requêtes de type skyline dans le cas certain ainsi que dans le cas des données incertaines (skyline probabiliste, skyline possibiliste, skyline stochastique, skyline évidentiel). Pour la deuxième partie, nous présentons les contributions de ce travail de thèse.

Chapitre 3: Trust Skyline Model Semantics and Experimentations

Il présente deux principales contributions. La première contribution vise à étendre l'opérateur classique de skyline au cas des données RDF pondérées par des degrés de confiance (Trust-Weighted RDF data). Le nouvel opérateur est appelé Trust-skyline et l'extension proposée est principalement basée sur l'adaptation de l'opérateur de dominance de sorte à ce qu'il produise non pas une dominance binaire (1 si domine ou 0 si ne domine pas) mais plutôt une dominance graduelle pondérée (un point X domine un point Y à un certain degré). Pour le calcul du Trust-skyline de nouvelles méthodes sont introduites. Un algo-

rithme naif appelé Naive T-skyline est introduit, ainsi qu'un algorithme optimisé appelé TRDF-Skyline basé sur l'exploitation de certaines propriétés de la sémantique associée à l'opérateur Trust-skyline proposé. Les résultats de cet algorithme sont comparés à ceux du Naive T-skyline. La deuxième contribution de ce chapitre consiste à analyser les résultats du Trust-skyline via l'utilisation de méthodes statistiques. L'objectif est d'étudier l'impact de la mesure du seuil de confiance ALPHA (trust measure) sur le résultat du T-skyline. Ainsi, avec une petite valeur d'ALPHA les résultats du T-Skyline sont de petite taille, alors que pour une valeur moyenne ou grande du seuil de confiance, les résultats du T-skyline peuvent être volumineux puisque plusieurs points, ayant un trust inférieur à ALPHA, entrent directement au T-skyline.

Chapitre 4: Possibilistic RDF Data

Ce chapitre fait partie des contributions de cette thèse. Il propose d'abord la modélisation des données RDF incertaines à travers la théorie des possibilités. Ensuite, un langage "Possibility-aware" nommé Pi-SPARQL a été proposé pour supporter les besoins de requêtage de données RDF possibilistes.

Enfin une variante d'opérateur skyline est proposée pour extraire les ressources RDF possibilistes qui ne sont éventuellement dominées par aucune autre ressource dans le sens de l'optimalité de Pareto. Une relation de dominance et un modèle skyline adaptés pour le contexte de données RDF possibilistes sont introduits. Pour valider cette approche, et à des fins d'expérimentations un nouvel algorithme plus performant que la méthode naïve a été proposé. Comme première étape, il résume les données précédemment explorées. De plus, et comme la réduction du nombre des comparaisons est cruciale, une autre optimisation liée à la réduction de la complexité de la fonction de dominance est introduite.

Acknowledgements

A special thank goes to my supervisors Professor Allel HADJALI and Professor Boutheina BEN YAGHLANE for introducing me to this interesting research domain and for giving me the opportunity to collaborate with other researchers. Thank you for all their advices and guidance during this work.

I would also like to acknowledge with much appreciation my co-supervisor Doctor Mohamed Anis BACH TOBJI for his crucial contributions during this work. I would like to express my gratitude to the support, advice and encouragement he gave to me. I have learned much from his guidance and corrections during my researches.

I would like to thank also the rest of my thesis committee: Prof. Djamal BENSLIMANE, Doctor Hajer BAAZAOUI, Prof. Juliette Débie Barthélemie, and Prof. Salah BEN ABDALLAH, for their insightful comments and encouragement, but also for the useful questions which incented me to widen my research from various perspectives.

I am profoundly grateful to Professor Ladjel BELLATRECHE, leader of Data Engineering team, for his encouragement and advises during this project. Furthermore I would also like to acknowledge with much appreciation the crucial role of the members of LIAS and LARODEC laboratories and specially Bénédicte Boinot, Audrey Veron Wiem Hajji, and Isabelle El Khiati whose support and assistance helped me to coordinate this thesis project.

I would have not finished this thesis project without the support of my family especially my parents for their unconditional love. A special thanks goes to my sisters Rawya, Imen Sarah, and my brothers Walid and Bilel for their encouragement and love.

To my friends Fatma, Lahna, Louis, Kods, Warda, Nourhene, Manel and Ravi thank you for being part of my life, thank you for all the precious moments we shared together. Your friendship is a major source of support when things would get a bit discouraging.

****

I dedicate this thesis to my parents: my father Anwer and my mother Hanifa for their endless love and encouragement, my journey would never be the same without their support!

****

Education is the most powerfull weapon we can use to change the world

---

Nelson Mandela

The process of scientific discovery is, in effect, a continual flight from wonder

---

Albert Einstein

Table of contents

List of figures **xix**

List of tables **xxi**

Nomenclature **xxiii**

Introduction **1**

Part I Preleminaries **7**

Chapter 1 RDF Formalism **9**

Introduction 10

1.1 Semantic Web vision 10

1.1.1 Introduction to Ontologies 11

1.1.1.1 Definition of ontology 11

1.1.1.2 Example of ontologies 12

1.1.2 Ontology languages 12

1.2 RDF data model 13

1.2.1 RDF triple 13

1.2.2 RDF Graph 14

1.2.3 RDF: XML-based syntax 15

1.2.4 RDF databases 16

1.2.4.1 Non-native RDF Databases 16

1.2.4.2 Triple Table 16

1.2.4.3 Property triple table store 17
1.2.4.4 Horizontal table store 19
1.2.4.5 Native RDF Databases 19
1.3 SPARQL Specifications 20
1.3.1 SPARQL General Form 20
1.3.1.1 Triple pattern 21
1.3.1.2 Graph pattern 21
1.3.2 SELECT query Form 22
1.3.3 Basic Graph Pattern 25
1.3.4 Solution Mapping 25
1.3.5 SPARQL Algebra 26
1.3.5.1 Filter 26
1.3.5.2 Join 27
1.3.5.3 LeftJoin 27
1.3.5.4 Union 27
1.3.5.5 OrderBy 28
1.3.5.6 Project 29
1.4 Extended RDF Formalism 30
1.4.1 Trust RDF data 30
1.4.2 Probabilistic RDF data 32
1.4.3 Other uncertain RDF models 33
Conclusion 35

Chapter 2 Background on Possibility Theory and Skyline Queries 37

Introduction 38
2.1 Possibility theory: An overview 38
2.1.1 Typology of imperfect information 38
2.1.1.1 Imprecision 39
2.1.1.2 Inconsistency 39
2.1.1.3 Uncertainty 39

2.1.2 Uncertainty theories: A refresher . . . 40
2.1.2.1 Probability theory . . . 40
2.1.2.2 Evidence theory . . . 41
2.1.3 Possibility Theory . . . 42
2.1.3.1 Possibility distribution . . . 42
2.1.3.2 Possibility and Necessity measures . . . 43
2.1.4 Possibility theory vs Probability theory and Evidence theory . . . 44
2.1.4.1 Possibility theory vs belief function theory . . . 46
2.1.5 Possibilistic Databases . . . 46
2.2 Skyline queries . . . 47
2.2.1 Principle . . . 47
2.2.2 Skyline Computation Algorithms . . . 49
2.2.3 Skyline Queries over Incomplete Data . . . 52
2.2.3.1 Probabilistic Skyline Queries . . . 52
2.2.3.2 Possiblistic Skyline Queries . . . 55
2.2.3.3 Stochastic Skyline Queries . . . 56
2.2.3.4 Evidential Skyline Queries . . . 57
Conclusion . . . 59

Part II Contributions **61**

Chapter 3 Trust Skyline Model: Semantics and Experimentations **63**

Introduction . . . 65
3.1 Trust-Skyline model . . . 66
3.1.1 Trust Dominance . . . 67
3.1.2 Trust-Skyline semantics . . . 71
3.1.3 Trust-Skyline computation . . . 72
3.1.3.1 SQL-like Trust-Skyline queries . . . 73
3.1.3.2 Naive T-Skyline Algorithm . . . 74

3.1.3.3 TRDF-Skyline Algorithm 75
3.2 Experimental Evaluation 76
3.2.1 Experimental Setup 77
3.2.2 Impact of the trust measure variation 77
3.2.3 Impact of the size of data set 79
3.2.4 Number of used properties in the skyline query 79
3.3 Statistical methods-driven Analysis 79
3.3.1 Trust-Skyline list Analysis 81
3.3.1.1 T-Skyline points with less trust 82
3.3.1.2 T-Skyline points with more trust 82
3.3.1.3 Behavior of Alpha 82
3.3.2 Alpha v.s. the distribution of Trust values 83
3.3.3 Central Tendency measures 84
3.3.4 Measures of spread: Quartile measure 85
3.3.5 Trust dependence 85
3.4 Analysis Experimental 86
3.4.1 Experimental Setup 86
3.4.2 Impact of trust threshold variation and Central Tendency measures 87
3.4.3 Impact of trust threshold variation and Quartile measures . . . 88
Conclusion 89

Chapter 4 Possibilistic RDF Data **91**

Introduction 92
4.1 Possibilistic RDF model 93
4.1.1 Possibilistic RDF databases 93
4.1.2 Possibilistic RDF graph 93
4.2 A SPARQL-like language for possibilistic RDF data 96
4.2.1 Possibility-aware Basic Graph Pattern Matching 97
4.2.2 Enhanced SPARQL algebra 99

4.2.2.1 Join(Θ) . . . 99
4.2.2.2 Project(Π) . . . 101
4.2.2.3 Filter operator . . . 101
4.2.3 SPARQL Extension for Possibility distributions Requirements . 102
4.2.3.1 Converting Graph Patterns . . . 102
4.2.3.2 Pi-SPARQL Algebra: Project Possibility Operator . . 103
4.3 Possibilistic Skyline over RDF data . . . 105
4.3.1 Comparison of two possibility distributions . . . 105
4.3.2 Possibilistic dominance on RDF data . . . 107
4.3.3 Possibilistic skyline on RDF data . . . 110
4.4 Possibilistic skyline computation . . . 110
4.4.1 Experimental Evaluation . . . 111
4.4.2 Experimental Setup . . . 113
4.4.3 Size of the Skyline on RDF Data . . . 113
4.4.4 Performance and Scalability . . . 114
Conclusion . . . 117

Conclusion **119**

Appendices **123**
Appendix . . . 122

Bibliography **129**

List of figures

1.1 RDF Graph representation. 14

1.2 Example of RDF graph model. 17

1.3 Relational Representation of Triple RDF Stores. 18

1.4 Relational Representation of Property Tables RDF Stores. 18

1.5 Relational representation of Vertical Partitioning Table. 19

1.6 SPARQL graph pattern. 22

1.7 Trust values' meaning. 31

1.8 Example trust weighted RDF graph. 32

1.9 SPARQL query on a probabilistic RDF database. 34

2.1 Uncertainty weighted models 40

2.2 Skyline of Hotels 49

2.3 An uncertain objects set. 53

3.1 Effect of α on skyline computation 78

3.2 Effect of data size on skyline computation 80

3.3 Effect of criteria number on skyline computation 81

3.4 Trust-Skyline points analysis. 83

3.5 Alpha and Central Tendency measures 88

3.6 Alpha and Quartile measure 89

4.1 Graph representation of uncertain RDF data. 95

4.2 *Pi*-SPARQL representation of query Q2 96

4.3 Basic Graph pattern G 97

4.4 Join operator(Θ) 100

4.5 Project possibility operator. 104

4.6 Size of the skyline on RDF Data modeled by possibility theory. 115

4.7 Elapsed time to compute the skyline on the RDF data modeled by possibility theory. 116

List of tables

1.1 SPARQL query result. 25

2.1 Possibility measure properties 43

2.2 Possibility and probability values associated with X. 45

2.3 A possibilistic relation *pr*. 46

2.4 Example of hotels properties. 49

2.5 Comparison of skyline computation algorithms 52

2.6 Imperfect skyline models 52

2.7 dominance beliefs 58

2.8 dominance beliefs 58

3.1 Example of trust RDF data 68

3.2 Example of hotels properties. 69

3.3 Dominance degree function. 70

3.4 Example of hotels candidate list of T-Sky. 72

3.5 Used functions Meaning. 73

3.6 Parameters under investigation. 77

3.7 Trust frequency distribution. 85

3.8 Example of trust distribution. 86

3.9 Parameters under investigation. 87

4.1 Example of possibilistic RDF database 94

4.2 The results of evaluating the BGP G against the query Q_2. 97

4.3 The results of evaluating the BGP G against the RDF graph 98

4.4 Comparison of two possibility distributions 106

4.5 RDF data modeled by possibility theory. 107

4.6 RDF Data: hotel properties 108

4.7 Parameters and Examined Values. 113

Nomenclature

Acronyms

$\tilde{G}^P$ A possibilistic graph

$card_D(\tilde{\mu^p})$ The cardinality of $\tilde{\mu^p}$ in a multiset D of possibilistic solution mappings

$d(Q \succ P)$ The trust dominance function between Q and P

exp An expression

$h_i.a_k^+$ The maximum value among all properties of a subject $h_i.a_k$

$h_i.a_k^-$ The minimum value among all properties of a subject $h_i.a_k$

$less(,)$ A function that returns the least trust degree between two points

$lessorequal(,)$ A function that returns the least trust degree between two points

p-Sky$_{\mathcal{H}}$ The possibilistic Skyline operator of a data set $\mathcal{H}$

$p - dominance$ The possibility dominance function

$P.t^-$ The minimum trust degree among all properties of a given point

PA The project possibility operator

$Proj^{\pi}()$ The Project possibilistic operator

$SPOT$ RDF quadruple $< s, p, o, t >$

$T - Sky^{\alpha}$ The Trust-Skyline operator

$T - Sky_{<\alpha}$ The T-Sky of points having trust degree less than α

$T - Sky_{>\alpha}$ The T-Sky points having trust degree greater than α

X^* RDF triple $< s, p, o >$

Trust(a ϕ b) The Trust degree of the comparison between a and b

Latin letters

α	A trust measure
λ	An arbiterary value
μ	A mapping function
π	Possibility measure
σ	A multiset of solution mappings
$\tilde{\mu^p}$	possibilistic solution mapping

Introduction

Context and Motivaton

The amount of data in the Web is growing more and more; therefore retrieving relevant and useful information becomes a hard task. Over traditional Web, the pieces of information are expressed in natural language and so understood only by humans. The semantic Web brings structure to the meaningful content of the Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users [Berners-Lee et al., 2001, Mark, 2009]. Hence, resources among the Web became marked with labels that describe their various parts; those labels are called the metadata. To achieve semantic Web, the W3C introduced Resource Description Framework (RDF) as a standard for representing metadata.
The large adoption of Semantic Web in research and industry developed the amount of RDF data on the Web, with a huge increase in diversity and size. However, variety of sources affects the reliability of collected data. Given that an RDF triple is a tuple of $< Subject, Predicate, Object >$ also denoted $< s, p, o >$. To control information trustworthiness, new metrics were introduced in RDF representation to express the intention of information provider about the information veracity issue [Hartig, 2009a, Tomaszuk et al., 2012, Fionda and Greco, 2015]. To manage information in presence of trust, we need then new methods to query RDF data. The last two decades have witnessed a profusion of research effort on supporting complex decision making over uncertain RDF data, such in [Huang and Liu, 2009, Lian and Chen, 2011]. Our ultimate aim in this thesis is to deal with uncertain RDF data in the possibility theory setting [Zadeh, 1978], that represents a non-classical theory of uncertainty. It constitutes an alternative to capture different kinds of imperfection, such as imprecision, total ignorance, and partial ignorance that are not faithfully represented in probability theory [Zimmermann, 1985].

On the other hand, we are interested in preference-based queries in order to extract data according to users' preference and then to reduce the massive amount of information.

Among all preference models returned to the users, skyline model, defined using Pareto dominance [Börzsönyi et al., 2001a], have been the most extensively studied [Jiang et al., 2012, Zhang et al., 2013, Chomicki et al., 2013] and extended over Graph Data such as in [Zou et al., 2010, Zheng et al., 2014]. The skyline operator aims to make multi-objective decisions over multi-dimensional data when different, and often contradictory criteria are to be taken into account. Given such a multi-criteria preference, the system should be able to identify all potentially interesting data records according to user preferences [Chomicki, 2002, Chomicki, 2011].

Thesis contribution

Our main contributions in this work are:

1. Extending Skyline operator over Trust RDF data: We are interested in querying trust RDF data (T-RDF) [Hartig, 2009a]. We particularly tackle the skyline computing problem, which consists in extracting the most interesting trusted resources according to user-defined criteria. To this end, we need to redefine the Pareto dominance relationship in the context of trust RDF data. While this operator produces a binary result in case of certain data, in the context of trust RDF data, it produces a weighted set of results (i.e., each result is associated with a degree of dominance) rather than a boolean (true/false) result. In addition, we need to provide a clear semantics for the trust-skyline, i.e., the set of resources that are dominated by no other resource with a degree exceeding a user-defined threshold (denoted α). A great effort for providing efficient methods to compute the trust-skylinehas been made as well.
2. Possibilistic RDF data: We propose to use possibility theory to express uncertainty on RDF data. We introduce a model for representing and managing possibilistic RDF data. Thus, we integrate in the structure of RDF data a possibility measure for each subject-property-object triple to reflect the user opinion about the truth of a statement. The possibility measure can be considered as a way to express a source reliability.
3. Possibility-aware SPARQL query language: We describe a general framework for supporting SPARQL-like queries on possibilistic RDF data, that we denote Pi-SPARQL. To query possibilistic RDF data, our main contributions in this part are:
   — Introducing the possibilistic solution mappings for a given RDF query Q. In order to define possibility-aware solution mapping, we need to introduce some terminology related to semantics of SPARQL graph pattern expressions.

— Revisiting the conventional SPARQL algebra operators to operate on multisets of possibilistic solution mappings.

4. Extending the skyline operator over possibilistic RDF data: Firstly, we introduce comparison operators between possibility distributions to allow dominance computations in an RDF data set context. Starting from the work of [Chen et al., 2011], we rethought the dominance operator between two possibilistic RDF objects. In addition, we define the possibilistic skyline, denoted p-Sky$_{\mathcal{H}}$, which retrieves the most interesting subjects over the RDF data set $\mathcal{H}$ according to a subset of predicates. The combination aims to filter the massive amount of uncertain resources among the Web according to user preferences.
   During the experimental evaluation step, we introduce two algorithms: A naive one and an optimized one to compute the skyline queries over RDF data. The second algorithm allows improving the performance of the naive one by summarizing the region of data explored in earlier iterations.

Manuscript guide

The remainder of this thesis is organized in four chapters as follows:
In Chapter 1, we describe the basic notions related to RDF formalism. First, we introduce the RDF data model, then we present the SPARQL query language developed to query RDF data. Finally, as RDF data is often pervaded with uncertainty (due to the openness of the Web and variety of sources). We, tackled thus the different extended forms of uncertain RDF data.

In Chapter 2 we present the possibility theory, which is a non-Bayesian theory of uncertainty. It constitutes the model that we use to model uncertain RDF data. Furthermore, we discuss the preference-based queries, especifically, the skyline preference relations that show encouraging results to personalize and filter the massive amount of information residing in today's databases and Information Systems according to the users' preferences.

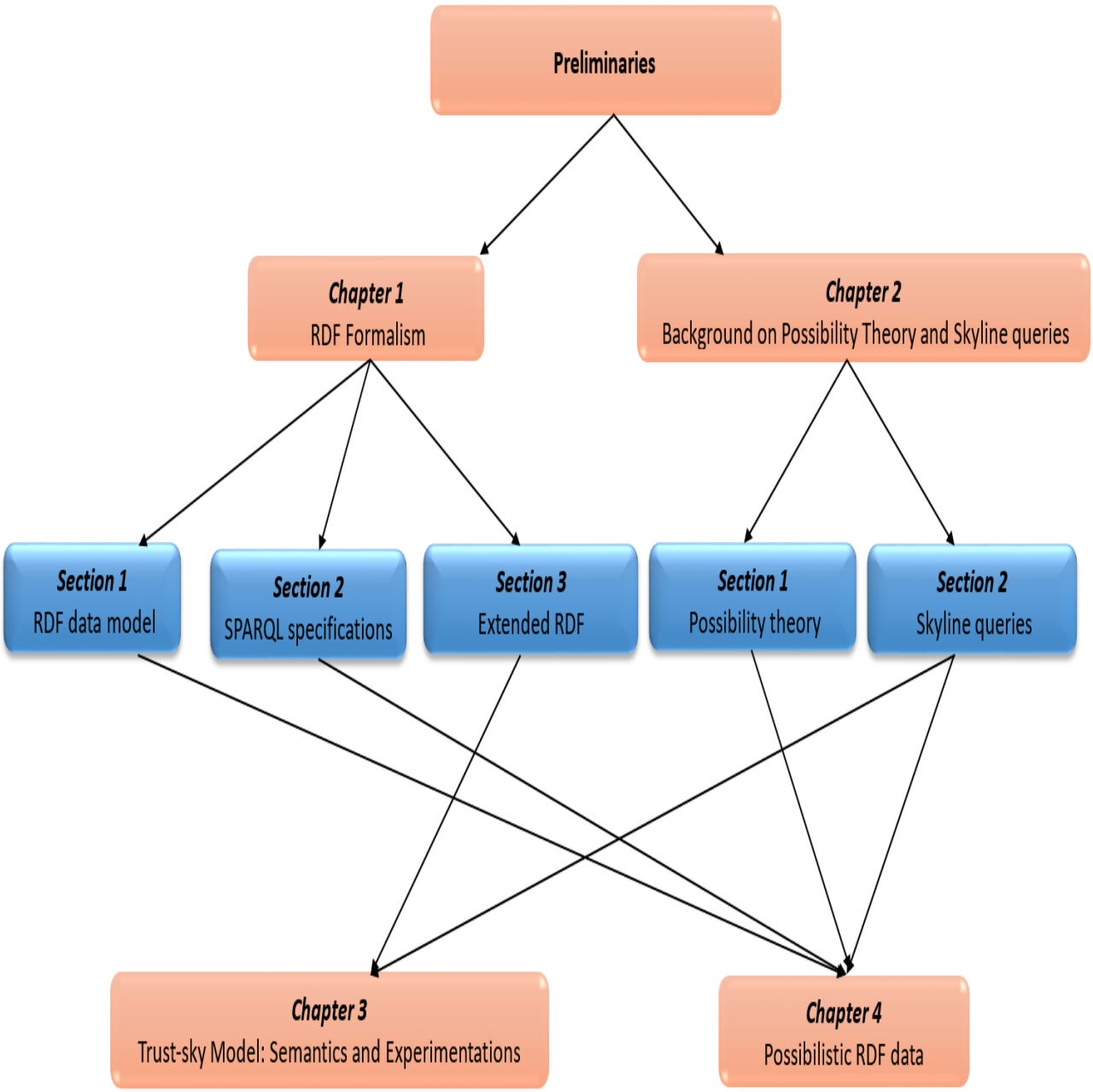
The contribution part starts by Chapter 3 which extend the skyline operator over trust-weighted RDF data (T-RDF). We subdivise this contribution in two main parts. First of all, we define the dominance over Trust-RDF data. Then, we provide a semantics for the trust-skyline, i.e., the set of resources that are dominated by no other resource with a certain degree of trust denoted α. We propose different methods for Trust-skyline computation, then describe the experiments that show interesting results as well.

Second, we analyze the results of the Trust-skyline list. To analyze the results we opt for using statistical methods to investigate the trust measures dependence. We check the impact of the trust measure α and the list of the generated trust measures on the resulting trust-Skyline list. The experiments show interesting results.

In Chapter 4, we propose to model uncertain and imprecise RDF data through possibility theory. Indeed, we add a possibility measure to each RDF triple (association between a subject, predicate and object) to model the possibility of such association. We introduce some comparison operators between possibility distributions to allow dominance computations in an RDF data set context. Second, we introduce a possibility-aware SPARQL query language for supporting SPARQL-like queries on possibilistic RDF data. Finally, we extend skyline queries over possibilistic RDF data by revisiting the dominance operator between two possibilistic RDF data.

Reading Guide

We provide a reading guide that summarises the thesis outline by listing the different chapters and their interactions. We illustrate also the prerequisites for the reading of each chapter.


Preliminaries
Chapter 1
RDF Formalism
Chapter 2
Background on Possibility Theory and Skyline queries
Section 1
RDF data model
Section 2
SPARQL specifications
Section 3
Extended RDF
Section 1
Possibility theory
Section 2
Skyline queries
Chapter 3
Trust-sky Model: Semantics and Experimentations
Chapter 4
Possibilistic RDF data

Part I

Preleminaries

Chapter 1

RDF Formalism

Contents

Introduction **10**

1.1 Semantic Web vision **10**

1.1.1 Introduction to Ontologies 11

1.1.2 Ontology languages 12

1.2 RDF data model **13**

1.2.1 RDF triple 13

1.2.2 RDF Graph 14

1.2.3 RDF: XML-based syntax 15

1.2.4 RDF databases 16

1.3 SPARQL Specifications **20**

1.3.1 SPARQL General Form 20

1.3.2 SELECT query Form 22

1.3.3 Basic Graph Pattern 25

1.3.4 Solution Mapping 25

1.3.5 SPARQL Algebra 26

1.4 Extended RDF Formalism **30**

1.4.1 Trust RDF data 30

1.4.2 Probabilistic RDF data 32

1.4.3 Other uncertain RDF models 33

Conclusion **35**

Introduction

The semantic Web is built on World Wide Web Consortium (W3C) standards. It was an enormous undertaking to build the concept of linked data where comes best practice for sharing data. There was a need to establish standards that make possible to change the Web to a semantic Web by allowing users to add descriptive tags or metadata to the Web content.
Therefore, many standards were created such as the Resource Description Framework (RDF) model, the RDF Schema and the Web Ontology Language (OWL) standards for storing vocabularies and ontologies.During this chapter we pinpoint the Semantic Web concepts specially the RDF data.
Note, the large adoption of Semantic Web in research and industry has led to the development of a large amount of Resource Description Framework (RDF) data on the Web [Berners-Lee et al., 1998a, Mark, 2009, Antoniou and vanHarmelen, 2004a]. However, due to the openness of the web and variety of sources in internet, the reliability of collected data is questioned. To model information trustworthiness, new metrics were introduced in RDF representation to express intention of information provider about information trust [Hartig, 2009a, Tomaszuk et al., 2012, Fionda and Greco, 2015]. Other works are proposed to handle uncertain RDF data, mostly modelled with probability theory such as [Huang and Liu, 2009, Meiser et al., 2011, Lian and Chen, 2011, Meiser et al., 2011].
During this first chapter, we present the basic concepts related to RDF formalism. We start with presenting the semantic Web vision in Section 1.1, Section 1.2 introduces RDF data model, Section 1.3 presents SPARQL query language and Section 1.4, tackled the different extended forms of uncertain RDF data. Finally, we end with a conclusion.

1.1 Semantic Web vision

The Semantic Web is a set of data that is processable by machine, as defined by Berners-Lee et al. [Börzsönyi et al., 2001a]. Changing the Web from a place where different types of information are merely displayed, to a place for interpretation and exchange of data, where agents (which are not compatible between each other) could exchange data in different types. Hence the web changes from a multiple databases and resources to a gigantic database, where resources are linked between each other and those links are understood by computers to provide a more useful and relevant content for users.
As mentioned above, semantic Web aims to structure web resources, hence the produced information becomes not only for human consumption but also for machine "understanding" of the data that it is queried. Authors in [Mark, 2009] show that the use of semantic

Web has two motivators; the first is data integration where a specific mapping is made between the data models or schemas of the data sources involved. Hence describing the data sources' semantics in a machine-interpretable way facilitates the mapping that could be at least semi-automatically. The second motivator, is having a more intelligent support for the internet users. If a computer can process the semantic of information on the web, it can give better results for user queries, by having a better selection, personalization and combination of information from different resources.
The Semantic Web vision is a network of information sources with rich metadata. It provides truly Web-scale integration of many information sources aided by automated algorithms for search and discovery. We could not have a semantic Web without metadata, but metadata alone will not suffice. The metadata in Web pages will have to be linked to special documents that define metadata terms and the relationships between the terms. These sets of shared concepts and their interconnections are called "ontologies." The authors in [Antoniou and vanHarmelen, 2004a] conclude that, in order to achieve the semantic web there is a need to provide three main goals: The first one is vocabularies for expressing the metadata (Ontologies), the second is syntax for representing metadata (RDF and OWL) and third having metadata for lot of Web pages.

1.1.1 Introduction to Ontologies

Ontology is a term which originates from philosophy; it is used as the name of subfield of philosophy. Recently, this term was hijacked by computer science and it changes to a technical meaning different from the origin. Indeed, ontology was introduced to Artificial Ontelligence (AI) by John McCarthy (about 1980) and then developed in other works. Actually, ontology is used in lot of domains such Samantic Web, Data Mining, natural language processing, etc.

1.1.1.1 Definition of ontology

Ontology is defined by [Gruber, 1993] as an explicit specification of conceptualization. Hence, an ontology consists of a list of terms and relationships between these terms. The terms are denoted by concepts which are classes and objects of the domain. Hence, there is a list of classes and subclasses defined in a hierarchy, a part from the subclass relationship contains the properties, the specification of logical relationships between objects. In general, ontology provides shared understanding of a specific domain to overcome differences in terminology (more details in [Antoniou and vanHarmelen, 2004a]).

1.1.1.2 Example of ontologies

We illustrate some well-known examples of ontologies in the domain of Semantic Web. For instance, DBpedia [Jens et al., 2015] is generated automatically by extracting data from Wikipedia. Also the ontology Yago [Suchanek et al., 2008], which is created by extracting data from Wikipedia and Wordnet.
For experiments, ontologies are also used to define benchmarks, such the LUBM benchmark [Guo et al., 2005]. This benchmark define an OWL ontology about the university domain, customizable and repeatable synthetic data, a set of test queries, and several performance metrics.

1.1.2 Ontology languages

As described by [Antoniou and vanHarmelen, 2004a] the most important ontology languages for the web are:

- **XML**: Extensible Markup Language (XML), created by the W3C and it is playing an important role in the exchange of a wide variety of data on the Web and elsewhere. Although, it provides a surface syntax for structured documents, but there still no semantic constraints on the meaning of the XML documents
- **XML Schema**: A language that restricted the structure of XML documents, in more specific way. It provides a means for defining the structure, content and semantics of XML documents.
- **RDF**: A data model for resources or objects and relations and links between them. Otherwise, it provides a semantic for the data model which could be represented in XML syntax (see Section 1.2 for more details).
- **RDF Schema**: A vocabulary description language for describing properties and classes of RDF resources. It implicates generalization hierarchies of properties and classes of those resources.
- **OWL**: Web Ontology Language (OWL) is a richer vocabulary description language for describing properties and classes such as relations between classes and richer typing and characteristics of properties.

The reason behind creating the semantic Web is to bring structure to the meaningful content of Web pages to facilitate and automate its access. To this end, resources on the Web are marked with labels that describe their structure; these labels are called *metadata*. The aim is to make content not only readable by humans, but also by machines, making it possible to analyse and manage huge volumes of Web data. Therefore, the

W3C community introduced a recommendation for semantic annotations; the Resource Description Framework (RDF). In the coming sections, we will focus our interest on the model used in this thesis project, the RDF one. We are specifically interested in managing uncertain RDF data that are used as an input for our data analysis to come up with new concepts.

1.2 RDF data model

Resource Description Framework (RDF) is a *W3C* framework for representing meta-data and describing the semantics of information in a machine-accessible way [W3C, 2004]. An RDF statement is a triple $< s, p, o >$ where s, p, and o stand for *subject*, *predicate* and *object* respectively.

1.2.1 RDF triple

RDF describes Web resources (subject) related/characterized (via a predicate or a property) to other resources/literals (object).

Definition 1.1. *Assume we have a finite set of RDF URI references (U); a finite set of Blank nodes (B); and a finite set of RDF Literals (L). A triple* $< Subject, Predicate, Object >$ *or* $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ *is called an* ***RDF triple****.*

An RDF triple allows defining statements in form of a triple Subject-Predicate-Object:

— A subject is a resource and could be a Uniform Resource Identifier (URI) or a Blank Node (BN). The resource is a thing, a person, a Web page. The aim from using URI is to provide a universally unique name for a resource or property. Thus, it's possible to link data from different sources around the world.
We illustrate the example of LUBM ontology, the resource FullProfessor0 has a URI, *http://www.Department0.University0.edu/FullProfessor0*. The book of [DuCharme, 2011a] is an excellent bibliographic reference for the whole specifications.

— A Predicate also called property is a relationship between a subject and an object. It is an RDF URI reference. An example from the LUBM Benchmark, the resource FullProfessor has as property teacherOf which is identified with the URI, *http://www.Department0.University0.edu/Course*.

— An Object is the value of a subject property. It is a URI reference, a literal or a blank node. the literals are used to identify values such as strings, numbers and

dates by means of a lexical representation. A literal could be the object of an RDF statement, but not the subject or the predicate. A Blank Node also called BNode is drawn from an infinite set, which is the set of all the RDF URI references and the set of all literals are pairwise disjoint. BNode is a node in an RDF graph representing a resource for which a URI or literal in not given. The resource represented by a blank node is also called an anonymous resource.

1.2.2 RDF Graph

A set of RDF triples is a graph for representing meta-data and describing the semantics of information in a machine-accessible way. Therefore, RDF data can be thought in terms of a decentralized directed labelled graph. The edges' labels are the "properties", also called "predicates" or "attributes". The RDF data is stored as a set of SubjectNode-PropertyArc-ObjectNode triples often called $< s, p, o >$ triples, and represented graphically as shown in Figure 1.1.

Example 1. *We present an example of RDF triple data (written in XML language) illustrating information about hotel* $H1$ *(Extracted from source http://www.hotel.org/H1). The hotel with ID* $H1$ *is "Philippos", the street address is "Solonos 9 & Dimitriados Str". The Figure 1.1 is the graph representation of this example:* <rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
<rdf:Description rdf:about="http://www.hotel.org/H1">
<hasName> Philippos </hasName>
<hasAdress>Solonos 9 & Dimitriados Str</hasAdress>
</rdf:Description>
</rdf:RDF>

Figure 1.1 – RDF Graph representation.

1.2.3 RDF: XML-based syntax

RDF is a data model it has several formats i.e several ways to represent data in the form of triples such as:

— Turtle family of RDF languages (N-Triples, Turtle, TriG and N-Quads);

— JSON-LD (JSON-based RDF syntax);

— RDFa (for HTML and XML embedding);

— RDF/XML (XML syntax for RDF).

In this section, we present XML which is one of the syntax representations of the RDF data model. Meanwhile, in XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications. XML Namespaces provide a method to avoid element name conflicts. The namespace can be defined by an xmlns attribute in the start tag of an element. The namespace declaration has the following syntax.

xmlns:prefix="Location".

Example 2. *The location is the DTD or schema*
<! Doctype owl [<! ENTITY xsd http://www.w3.org/2001/XMLSchema# >]>.
< rdf : RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:uni="http://www.mydomain.org/uni-ns#" >
< rdf: Description rdf: about="949318">
<uni : name>Daniel Alberto</uni: name>
<uni : title>Professor</uni: title>
</ rdf: Description>
< rdf: Description rdf: about="949333">
<uni: courseName>Data Base</uni: courseName>
<uni: isToughtBy> Daniel Alberto </uni: isToughtBy>
</ rdf: Description>
</rdf:RDF>

The two elements **uni:name** and **uni:title** both define property-value pair for 949333, the content of rdf: Description elements are called property elements. As shown in the last example the two elements courseName and isToughtBy, both of them define the property-value pairs for 949333. The description of a document could be in many categories, it could be clear for human readers, but not for machine. In XML the fact is not formally

cleared anywhere. In, RDF it is possible to add such statement by using the rdf: type element, that could be in our example a course type, we add to the syntax <rdf: type rdf: resource="&uni;course"/> to specify that the resource is a course type not lecturers for example. There exists much more specification about XML syntax that was added to fit with the RDF data, we did not illustrate all of them because it is not the aim of this dissertation.

1.2.4 RDF databases

Within semantic Web applications, storing and querying RDF data is one of the basic tasks. Therefore, many storage approaches have been proposed, some of them are native approaches and others are not. This classification is not the most important criteria for RDF data storage, since some non-native techniques performed well and are the most used ones for storing and querying RDF data.

Example 3. *We illustrate an example of RDF graph model about hotel's properties as shown in Figure 1.2. The hotel with id $H1$ has a price value equal to 5 (per night), a distance (from the beach) equal to 3, a name value "Philippos", etc. We use this example to present the different storage approaches of RDF data.*

1.2.4.1 Non-native RDF Databases

The non-native approaches are a set of techniques proposed for storing RDF data in RDBMS. Currently, this is widely considered to be the best performing approach for their persistent data store due to the great amount of work achieved on making it efficient, extremely scalable and robust [Sakr and Al-Naymat, 2009].

1.2.4.2 Triple Table

Each RDF statement of the form (Subject, Property, Object) is stored as a triple in one large table with a three-columns schema . The storage system of 3Store is based on a central triple table which holds the hashes for the subject, predicate, object and graph identifier [Sakr and Al-Naymat, 2009] [Harris and Gibbins, 2003a]. Indexes are then added for each of the columns in order to make joins less expensive. However, since the collection of triples is stored in one single RDF table, the queries may be very slow to execute. Indeed, when the number of triples scales, the RDF table may exceed main memory size. Therefore [Neumann and Weikum, 2008] proposed a work over that field,

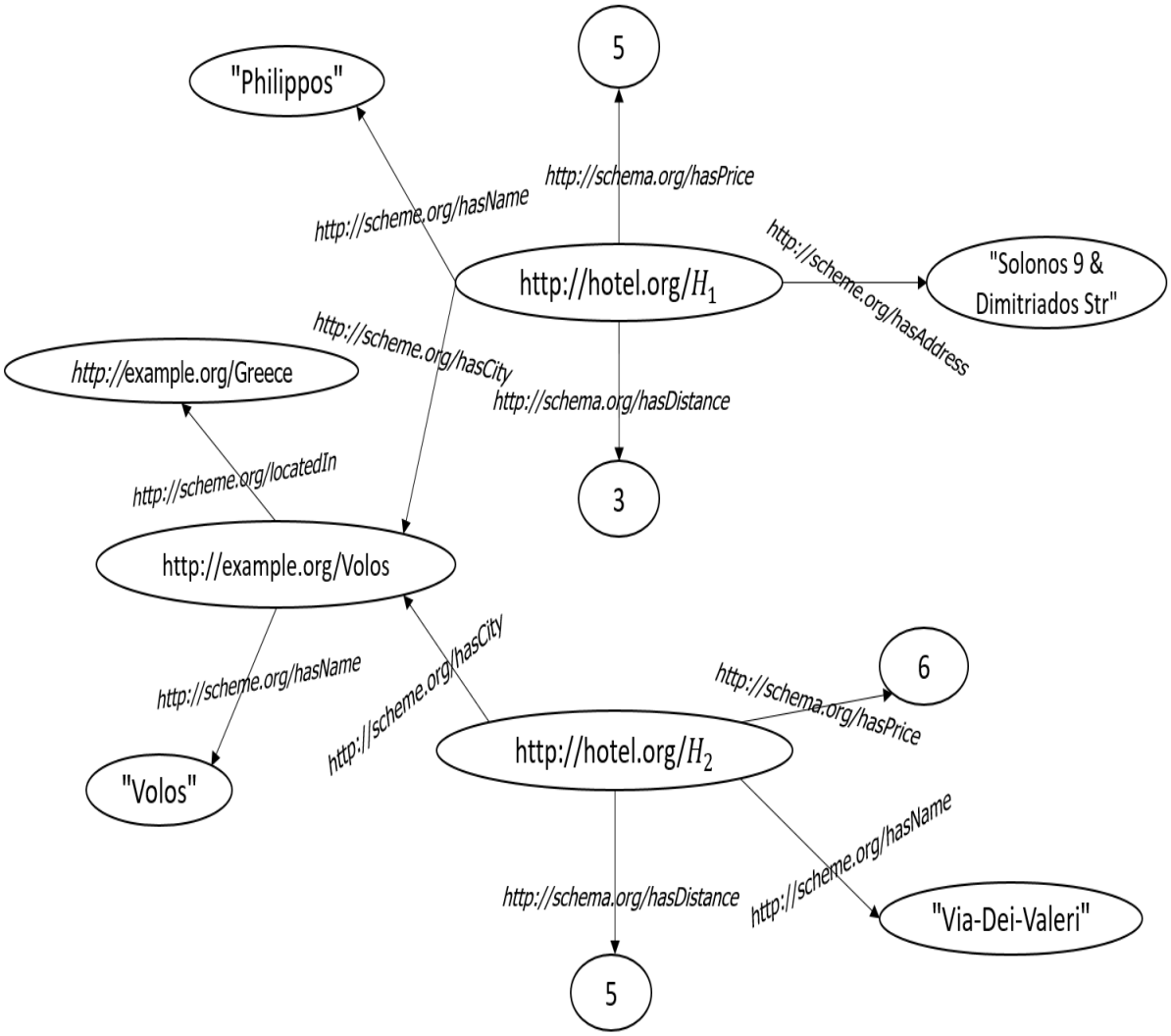


FIGURE 1.2 – Example of RDF graph model.

they have presented the RDF-3X (RDF Triple eXpress), an RDF query engine which tries to overcome the criticism that triples stores incurs too many expensive self-joins by creating the exhaustive set of indexes and relying on fast processing of merge joins. The Figure 1.3 shows the triple table store of the RDF graph presented previously in Figure 1.2.

1.2.4.3 Property triple table store

Authors in [McBride, 2002] presented Jena as an open source toolkit for Semantic Web programmers. It implements persistence for RDF graphs using a SQL database through a JDBC connection. The schema of the first version of Jena, Jena1, consisted of a statement

Subject	Predicate	Object
H1	hasName	Philippos
H1	hasPrice	5
H1	hasDistance	3
H1	hasAddress	Solonos 9 & Dimitriados Str
H1	hasCity	http://example.org/Volos
http://example.org/Volos	hasName	Volos
http://example.org/Volos	locatedIn	*http://example.org/Greece*

FIGURE 1.3 – Relational Representation of Triple RDF Stores.

table, a literals table and a resources table. Jena2 uses property table as a general facility for clustering properties that are commonly accessed together. During their work [Sakr and Al-Naymat, 2009] consider that a property table is a separated table that stores the subject value pairs related by a particular property.
The property table technique was proposed to improove RDF data organization by allowing multiple triple patterns referencing the same subject to be retrieved without an expensive join.
A variant of the property table, named property-class table, uses the "rdf:type" property of subjects to cluster similar sets of subjects together in the same table. The Figure 1.4 shows the RDF graph model stored over a property table, in which multiple triple patterns refer the same subject.

Hotel

ID	hasName	hasPrice	hasDistance	hasAddress	hasCity
H1	Philippos	5	3	Solonos 9 & Dimitriados Str	http://example.org/Volos

City

ID	hasName	locatedIn
http://example.org/Volos	Volos	http://example.org/Greece

FIGURE 1.4 – Relational Representation of Property Tables RDF Stores.

1.2.4.4 Horizontal table store

Among the horizontal table store, the RDF triples are modeled as one horizontal table or into a set of vertically partitioned binary tables (one table for each RDF property) [Sakr and Al-Naymat, 2009]. For the vertically partitioned binary table, RDF data are vertically partitioned using a fully decomposed storage model (DSM). Each triple table is divided into n two column tables where n is the number of unique properties in the data as shown in Figure 1.5.
In each of these tables, the first column contains the subject and the second column the object value of that subject. The tables are stored, by using a column-oriented DBMS (DBMS designed especially for the vertically partitioned case, as opposed to a row-oriented DBMS, gaining benefits of compressibility and performance), as collections of columns rather than collections of rows. The goal is to avoid reading entire row into memory from disk, like in row-oriented databases, if only a few attributes are accessed per query.

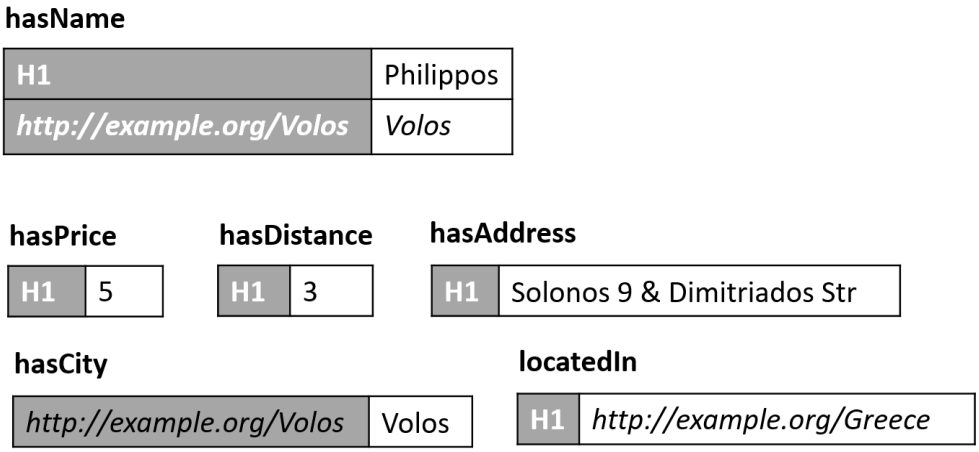


FIGURE 1.5 – Relational representation of Vertical Partitioning Table.

1.2.4.5 Native RDF Databases

The native storage solutions store RDF with respect to the data model, avoiding the mapping to a Database Management System (DBMS). Those techniques aim to be closer to the query model of the semantic Web and are based on the characteristic of multiple indexes of the RDF triples. They can be subdivided in two main categories:

1. In memory storage, where the ontology is stored in central memory starting from the launch of the application;
2. Non-memory storage, in which ontology and data are stored on disk, such that

query processing can be performed more efficiently compared to straight-forward approaches like triple tables.

The data representation in native RDF databases differs from one another. In fact, this representation is specific and relative to each native RDF database. Nevertheless, it is possible to characterize them from the data structure. Indeed, we found an RDF database family that uses indexes and tree-structures to model data such RDF-3X [Neumann and Weikum, 2008] , Hexastore [Weiss et al., 2008], or IBM DB2RDF [Bornea et al., 2013]. We could identify another family wich uses graph to model data such the case of gStore [Özsu, 2016].

To query RDF data, several query languages were proposed. Indeed, they differ in many aspects and can be grouped into several families according to data model, expressivity, support for schema information, etc. Mainly there is six families: *SPARQL, RQL [Karvounarakis et al., 2002], XPath-, XSLT-, and XQuery-based Languages, Metalog, Reactive Languages, and Deductive Languages* [Bailey et al., 2005]. For instance, the SPARQL family consists of the four query languages SquishQL [Miller et al., 2002], RDQL [Seaborne, 2004], SPARQL, and TriQL. The common property to this family is that they consider RDF as triple data without considering the semantics linked to triples.

In our work we are interested in SPARQL. We present in the coming section, more details about SPARQL.

1.3 SPARQL Specifications

SPARQL query language for RDF (SPARQL) is a SQL-like language for querying RDF data. By 2004, the W3C formed the RDF Data Access Working Group (DAWG) that gathered use cases and requirements to make the first draft of SPARQL Query Language specifications. Before that, a dozen query languages had been developed as commercial, academic, and personal projects. Later, the query language, protocol, and query results XML format became Recommendations/official W3C specifications [DuCharme, 2011a].

1.3.1 SPARQL General Form

SPARQL is similar to SQL. It selects data from an RDF data set by using a SELECT statement to determine which subset of the selected data is returned. Also, SPARQL uses a WHERE clause to define graph patterns to find a match in the data set. Hence, a graph pattern in a "SPARQL WHERE clause" consists of the subject, predicate and objects

triple to find a match in the data. Given that SPARQL is a query language developed primarily to query RDF graphs, the vocabulary for RDF graphs is three disjoint sets: a set of URIs *Suri*, a set of bnode identifiers *Sbnode*, and a set of well-formed literals *Slit*. The union of these sets is called the set of RDF terms. Authors in [Evren and Bijan, 2007] defined an RDF triple as a tuple (s, p, o) ∈ (Suri ∪ Sbnode) × Suri × (Suri ∪ Sbnode ∪ Slit). An RDF graph is a finite set of RDF triples.

SPARQL has four query forms which use the solutions from pattern matching to form result sets or RDF graphs. The query forms are:

— SELECT: Returns all, or a subset of, the variables bound in a query pattern match.

— CONSTRUCT: Returns an RDF graph constructed by substituting variables in a set of triple templates.

— ASK: Returns a Boolean indicating whether a query pattern matches or not.

— DESCRIBE: Returns an RDF graph that describes the resources found.

In our thesis project, we are interested in using the SELECT query form. Given that the SELECT query is the most used one to query RDF data [Francois and Stijn, 2011,Saleem et al., 2015].

1.3.1.1 Triple pattern

A triple pattern is the atomic element of a constraint's description of a SPARQL query. Indeed, SPARQL allows selecting values by letting triple patterns (one or more element of the subject, predicate or object) contain values denoted by ? or $ before a string. The aim from using triple patern is finding the corresponding RDF triples, this process is called matching, it produces a binding of each variable.
Let us illustrate the example of the RDF triple pattern (?hotel, hasName, ?name), the matching of that triple pattern to the RDF graph on Figure 1.2 consists on finding the values of ?*hotel* having the requested properties ?name and ?price. The variable ?price takes the value 10, hence, the triple pattern becames (H_1, Distance, 10) that belongs to the graph in Figure 1.2 indeed, forms a solution. The combination of multiple triple patterns makes a graph pattern.

1.3.1.2 Graph pattern

A graph pattern is a set of triple patterns. Triple patterns are linked to eachother with resources in common (variables, literals, etc.) to form the graph pattern. For a graph pattern, the common variables between two or more triple patterns are called join operator

variables.
Querying an RDF graph using a triple pettern means searching values for each variable of the graph pattern. Indeed, the resulting graph (after replacing the variables with the corresponding values) is considered as sub-graph of the RDF graph.
Authors in [Pérez et al., 2009] defined recursively a SPARQL graph pattern expression as follows.

1. A tuple from (U ∪ B) × U × (U ∪ B ∪ L) is a graph pattern (a triple pattern).
2. If P_1 and P_2 are graph patterns, then expressions corresponding to conjunction graph pattern (P_1 AND P_2), optional graph pattern (P_1 OPT P_2), and union graph patternv (P_1 UNION P_2) are graph patterns.
3. If P is a graph pattern and R is a SPARQL built-in condition, then the expression (P FILTER R) is a graph pattern (a filter graph pattern).

Example 4. *We illustrate the example of two triple patterns t_1 and t_2:*
?hotel hasName ?name (t_1)
?hotel hasPrice ?price (t_2)

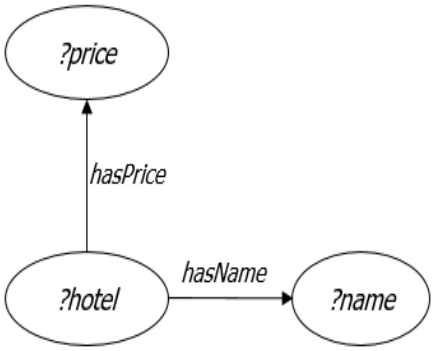


FIGURE 1.6 – SPARQL graph pattern.

The Figure 1.6 is a graph pattern representation, that should correspond to a part of the graph (see Figure 1.2) while executing the query. For instance, H_1 is a join operator variable that links both graph patterns t_1 and t_2.

1.3.2 SELECT query Form

We present the different clauses that define the SELECT query syntax:

— SELECT: This clause let the users specify the result set, it has generally the form of: SELECT ?Result-Set.

— FROM: Allows users to specify the Data set from which the result set will be selected. It should be written in the form of: From <Location>.

— WHERE: Allows users to pinpoint the Query Triple Pattern. it is generally specified as: WHERE { ?Set-conditions }

SPARQL also uses other operators in the SELECT clause such as FILTER, OPTIONAL, Distinct, LIMIT, ORDER BY and GROUP BY.

— UNION: This operator creates multisets of responses from sets of two or more graph patterns. They either have variables in common or they do not.

— FILTER: SPARQL FILTERs restrict the solutions of a graph pattern match according to a given expression. Specifically, FILTERs eliminate any solutions that, when substituted into the expression, either result in an effective boolean value of false or produce an error.

— OPTIONAL: Allows information to be added to the solution where the information is available, but do not reject the solution because some part of the query pattern does not match (the optional part does not match). Hence it creates no bindings but does not eliminate the solution.

The sequence modifiers are applied to create sequence (a solution sequence) with specific orders. This sequence is used to generate one of the results of a SPARQL query form, we illustrate below these sequence modifiers:

— ORDER BY: Allows sorting the fetched data in either ascending or descending according to one or more columns.

— Projection modifier: choose certain variables

— Distinct modifier: ensure solutions in the sequence are unique

— Reduced modifier: permit elimination of some non-distinct solutions

— Offset modifier: control where the solutions start from in the overall sequence of solutions

— Limit modifier: restrict the number of solutions

The syntax of the SELECT query in SPARQL is as follows [Eric and Andy, 2008]:

⟨**SelectQuery**⟩::= SELECT ⟨DISTINCT|REDUCED⟩?⟨Var+⟩| ∗
⟨DatasetClause∗⟩⟨WhereClause⟩⟨SolutionModifier⟩

`SELECT` is the keyworld that identifies the type of the query. The clauses `DISTINCT` and `REDUCE` allow modifying the structure of the answer. The symbol `Var+` denotes the projection of at least one variable as an answer for the query and $*$ for all the variables. The symbol `DatasetClause`$*$ is used to define the set of queried data.
The symbol `WhereClause` represents the set of query constraints. Finally, the symbol `SolutionModifier` denotes a set of modifiers, precisely `ORDER BY` and /or `GROUP BY`.

Example 5. *We illustrate an example of SPARQL query that returns hotels having the minimum values of the properties price and distance(from the beach):*

```
PREFIX p: <http://www.semanticweb.org/ontology-exp#>
SELECT DISTINCT ?name ?distance ?price
WHERE {
    ?hotel a p:Hotel .
    ?hotel p:hasName ?name .
    ?hotel p:hasDistance ?distance .
    ?hotel p:hasPrice ?price .
    OPTIONAL {
        ?H a p:Hotel .
        ?H p:hasDistance ?dist .
        ?H p:hasPrice ?price .
        FILTER (( ?dist < ?distance ) && ( ?pri < ?price )) .
        }
    FILTER ( !bound (?H) ).
}
```

We detail the SPARQL query here. Indeed, the first three triples match any hotel for which a place and a distance are known. The pattern inside the OPTIONAL clause also matches a hotel, and gets it's properties (distance and price). We use the variable name $?H_x$ for this hotel because of the FILTER clause; we retain only the bindings of $?H_x$ that have least values of distance and price than ?Hotel. Despite, what happens if we can't find anyone with less price and distance? Then all matches to the pattern inside the OPTIONAL braces will be filtered out, and no bindings will remain for $?H_x$. Back outside the OPTIONAL, we filter based on the binding of $?H_x$; if $?H_x$ is not bound, then we didn't find any hotel with a less price and distance values.

The result of this query is shown in Table 1.1 below, where we find hotel H_1 associated with its price and distance values. For instance, H_1 has less price and distance value than H_2.

Hotel	Price	Distance
H1	3	5

TABLE 1.1 – SPARQL query result.

1.3.3 Basic Graph Pattern

The core of each SPARQL query is a Basic Graph Pattern (BGP). A BGP is the elementary graph pattern; it is a set of RDF triple patterns that may contain variables at the subject, predicate, and object position. Indeed, a BGP is a graph structured query that should be matched against the graph database G. Intuitively, a match for a BGP is a mapping from variables in the query to constants (graph database). During evaluation, a solution for a BGP is each solution mapping which, in combination with an RDF instance mapping, maps the BGP to a subgraph of the queried RDF graph.

1.3.4 Solution Mapping

A solution mapping is a mapping from a set of variables to a set of RDF terms. The work of [Eric and Andy, 2008] is an excellent bibliographic reference for the whole specifications. A formal definition is given as follows:

Definition 1.2. *A mapping μ from X to Y is a partial function $\mu : X \rightarrow Y$. The domain of μ denoted by $dom(\mu)$ is the subset of X where μ is defined. Let σ be a multiset of solution mappings, $card[\sigma](\mu) = card[\sigma]$ is the number of distinct RDF instance mappings, σ, such that $P = \mu(\sigma)$ is a pattern instance mapping and P(BGP) is a subgraph of G.*

The solution sequence modifiers defined by W3C in [Eric and Andy, 2008] are:

— Order By modifier: The aim from using it is to put the solution in order.

— Distinct modifier: It ensures solutions in the sequence are unique.

— Reduced modifier: It permits non-distinct solutions to be eliminated.

— Offset modifier: It controls where the solutions start from in the whole sequence of solutions.

— Limit modifier: It restricts the number of solutions.

1.3.5 SPARQL Algebra

The outcome of executing a SPARQL is defined by a series of steps as presented in [Eric and Andy, 2008]:

1. Starting from the SPARQL query as a string;
2. Turning that string into an abstract syntax form;
3. Turning the abstract syntax into a SPARQL abstract query comprising operators from the SPARQL algebra;
4. Evaluating this abstract query on an RDF dataset.

Given that a SPARQL abstract query is a tuple (exp, D, Q) where:

— exp is a SPARQL algebra expression

— D is an RDF Dataset

— Q is a query form

For each symbol in a SPARQL abstract query, an operator for evaluation is defined. Besides BGPs, the SPARQL specification [Eric and Andy, 2008] introduces other graph patterns CGPs (Complex Graph Patterns). CGPs extend BGPs with further traditional relational operations (projection, union, difference, optional (left-outer-join) and filter) [Angles et al., 2017]. During query evaluation, CGPs are represented by algebra operators. Those operators operate on multisets of solution mappings. We present below some operators which we redefined in Section 4.2 to consider the possibilistic data model.

1.3.5.1 Filter

The FILTER SPARQL clause restrict the solutions, which correspond to a graph pattern, those who satisfy the specified condition (the filter expression is evaluated to true).

Example 6. *We illustrate an example of a SPARQL query Q_{u1}: "Give the name of hotels with price value less than 30"*

PREFIX ns: <http://hotel.org/example>

SELECT ?name ?price

WHERE {

?x a ns:Hotel .

?x ns:hasPrice ?price .

FILTER (?price < 30) .

?x ns:hasName ?name }

1.3.5.2 Join

The join Θ operator represents a group of graph pattern (minimum two). $\mu_1 \Theta \mu_2$ contains all variable bindings from the two mappings in a new one. Knowing that, the mapping from one operand is combined or merged to every mapping from another if they are compatible. Indeed, $\mu_1 \Theta \mu_2$ is the set of mappings that result from extending mappings in μ_1 with their compatible mappings in μ_2.

1.3.5.3 LeftJoin

The LeftJoin performs a join starting with the left side. Indeed, it executes each operand with the graph to query and performs the leftjoin operation by adding every solution from the left, merging compatible solutions from the right that match an optional filter.

Example 7. *We illustrate an example of LeftJoin query from [Eric and Andy, 2008], group consisting of a basic graph pattern and two optional graph patterns:*

{ ?s :p1 ?v1 OPTIONAL {?s :p2 ?v2 } OPTIONAL { ?s :p3 ?v3 } }

LeftJoin(

LeftJoin(

BGP(?s :p1 ?v1),

BGP(?s :p2 ?v2),

true) ,

BGP(?s :p3 ?v3),

true)

1.3.5.4 Union

The Union operator is useful in case of joining the criteria from two separated WHERE statements or to join data using two different predicates and and there is a need to consider

them as the same.

Example 8. *We illustrate an example of a Union SPARQL query illustrated from [Eric and Andy, 2008], the queried data model is as follows:*
@prefix dc10: <http://purl.org/dc/elements/1.0/> .

@prefix dc11: <http://purl.org/dc/elements/1.1/> .

_:a dc10:title "SPARQL Query Language Tutorial" .

_:a dc10:creator "Alice" .

_:b dc11:title "SPARQL Protocol Tutorial" .

_:b dc11:creator "Bob" .

_:c dc10:title "SPARQL" .

_:c dc11:title "SPARQL (updated)" .

The SPARQL query is as follows:

PREFIX dc10: <http://purl.org/dc/elements/1.0/>

PREFIX dc11: <http://purl.org/dc/elements/1.1/>

SELECT ?title

WHERE { { ?book dc10:title ?title } UNION { ?book dc11:title ?title } }

1.3.5.5 OrderBy

The ORDER BY clause establishes the order of a solution sequence. Following the ORDER BY clause is a sequence of order comparators, composed of an expression and an optional order modifier (either ASC() or DESC()). Each ordering comparator is either ascending (indicated by the ASC() modifier or by no modifier) or descending (indicated by the DESC() modifier).

Example 9. *We illustrate an example of a SPARQL query Q_{u1}: "Give the name of hotels ordered by their price values"*

PREFIX ns: <http://hotel.org/example>

SELECT ?name

WHERE {

?x a ns:Hotel .

?x ns:hasPrice ?price .

ORDER BY ?name ASC (?price) }

1.3.5.6 Project

The project(π) operator allows for selecting a subset of the output variables of a graph pattern as the new output variables. It allows for stating which variables deemed relevant in the evaluation of Complex Graph Pattern (CGPs).
In the coming section, we present the main extended RDF formalism to manage uncertainty in RDF data.

Example 10. *We illustrate an example from [Eric and Andy, 2008] that shows a query to extract the names of people described in an RDF graph using FOAF properties, the queried data model is as follows::*

@prefix foaf:<http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Alice" .

_:a foaf:mbox <mailto:alice@work.example> .

_:b foaf:name "Bob" .

_:b foaf:mbox <mailto:bob@work.example> .

The SPARQL query is as follows:

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name

WHERE

{ ?x foaf:name ?name }

1.4 Extended RDF Formalism

RDF has been used in many real applications. A large amount of RDF data has been published on the Web; the ease to combine it from different resources creates new challenges linked to efficient query answering and RDF data trustworthiness. Therefore, unreliable data could dominate results of queries, affect knowledge bases, and have negative or misleading impact on software agents. In this section, we present the main extended formalism to manage uncertainty residing in RDF data.

1.4.1 Trust RDF data

Uncertainty can result either from inconsistency between sources or from imprecision/inaccuracy of them. Recently, there is a profusion of research works on this topic. There is an urgent need of standardized mechanisms for data trustwothiness evaluation and a uniform way to manage imperfection of the Web of data [Buche et al., 2005]. Authors in [Richardson et al., 2003, Golbeck, 2006] introduced the Web of trust. For instance, in [Richardson et al., 2003], they estimated user believes in statements supplied by any other user and defined properties for combination functions to merge trusts. The work of [Golbeck, 2006] deals with social networks. Indeed, authors presented an approach to integrate trust, provenance and annotations in semantic Web systems. The work of [Golbeck et al., 2003] investigated the applicability of social network analysis to the semantic Web. Authors discussed the multi-dimensional networks evolving from ontological trust specifications.
To rate the trustworthiness of RDF Data, Olaf Hartig in [Hartig, 2009a] advocated the need of a uniform way. He introduced a new model that associates RDF statements with trust values (trust model). Trust-RDF or T-RDF model expresses the trustworthiness or the trust degree associated to the RDF triple as a subject of belief and disbelief in the truth of the information represented by this triple. In fact, belief (disbelief) of an RDF triple is expressed as the degree of confidence in the truth (untruth) of the information (see Figure 1.7). Indeed, the trust measure, denoted t, is either a value in the interval $[-1, 1]$ or unknown. In case, $t = 1$, the user is absolutely sure about the truth of the triple. A positive trust value less than 1 represents belief in the information veracity. Meanwhile, a negative value expresses a disbelief and a value $t = -1$ expresses a certitude in the information untruth. In the last two decades There is a great research effort to develop the trust RDF model such in [Hartig, 2009a, Hartig, 2009b, Tomaszuk et al., 2012, Ceolin et al., 2014, Olaf, 2014, Fionda and Greco, 2015]. The trustworthiness of RDF triples could be represented by a trust degree which is either a value in the interval $[-1, 1]$ or unknown. For a trust value of 1, the user is absolutely sure about the truth of the corresponding

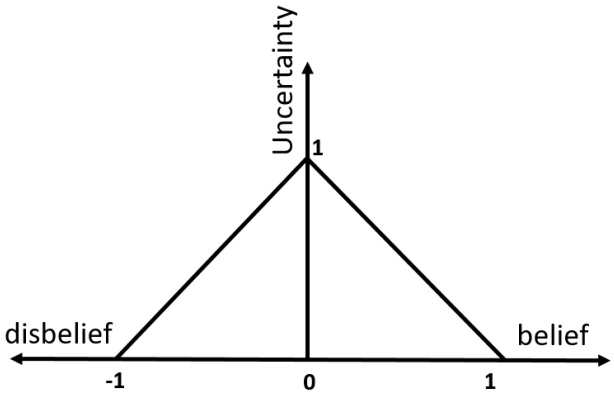


FIGURE 1.7 – Trust values' meaning.

triples; a positive trust value less than 1 represents belief in the truth; meanwhile, to a certain degree the user is not sure regarding the assessment.

In the trust model the RDF triple is extended to a quadruple $< s, p, o, t >$ where the value t expresses the trust degree of the RDF triple $< s, p, o >$. Therefore, authors in [Hartig, 2009a] proposed a trust-aware query language, denoted tSPARQL, which extends SPARQL to describe trust requirements and access the query solutions' trustworthiness. Indeed, tSPARQL let users access trust values which represent the matching subgraphs' trustworthiness.

Example 11. *We illustrate an example from [Hartig, 2009a] in Figure 1.8, which represents a trust-weighted RDF graph of the query "Return the most trustworthy review for each hotel in the city of "Heraklio". For instance, the graph's edges represent the predicates of triples annotated with predicate identifier and with a consumer trust value of the corresponding triple. As an example of nodes, the resource node* ***ex1:Kastro*** *represents a hotel building. This triple is associated with a trust value of* 0.95*. Hence, each triple has a trustworthiness value, in order to compute the trustworthiness of an RDF graph [Hartig, 2009a] defined an aggregation function that computes the trustworthiness of its triples. Therefore, the author introduced a TRUST AS clause into SPARQL query language to express the trust measure ?t. Indeed, the variable ?t allows access to the trust values of the subgraphs that match the given query pattern.*

```
SELECT ?h ?txt1 WHERE {
?h rdf:type <http://umbel.org/umbel/sc/HotelBuilding> ;
p:location <http://dbpedia.org/resource/Heraklion>.
{?hotel rev:hasReview [rev:text ?txt1]
TRUST AS ?t1 }
```

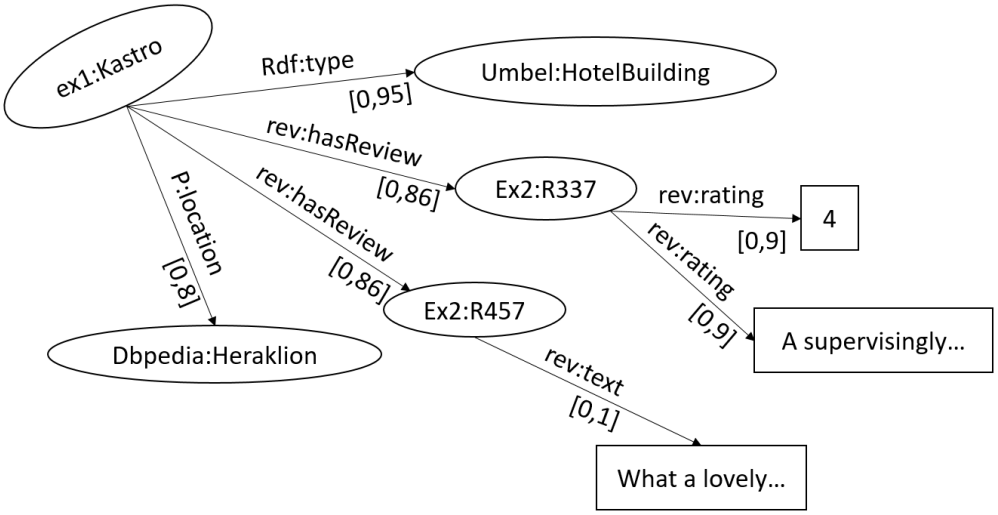


FIGURE 1.8 – Example trust weighted RDF graph.

```
OPTIONAL { ?h rev:hasReview [rev:text ?txt2]
TRUST AS ?t2 FILTER ( ? t2 > ? t1 ) }
FILTER ( !BOUND (?txt2))
}
```

1.4.2 Probabilistic RDF data

To manage uncertainty residing in RDF data, an effort of research was conducted such in [Huang and Liu, 2009,Meiser et al., 2011,Lian and Chen, 2011]. Lian et al. [Lian and Chen, 2011] proposed to model uncertain RDF data by means of probability theory. Indeed, an RDF query is equivalent to a search over subgraphs of probabilistic graphs. The result of the query is the subgraphs that have high probabilities to match with a given query graph. Huang et al. [Huang and Liu, 2009] proposed to evaluate queries over probabilistic RDF data model. The fact that the sum degrees from a probabilistic distribution must be equal to 1 makes dealing with incompletely known information difficult in probability theory.

Definition 1.3 (Probabilistic RDF Database)**.** *A probabilistic RDF datatabase D as defined in [Huang and Liu, 2009] is a finite set of probabilistic triples. They denoted triples(D) as all triples in the database. Indeed, each triple t is associated with a unique identifier and a probability value. The triple t has the form* $< s, p, o, Pr(t), \tau(t) >$ *where* $s \in Indvs, p \in Props \cup \{rdf : type\}$, $o \in Indvs \cup L \cup C$. *s, p and o are subject, property and object of triple t. Pr: triples(D)→[0,1] is a probability function and* τ *: triples(D)* →*Strings*

is a mapping form each t to a unique identifier which they called the event of triple t. They interpreted the RDF probabilistic database D in terms of possible worlds. A database instance is any subset $I_i \subseteq triples(D)$*.*$I_1, I_2, ..., I_n$ *are called possible worlds.* $Pr(I_i) = \prod_{t \in I_i} Pr(t). \prod_{t \notin I_i}(1 - Pr(t))$*. The sum of all probabilities of possible instances is 1, i.e.,* $\sum_{I_i} Pr(I_i) = 1$

A set of RDF triples can be represented as a graph. The concept of probabilistic RDF graph was introduced by [Huang and Liu, 2009] as follows:

Definition 1.4 (Probabilistic RDF Graph)**.** *A probabilistic RDF graph is a labeled directed graph denoted by* $G = (N, E, \ell_E, \tau, Pr)$ *where:*

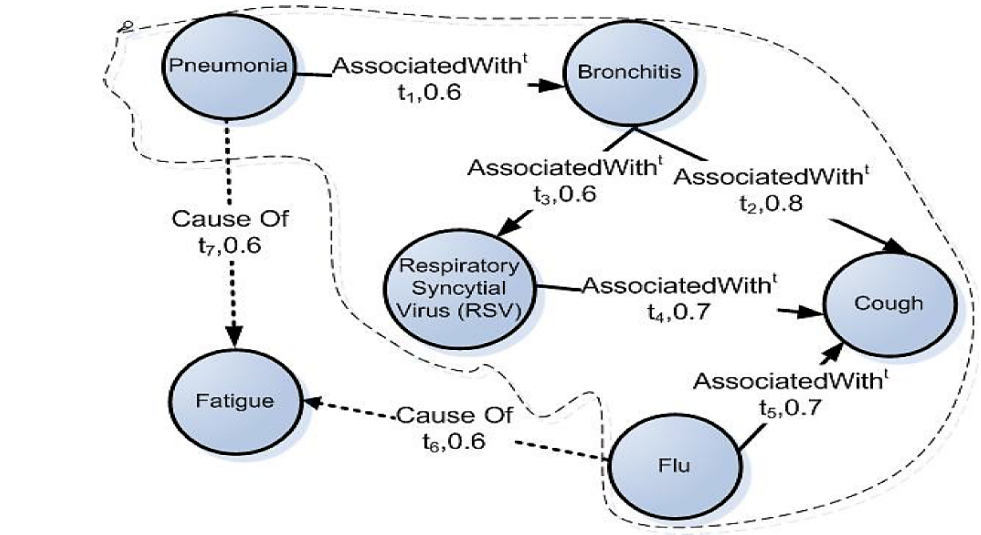
1. $N \subseteq Indvs$ *is a set of nodes.*
2. $E = \{edge_{s,p,o} : (s, p, o) \in G\}$
3. ℓ_E *is a labeling function such that* $\ell_E(edge_{s,p,o}) = p$*.*
4. $\tau : E \rightarrow Strings$*, is a mapping from edges to a unique identifier (event).*
5. $Pr : E \rightarrow [0, 1]$ *is a mapping from edges to a probability value.*

Example 12. *We illustrate an example of probabilistic RDF graph from [Lian and Chen, 2011] in Figure 1.9, where a SPARQL query is posed on a probabilistic diseases database. The database consists of RDF triples with probabilities showing the confidence about the relationship. The query asks for the diseases that are associated with cough and cause of fatigue.*

As a conclusion, to deal with uncertain RDF data the trust model associates RDF statements with trust values in the interval [-1, 1] and explains the trustworthiness of data as a subject of belief and disbelief in the truth of the information. Meanwhile, the probabilistic one models uncertain RDF data through the means of a mathematical model (i.e., the probability theory). The veracity of data is expressed with a probability measure in the interval [0, 1]. Moreover, using the probability theory leads for more expressivity to reason on uncertain RDF data, to benefit from its rich mathematical model such in information fusion, in reasoning, etc.

1.4.3 Other uncertain RDF models

In [Zhu et al., 2013], authors extends description logics with possibilistic semantics to reason with inconsistent and uncertain knowledge, which in a sense would subsume possibilistic RDF data. Straccia et al. [Straccia and Mucci, 2015] applied fuzzy logics to deal




```
SELECT ?X
WHERE {
    ?X  CauseOf   Fatigue      (q1)
    ?X  Associatedwith^t  Cough   (q2)
   }
```

FIGURE 1.9 – SPARQL query on a probabilistic RDF database.

with uncertain semantic data. The authors focused on the problem of automatically learn concept descriptions from OWL 2 data.

Furthermore, uncertainty over RDF data could also be seen as if we make an assessment over a triple, where this assessment is the possibility degree of the triple. This is basically the definition of Reification [Berners-Lee, 2004] which is a well known tool to express properties of triples using the triples format (statement-level metadata), where a large amount of work has been done [Hernández et al., 2015] [Hartig and Thompson, 2014].

To refer to a reified triple, it's necessary to include four additional RDF triples. Such addition for every reified triple makes writing queries to access statement-level metadata awkward. Furthermore, This form is inefficient for exchanging as well as for managing RDF data. To addresses the aforementioned shortcomings, other works offer alternative to reification for the treatment of uncertainty [Nguyen et al., 2014].

In our thesis project we do not claim that the posssibilistic framework is "preferable" to probabilistic one but it provides an interesting alternative to deal with different facets of uncertainty. Therefore we present in Chapter 4 a possibilistic model for RDF data.

Conclusion

The reason for creating the semantic Web is to bring structure to the meaningful content of Web pages to facilitate and automate its access. To reach the semantic Web the W3C introduced a recommendation for semantic annotations; the RDF data model. During this chapter, we presented the RDF data that have reached reasonable degree of maturity. RDF is a data model that expresses statements about objects or resources over the Web. Within semantic web applications, storing and querying RDF data is one of the basic tasks. Therefore, we presented some native and non-native storage approaches. To query RDF data, many querying languages have been proposed, the most used querying language is SPARQL. We presnted its specification, syntax and utility for querying RDF data. SPARQL is an SQL-like for querying language, of declarative nature. As a main part of our dissertation we aim to model and query uncertain RDF data. Hence, SPARQL needs to be extended to an uncertainty-aware query language.
The semantic Web becames largely adopted, thus, the amount of RDF data on the Web evoluates, with datasets increasing in variety and volume. However, during data integration, uncertainty on RDF data can result from either imprecision/inaccuracy of sources or from inconsistency between them. Therefore, a research effort have been done to extend RDF formalism, we presented in the last Section, the main works done to model imperfect RDF data, i.e, trust-weighted and probabilistic RDF data.

Chapter 2

Background on Possibility Theory and Skyline Queries

Contents

Introduction **38**

2.1 Possibility theory: An overview **38**

2.1.1 Typology of imperfect information 38

2.1.2 Uncertainty theories: A refresher 40

2.1.3 Possibility Theory 42

2.1.4 Possibility theory vs Probability theory and Evidence theory 44

2.1.5 Possibilistic Databases 46

2.2 Skyline queries **47**

2.2.1 Principle 47

2.2.2 Skyline Computation Algorithms 49

2.2.3 Skyline Queries over Incomplete Data 52

Conclusion **59**

Introduction

Volume and veracity of data on the Web are two main issues in managing information. In this chapter, we tackle these two issues, for veracity management, we rely on a powerful uncertainty theory, namely possibility theory, which is a non-classical and non-bayesian theory of uncertainty [Zadeh, 1978]. We recall here in Sction 2.1, the basic foundations of this theory to make our contribution presented in Chapter 4 more easy to read and to understand. On the other hand, to manage the large volume of information available in today's databases and Information Systems, we present the preference-based queries. The aim from using such queries is to extract and filter data according to users' preferences. Among all preference relations, skyline queries, defined using Pareto dominance, have been the most extensively studied [Börzsönyi et al., 2001a, Jiang et al., 2012, Zhang et al., 2013, Chomicki et al., 2013] and extended over Graph Data such as in [Zou et al., 2010, Zheng et al., 2014]. The skyline operator aims to make multi-objective decisions over complex data. Given such a multi-criteria preference, the system needs to identify all potentially interesting data records according to users' preference [Chomicki, 2002, Chomicki, 2011]. The basis of skyline model are presented in Section 2.2 which are necessary for the reading both chapters 3 and 4. We recall in this Chapter two main concepts for the understanding of our work. Section 2.1 presents the rich Possibility theory, while, Section 2.2, recalls the main concepts related to the skyline operator.

2.1 Possibility theory: An overview

2.1.1 Typology of imperfect information

The information about the real world needs to be stored in an information system to be used in the process of decision-making. However, information is almost always imperfect [Smets, 1996] [Dubois and Prade, 1988]. Therefore, imperfection must be incorporated into every information system that attempts to provide a complete model of the real world. This remains a major challenge because of the difficulty of understanding the various aspects of imperfection. The use of inappropriate models can lead to results that might be misunderstood by end users.
There exist many aspects of imperfection. We can group them into three main groups, namely inconcistency, imprecision and uncertainty.

2.1.1.1 Imprecision

A statement is imprecise (or incomplete), if it is insufficient for an agent to answer to a given question. Imprecision is a property related to the information itself (i.e content of the statement). Indeed, more than one world is compatible with the available information.

Example 13. *Assume a query: Give the price (per night) of hotel h. The imprecision is not an absolute notion. It depends on the proper frame Pr. Let v denotes the price of a given hotel:*

- $Pr = \{cheap, expensive\}$*, v=cheap is precise,*
- $Pr = \{0, 1, \ldots, 500\}$*, the term cheap is imprecise in this case, it provides incomplete information.*

2.1.1.2 Inconsistency

When combining several statements, different aspects of imperfection can appear. Indeed, anything that affects the data integrity results in data inconsistency. [Smets, 1996] explains that inconsistency is better used when time is involved, for example, someone claimed that at 3 p.m. hotel H_1 is 3 stars, at 3.15 p.m. the same hotel is 2 stars.
Inconsistency is used to define the incoherence that results from a conflicting information. Conclusions will be confused when incoherent and when the involved incoherence can be recovered by some small modifications of the data.

Example 14. *A person P announced to arrive to hotel H_1 by train at 3.05 p.m. but the train is scheduled to arrive at 3.15 p.m. The incoherence is much smaller than would be the case if there was no train arriving in the afternoon. In the first case, the hotel's reception accept that he will arrive just after 3 p.m. whereas in the second case they hardly know what to accept.*

2.1.1.3 Uncertainty

Uncertainty is an epistemic or random property that concerns the state of knowledge of an agent about the relationship between the world and a statement about the world. Therefore a statement is said to be uncertain if we have lack of information about the world for deciding if this statement is true or false.
Uncertainty could be seen as subjective or objective property [Smets, 1996]. Indeed, uncertainty related to randomness is an objective property and the term likely qualifies an event that will probably occur. Objective properties of uncertainty are linked to the world

and to the information. Subjective properties of uncertainty are linked to agents opinions about the true value of the data.
Note that any uncertain information can have either an uncertain measure which can be numeric (line 1), symbolic/linguistic(lines 2 and 3) or an interval.

Example 15. *An example of uncertain information can be as follows:*

1. *The probability that hotel H_1 has a distance (from the beach) equal to 2Km is 0.9.*
2. *It is possible that hotel H_1 is the closer one to the beach.*
3. *I believe that hotel H_1 is 2 Km away from the beach.*

We present in Figure 2.1 the uncertainty models which we detail in Section 2.1.2.

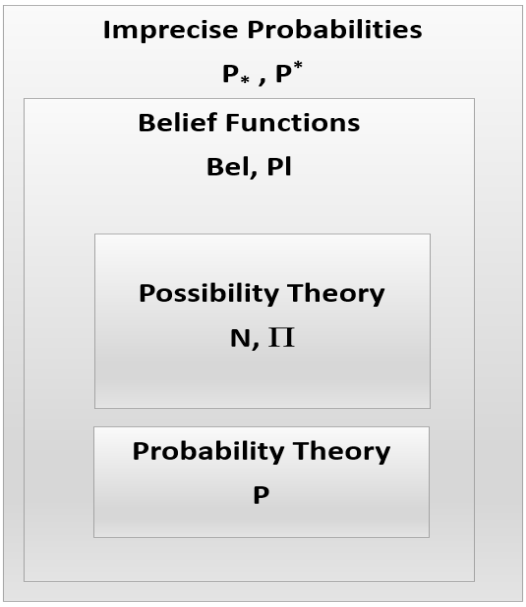


FIGURE 2.1 – Uncertainty weighted models

2.1.2 Uncertainty theories: A refresher

2.1.2.1 Probability theory

Probability theory, introduced in the seventeenth century by Laplace (1774), is the oldest among uncertainty theories and the most widely acknowledged.

Definition 2.1 (Probability distribution)**.** *The probability distribution is a non-negative mapping $p : \Omega \rightarrow [0, 1]$ that quantifies the degree of probability P(A) that an arbitrary element $x \in \Omega$ belongs to a well-defined subset $A \subseteq \Omega$ such that $\sum_{A \in \Omega} p(x)$=1.*

Definition 2.2 (Measure of probability P)**.** *Let $A \subseteq \Omega$ an event:*

$$P(A) = \sum_{\omega \in A} p(\omega).$$

It satisfies the following properties:

— $P(\emptyset) = 0$ *and* $p(\Omega) = 1$

— $\forall A, B \subseteq \Omega : if A \cap B = \emptyset \Rightarrow P(A \cup B) = P(A) + P(B)$ (Additivity)

— $\forall A, B \subset \Omega$, $P(A) = 1 - P(A^c)$, with A^c is the opposite event of (A) (Duality)

Probability theory could have two main interpretations [Smets, 1996]:

Frequentist (randomness) probability theory: The probability of an event is the ratio of the number of favorable cases to the number of all possible cases. It capture variability through repeated observations and rely on statistical data.

Subjective (Bayesian, belief) probability theory: The probability of an event is a subjective measure. Formally given, it is the degree of credibility that an agent grants to the occurrence of an event A or that a proposition is true. It models unreliable evidence and not necessary related to statics. Therefore, precise probability values are difficult.

2.1.2.2 Evidence theory

Evidence theory is a formal framework for representing and reasoning with imperfect data. Also known as Dempster-Shafer theory or Belief Function theory, it originates from the work of Dempster [Dempster, 1967] in the context of statistical inference after that formalized by Shafer [Shafer, 1976] as a theory of evidence. It extends both the set-membership approach and Probability Theory notions. Indeed, it is a generalization of the Bayesian theory of subjective probabilities.

Definition 2.3 (Frame of Discernment)**.** *A frame of discernment denoted as θ and defined as $\theta = \{\theta_1, \theta_2, ..., \theta_n\}$, a finite set of n exhaustive and mutually exclusive hypothesis for a given problem.*

Definition 2.4 (Mass Function)**.** *A mass function m or basic belief assignment (bba) is defined as m_θ. Let A be a variable taking values in a finite set (frame of discernment). Evidence about A may be represented by a mass function: $2^\theta \rightarrow [0, 1]$ and has the properties below:*

$$m_\theta(\varnothing) = 0, \; m \text{ is normalized}$$

$$\sum_{A \subseteq \theta} m(A) = 1$$

Every proposition A of θ such that $m(A) > 0$ is a focal set of m.

Definition 2.5 (Belief function)**.** *A function bel :* $2 \rightarrow [0;1]$ *also called belief function is defined as follows:*

$$bel_\theta(A) = \sum_{B \subseteq A} m_\theta(B)$$

Definition 2.6 (Plausibility function)**.** *A dual function of belief. The plausibility function:* $pl_\theta : 2 \rightarrow [0;1]$ *is defined as the sum of the masses assigned to every subset B of θ that intersects A, i.e.,*

$$pl_\theta(A) = \sum_{B \cap A \neq \emptyset} m_\theta(B)$$
$$pl_\theta(A) = 1 - Bel_\theta(\bar{A})$$

2.1.3 Possibility Theory

Possibility theory is a non classical theory of uncertainty devoted to handle imperfect informations. This section presents some basic concepts of this theory, more details are available in [Zadeh, 1978] [Dubois and Prade, 1988].

2.1.3.1 Possibility distribution

Imperfection has many forms, mainly uncertainty, imprecision and inconsistency. Possibility theory, introduced in [Zadeh, 1978] and developed by [Dubois and Prade, 1988] is an uncertain theory devoted to the handling of incomplete knowledge.
Given U a referential (set of the states of the world), a possibility distribution π is a mapping from U to a totally ordered scale L (e.g., L=[0, 1]). The function π represents the state of knowledge of an agent distinguishing what is possible (plausible) from what is less possible. Note that possibility theory provides a good representation of extreme forms of partial knowledge such total ignorance (all propositions are possible) and complete knowledge:

— $\pi(u) = 0$ means that state u is rejected as impossible;

— $\pi(u) = 1$ means that state u is totally possible (plausible).

Meanwhile, when an information is vague possibility can admit degrees, and the larger the degree, the larger the possibility. The normalization condition of the possibility distribution imposes that at least one of the values of the domain (a_0) is completely possible (plausible) i.e., $\pi(a_0) = 1$.

We illustrate an example of incomplete information, «hotel h_1 distance from the beach $d.h_1$ is above 10 kilometers», means that any distance above 10 km is possible and any distance equal to or below 10 is impossible. By means of possibility theory we represent such information as

$$\pi(d.h_1) = \begin{cases} 0 & if\ d.h_1 < 20 \quad (impossible\ proposition) \\ 1 & if\ d.h_1 > 20 \quad (totally\ possible\ proposition) \end{cases}$$

2.1.3.2 Possibility and Necessity measures

Probability is self dual, for any event A, $P(\neg A) = 1 - P(A)$. Indeed, if A is not probable, then $\neg A$ is necessarily probable. Meanwhile, it is possible that A does not entail anything about the possibility of $\neg A$.

Thus, the description of uncertainty about the occurrence of A needs two dual measures: the possibility measure $\Pi(A)$ and the necessity measure N(A). These two dual measures are defined in possibility theory as follows:

Possibility measure
Given a subset of states A, it's possibility to occur in a possibility scale L=[0, 1] is as follows:

$$\Pi(A) = sup_{s \in A} \pi(s)$$

$\Pi(A)$ evaluates to what extent A is consistent with π. Table 2.1 illustrates the possibility measure properties.

$\Pi(A)$=1 and $\Pi(\neg A)$=0	A is certainly true
$\Pi(A)$=1 and $\Pi(\neg A) \in]0, 1[$	A is somewhat certain
$\Pi(A)$=1 and $\Pi(\neg A)$=1	total ignorance
$\Pi(A) > \Pi(\neg A)$=0	A is more plausible than B
$\max(\Pi(A), \Pi(\neg A)$=1	A and B cannot be both impossible

TABLE 2.1 – Possibility measure properties

Example 16. *Let us treat a classification problem of hotels choice. Suppose that the universe of discourse related to this problem is defined as follows:* $\Omega = \{H_1, H_2, H_3\}$. *Suppose a client who is providing his opinion about the best quality (BQ) between the hotels. The opinion is given in the form of a possibility distribution* π_1 *defined as follows:*

π_1($BQ = H_1$) = 0.2;
π_1($BQ = H_2$) = 0.4;
π_1($BQ = H_3$) = 1;
For instance, the degree 0.2 represents the degree of possibility that the best quality is of H_1. π_1 is normalized since max (0.2, 0.4, 1) is equal to 1. $\pi_1(BQ = H_3) = 1$ means that it is fully possible that the hotel's best quality is H_3.

Necessity measure
In addition to its possibility measure, an event A is characterized by its necessity N (expressing that A will occur more or less for sure). The possibility-necessity duality is expressed by:

$$N(E) = 1 - \Pi(\overline{A}) = 1 - \max_{x \in \overline{A}} \pi(x), \text{ where A is the event opposite to } \overline{A}.$$

Necessity measure evaluates at which level A is certainly implied by our knowledge represented by π. The following properties of possibility and necessity measures, where A, B *and* E denote events, are of interest further.

- $\Pi(A \cup B) = max(\Pi(A), \Pi(B))$
- $\Pi(A \cap B) = min(\Pi(A), \Pi(B))$ if A and B are logically independent.
- $N(A \cap B) = min(N(A), N(B))$
- $N(A \cup B) = max(N(A), N(B))$ if A and B are logically independent.
- $\Pi(E) < 1 \Rightarrow N(E) = 0$
- $N(E) > 0 \Rightarrow \Pi(E) = 1$

Example 17. *Let us consider the problem of disease detection. The universe of discourse related to this problem is defined as follows $\omega = \{d_1, d_2, d_3, h\}$. Suppose that a doctor gave his opinion on the patient state in the form of a possibility distribution π_1 i.e.: $\pi_1(d_1)$= 0.5, $\pi_1(d_2)$=1 , $\pi_1(d_3)$=0.7 ,$\pi_1(h)$=0, where d_i stands for disease and h stands for healthy. A:"The patient suffers from d1 or d3, then we have:*

- Π(A)=max{0.5,0.7}=0.7
- N(A)=min{(1-1), (1-0)}=0

2.1.4 Possibility theory vs Probability theory and Evidence theory

Probability theory is the classical method to manage statistical uncertainty. Probabilistic logic provides insufficient tools to handle all facets of imperfection, thus new theories has

emerged such belief function [Smets, 1996] and possibility theory. The latter differs from the probability theory by the use of dual set functions (possibility and necessity). While, probability theory relies on the use of a single probability distribution to represent uncertainty, this can raise some serious problems such as:
Ambiguity: No difference between uncertainty due to incomplete information and uncertainty due to randomness. In case of absence of information about some quantity Q, we should assign equal probability to any possible value of Q. A probability agent is unable to represent ignorance and does not have a plausibility model of how people make decisions based on weak information.

Example 18. *We illustrate the example of choosing between two hotels H_1 and H_2:*
Agent 1 knows that hotels $H1$ and H_2 are well-known:
$P(H_1) = P(H_2) = 1/2$ (Pure randomness)
Agent 1 ignore whether there is a disponibility in the both hotels:
$P(H_1) = P(H_2) = 1/2$ (Insufficient Reason Principle)

Instability: The same state of knowledge represented by incompatible distribution probabilities More details to understand the relationship between the uncertainty theories are given in [Dubois and Prade, 2001].

Example 19. *We introduce an example to show the difference between possibility and probability theory (inspired from [Zadeh, 1978]). Consider the statement "The price value of hotel H_1 is X per night", with $X \in \{10, 20, 30, 40\}$. A possibility distribution $\pi(X)$ is associated to each value of X($) to express the possibility of the event to accur. Also a probability distribution $P(X)$ is associated to express the probability that H_1 will be X. Table 2.2 below shows the corresponding values: We can observe from table 2.2 that a*

X($)	10	20	30	40	50	60
$\pi(x)$	1	1	0.5	0.6	0	0.8
P(x)	0.2	0.4	0.1	0.3	0	0

TABLE 2.2 – Possibility and probability values associated with X.

high degree of possibility does not imply a high degree of probability (e.g. $\pi(x = 20) = 1$ and $\pi(x = 20) = 0.4$) neither low degree of probability imply a low degree of possibility. Meanwhile, impossible events are bound to be improbable (e.g. $\pi(x = 50) = 0$ and $\pi(x = 50) = 0$).

2.1.4.1 Possibility theory vs belief function theory

Let us note that possibility theory in its numerical setting is considered as a special case of belief function theory [D. Dubois, 1982]. In case focal elements F_i are nested (i.e. $F_1 \subset F_2 \cdots \subset F_n$), the believe function Bel is called consonant bba and for all $f, g \in \Omega$ we have:

— $Bel(f \wedge g) = \min(Bel(f), Bel(g))$; a belief function is a necessity measure N.

— $Pl(f \vee g) = \max(Pl(f), Pl(g))$; a plausibility function is a possibility measure Π.

2.1.5 Possibilistic Databases

Possibilistic databases are first introduced by [Prade, 1984]. In fact, a possibilistic database D can be interpreted as a weighted disjunctive set of regular databases (i.e. Worlds Wor interpretations) denoted by rep(D). Any world W_i corresponds to a conjunction of independent choices. Accordingly, to each world W_i is associated the minimum degree of possibility tied to each of the chosen candidate values in the possibilistic database D. At least one possible world is completely possible ($\Pi = 1$).

Example 20. *Assume a possibilistic database D involving a relation pr, having a schema PR(#i, pri, dist, loc) representing properties of some hotels. Each hotel is identified with an attribute (#i), a price value/night (pri), a distance from the beach (dist) and specific location (loc) as illustrated in Table 2.3.*

$\#i$	pri	dist	location
H_1	$\{p_1/1, p_2/0.8\}$	$\{d_1/1, d_2/0.3\}$	l_1
H_2	$\{p_3/1, p_4/0.6\}$	d_1	l_2

TABLE 2.3 – A possibilistic relation pr.

- $W_1 = \{< H_1, p_1, d_1, l_1 >, < H_2, p_3, d_1, l_2 >\}$, $\Pi = 1$
- $W_2 = \{< H_1, p_1, d_2, l_1 >, < H_2, p_3, d_1, l_2 >\}$, $\Pi = 0.3$
- $W_3 = \{< H_1, p_1, d_1, l_1 >, < H_2, p_4, d_1, l_2 >\}$, $\Pi = 0.6$
- $W_4 = \{< H_1, p_1, d_2, l_1 >, < H_2, p_3, d_1, l_2 >\}$, $\Pi = 0.3$
- $W_5 = \{< H_1, p_2, d_1, l_1 >, < H_2, p_3, d_1, l_2 >\}$, $\Pi = 0.8$
- $W_6 = \{< H_1, p_2, d_2, l_1 >, < H_2, p_3, d_1, l_2 >\}$, $\Pi = 0.3$
- $W_7 = \{< H_1, p_2, d_1, l_1 >, < H_2, p_4, d_1, l_2 >\}$, $\Pi = 0.6$

- $W_8 = \{< H_1, p_2, d_2, l_1 >, < H_2, p_4, d_1, l_2 >\}$, $\Pi = 0.3$

As illustrated in Table 2.3, we have 8 possible words, to each possible world is associated a possibility degree Π and one of them pr_1 is completely possible.

2.2 Skyline queries

In this section we illustrate some basic concepts related to preference queries, such as preference relation and Pareto dominance. Finally, we present the most extensively studied preference queries, the Skyline queries.

2.2.1 Principle

In a choice case with no uncertainty, choice became easier knowing the consequences of each option. The alternatives choice became difficult when available options have strengths and weakness that trade off against each other. The researches in literature about preference are extensive; meanwhile a few of them tackle the problem of the user preferences over database queries. In that field, two approaches are pursued qualitative and quantitative [Chomicki, 2002].
Under Pareto efficiency, the dominance relationship has a particularly simple structure: Let be two alternatives $A1$ and $A2$. We say that A_1 dominates A_2 if A_1 is better than or equal to A_2 in all dimensions and strictly better than A_2 in at least one dimension.

Definition 2.7 (Pareto Dominance)**.** *Let X and Y be two points defined in a set of points denoted $\mathcal{H}$ with m attributes. A point Y dominates a point X denoted by $Y \succ X$, if $\forall i \in$ [1, m] $y_i \leq x_i \wedge \exists j, y_j < x_j$ [1]. The logical dominance concept between two points is modelled as follows:*

$$Y \succ X = \bigwedge(\bigwedge_{1 \leq i \leq n} y_i \leq x_i, \bigvee_{1 \leq i \leq n} y_i < x_i)$$

Under the Pareto efficiency, the preference relation between alternatives could be defined in two ways, Strong Pareto-dominance and Weak Pareto-dominance. A point Y strongly Pareto-dominates a point X if $\forall i \in$ [1, n] $y_i < x_i$ otherwise, it is considered as weak. The algorithms for evaluating Pareto preferences have been examined in the context of skyline algorithms.

1. We assume in this definition, that the smaller value is the more preferable

The skyline preference operator uses Pareto accumulation. In a set of tuples denoted by S, the skyline consists of the tuples which are dominated by no other tuple [Börzsönyi et al., 2001a]. Skyline aims to make intelligent decisions over multi-dimensional data. Hence, it consists in extrecting the most interesting objects according to user-defined criteria (user's preference). A skyline analysis involves multiple attributes. A user's preference on the values of an attribute can be modeled by a partial order on the attribute. A partial order is irreflexive, asymmetric and transitive relation.

A formal definition of the skyline is given as follows:

Definition 2.8 (Skyline)**.** *Let $\mathcal{H}$ be a set of points having m attributes. The skyline of $\mathcal{H}$ denoted by S is defined as:*

$$S = \{X \in \mathcal{H} / \nexists Y \in \mathcal{H}, Y \succ X\}$$

In order to specify skyline queries, [Börzsönyi et al., 2001a] proposed a new relational operator to extend SQL's SELECT statement by an optional *SKYLINE OF* clause as follows:

```
SELECT ... FROM ... WHERE ...

GROUP BY ... HAVING ...

SKYLINE OF [DISTINCT] d1 [MIN | MAX | DIFF],..., dm [MIN | MAX | DIFF]

ORDER BY...
```

The SKYLINE OF clause selects all the interesting records that are dominated by no other point; the dominance between records depends on the user preference (i.e. minimum, maximum, etc.). Hence many constraints could be satisfied at the same time between different dimensions having specific preference for each dimension.

Example 21. *We illustrate an example from the work of [Börzsönyi et al., 2001a]. Given a list of hotels having two attributes* price *and* distance (from the beach)*, we aim to find the hotels having the least price and distance values (skyline candidate list) from the data set in Table 2.4.*
We illustrate in Figure 2.2 the set of hotels where each point is characterized by two values: a price and a distance. Indeed, points in the curve represent the skyline set, i.e., hotels dominated by no other point according to the above-mentioned criteria (minimum price and distance). Tuple (h_1, 27, 2) is dominated by no other tuple since it has the least price among all the tuples. Tuple (h_6, 70, 1.9) is dominated by the tuple (h_5, 70, 1.4) therefore it is pruned from the skyline candidate list.

Hotel	Price	Distance
h_1	27	2
h_2	43	1.75
h_3	55	1.50
h_4	66	1.7
h_5	70	1.40
h_6	70	1.9
h_7	80.12	0.80
h_8	90	1
h_9	125	0.56
h_{10}	125	1.4
h_{11}	140	0.50
h_{12}	160	1
h_{13}	190	0.30

TABLE 2.4 – Example of hotels properties.

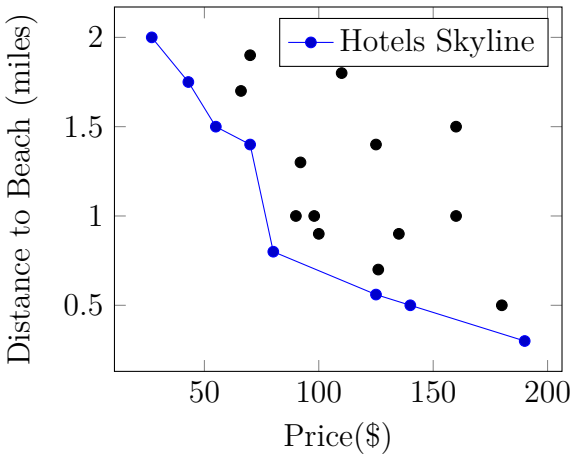


FIGURE 2.2 – Skyline of Hotels

Many researches after [Börzsönyi et al., 2001a] were conducted to improve the use of skyline over database sets such as [Tan et al., 2001, Chomicki, 2002] . We present in the next Section the main skyline Algorithms used in literature.

2.2.2 Skyline Computation Algorithms

Several algorithms to compute the skyline are proposed in the literature. Here after, we present three families of these algorithms. **Block Nested Loop Algorithm**

The block nested loop algorithm is an iterative algorithm that repeatedly scans a set of records. In each iteration, a window of incomparable records is kept in the main memory. When a record r is read from the input relation, r is compared with the records in the window. There are three possible outcomes:

1. If r is dominated by a record in the window, it means that r cannot be in the skyline. Thus, r is prunned.
2. If r dominates one or more records in the window, these records are eliminated (since they cannot be in the skyline), and r is inserted into the window.
3. If r is incomparable with all records in the window (i.e., it neither dominates nor being dominated), it is either inserted into the window if there is sufficient room in the window, or written to a temporary file on disk.

At the end of each iteration, those records in the window that have been compared against all records that have been written out to the temporary file are certain to be in the skyline, and hence can be returned to the user (removed from the window). In the next iteration, the algorithm will proceed in the same manner with the remaining records in the window and the records in the temporary file as the input relation.

The nested loop algorithm is a naïve way to compute the skyline, where every tuple is compared to every other tuple. When a new tuple h_i is read from the input, h_i is compared to all tuples of the window. Based on this comparison, h_i is either eliminated (pruned), placed into the window or into a temporary file (no enough rooms in the window) to be considered in the next iteration of the algorithm. The complexity of the BNL algorithm is O(n) in the best case and is of the order of $O(n^2)$ in the worst case. The researches continue to improve the results of this algorithm [Börzsönyi et al., 2001a] [Deepak et al., 2011].

Divide and Conquer Algorithm

Authors in [Börzsönyi et al., 2001a] extended the basic divide-and-conquer algorithm for computing skyline from two-way to m-way partitioning. The basic divide and conquer was proposed by [Preparata and Shamos, 1985] [Kung et al., 1975] , the best known algorithm in the worst case $\mathcal{O}(n * (log\ n)^{(d-2)}) + \mathcal{O}(n * log\ n)$; where n is the number of input and d is the number of dimensions in the skyline.

The basic Divide and Conquer algorithm presented in [Preparata and Shamos, 1985] [Kung et al., 1975] works as follows:

1. Computing the median M_p of the input for some dimension d_p (e.g., distance) then Dividing the input into two partitions. Partition $P1$ contains all tuples whose attri-

bute's values d_p is better (e.g., greater) than M_p. $P2$ contains all other tuples.

2. Computing the skylines S_1 of partition P_1 and S_2 of partition P_2. This is computed by recursively applying the whole algorithm to P_1 and P_2 (i.e., P_1 and P_2 are again partitioned). The recursive partitioning stops if a partition contains only one (or very few) tuple. For such case, computing the skyline is trivial.
3. Computing the overall skyline as the result of merging S_1 and S_2. That is pruning tuples of S_2 which are dominated by anothor tuple in S_1. (None of the tuples in S_1 can be dominated by a tuple in S_2 because a tuple in S_1 is better in dimension d_p than all tuples of S_2.)

Authors in [Börzsönyi et al., 2001a] proposed an extension of the basic divide and conquer algorithm to an M-way Partitioning. indeed, authors noticed that in case the input does not fit into main memory, the basic algorithm shows terrible performance. The M-way partitioning algorithm can be used in the first step of the basic algorithm and also in the third step. In the first step, M-way partitioning is used to produce M partitions denoted $P_1, \ldots, P_M$. Hence, each P_i fits into memory and S_i, the skyline of P_i, can be computed in memory using the basic algorithm. In the third step, the final answer is produced by merging the S_i pairwise. Within the merge function, M-way partitioning is applied thus all sub-partitions can be merged in main memory.

B-trees Algorithm

B-trees generalize binary search trees in a natural manner. If a B-tree node x contains $n[x]$ keys, then x has $n[x] + 1$ children. The keys in node x are used as dividing points separating the range of keys handled by x into $n[x] + 1$ subranges, each handled by one child of x. When searching for a key in a B-tree, we make an $(n[x]+1)$-way decision based on comparisons with, then[x] keys stored at node x.

The algorithm B-tree allows the use of an ordered index for a two-dimensional skyline by scanning throw the whole index and getting the tuples in sorted order and filter out the tuples of the skyline. In this algorithm [Börzsönyi et al., 2001a] proposed to use two ordered indices: one on *hotel.price* and one on *hotel.distance* they used those indices to find a superset of the skyline. They used the first step of the Fagin's FA algorithm for merging scores from multimedia databases by scanning simultaneously through both indices and stop as soon as there is a match.

Firthermore, authors exploited the fact: Given a hotel h, there is no need to search in any branches of the R-tree which are guaranteed to contain only hotels that are dominated by h. The idea is to traverse the R-tree in a depth first way and to prune branches of the R-tree with every new hotel found.

We present in Table 2.5 a comparison between the three algorithms [Kalyvas and Tzouramanis, 2017]:

Algorithm	Based on	Pre-processing	Complexity	Main problem
BNL	Naive Nested Loop	Sort-based	$O(n^2)$	Not online
D&C	Maximal Vector Computation	Partial skylines can be assumed	$O(n^2)$	Not online/ curse of dimensionality
B-trees	Merging scores from multi-media databases	ordered index	O(log n)	Lack of user interaction

TABLE 2.5 – Comparison of skyline computation algorithms

2.2.3 Skyline Queries over Incomplete Data

As uncertainty infects most of modern real world applications, conducting advanced analysis on uncertain data remains a main and hot topic since the late 2000s, a research effort has been done to extend skyline queries over uncertain data. We present in Table 2.6 a comparative study between the probabilistic model and the possibilistic one.

Distributions		Dominance relationship	Degree to be in S
Probabilistic	Continuous case	$Pr[V \prec U] = \int_{u\in D}\int_{u\prec v} f(u)f'(v)\ dv\ du$	$Pr(U) = \int_{u\in D} f(u)\prod_{V\neq U}(1 - f'(v)dv)du$
	Discrete case	$Pr[V \prec U] = \frac{1}{l_1 l_2}\sum_{i=2}^{l_1}\lvert\{v_j \in V/\ v_j \prec u_i\}\rvert$	$Pr[U] = \frac{1}{l}\sum_{u\in U} Pr(u)$
Possibilistic		$\begin{cases} 0 \ \ if\{\pi_i/t'_j \in int(t')\lvert t_i \prec t'_j\} = \varnothing \\ \max_{\pi_j/t'_j\in int(t')\lvert t_i\prec t'_j}\pi_j \ \ Otherwise \end{cases}$	$\Pi(t) = \max_{\pi_i/t_i\in int(t)}\Pi(\pi_i/t_i)$

TABLE 2.6 – Imperfect skyline models

2.2.3.1 Probabilistic Skyline Queries

The probabilistic skyline queries were firstly introduced by [Pei et al., 2007]. In the uncertain case, a set of object has multiple instances. Indeed, each object is associated with a probability function. A main task to model skyline queries on uncertain data is to redefine the dominance relation between uncertain objects and to choose the skyline on those uncertain objects.

Given a set of uncertain object U representing a set of multiple points in the data space having instances, denoted by $U = u1, \ldots, ul$. It can be considered as the discrete case. For an uncertain object U, The number of its instances is denoted as $|U| = l$.
Assume two uncertain objects U and V and f and f' be their corresponding probability density functions, respectively. The probability that V dominates U is expressed as follows:

$$\begin{aligned} Pr[V \prec U] &= \int_{u \in D} f(u)(\int_{u \prec v} f'(v)\ dv)\ du \\ &= \int_{u \in D} \int_{u \prec v} f(u) f'(v)\ dv\ du \end{aligned} \tag{2.1}$$

For the discrete case, let $U = u1, \ldots, ul1$ and $V = v1, \ldots, vl2$ be two uncertain objects and their instances. The probability that V dominates U is given by:

$$\begin{aligned} Pr[V \prec U] &= \sum_{i=2}^{l_1} \frac{1}{l_1} . \frac{|\{v_j \in V /\ v_j \prec u_i\}|}{l_2} \\ &= \frac{1}{l_1 l_2} \sum_{i=2}^{l_1} |\{v_j \in V /\ v_j \prec u_i\}| \end{aligned} \tag{2.2}$$

Example 22. *We illustrate an example from [Pei et al., 2007] to explain a probabilistic dominance ralation. Consider a set of four uncertain objects as shown in Figure 2.3. The*

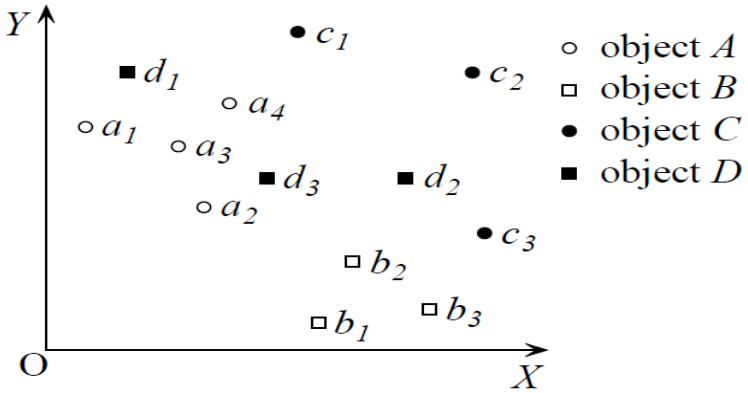


FIGURE 2.3 – An uncertain objects set.

object C has three instances, where, c_1 and c_2 are dominated by every instance of A, and c_3 is dominated by no other instance of A. Therefore, the probability that object A dominates object C is $Pr[A \prec C] = \frac{1}{3} \times \frac{1}{4} + (4 + 4 + 0) = \frac{2}{3}$.
We detail the result here, $\frac{1}{3}$*(C has 3 instances),* $\frac{1}{4}$*(A has 4 instances), 4(c_1 is dominated*

by every instance of A), $4(c_2$ *is dominated by every instance of A),* $0(c_3$ *is dominated by no instance of A). Similarly,* $Pr[B \prec C] = \frac{2}{3}$.

Definition 2.9 (The probability skyline of U)**.** *Let an uncertain object U having a probability density function f. The function, f(u) represents the probability that U appears at position u in a data space D. For any other object* $V \neq U$ *having probability density function f′, the probability that V dominates u is given as* $\int_{u \prec v} f'(v)\, dv$*. Therefore, the probability that u isdominated by no other object is modelled by* $\prod_{V \neq U}(1 - f'(v)dv)$*. The probability that U is in the skyline set is as follows:*

$$Pr(U) = \int_{u \in D} f(u) \prod_{V \neq U} (1 - f'(v)dv)du \tag{2.3}$$

In the discrete case, given an uncertain object $U = \{u_1, \ldots, u_l\}$*, the probability that U is in the skyline is as follows:*

$$Pr(U) = \frac{1}{l}\sum_{i=1}^{l}(1 - \frac{|\{v \in V/\ v \prec u\}|}{|V|}) \tag{2.4}$$

Authors in [Pei et al., 2007] extended the notion of skyline to probabilistic skyline denoted p-skyline. Indeed, an uncertain object can take a probability measure to be in the skyline set.

Definition 2.10 (Probabilistic Skyline)**.** *Given a set of uncertain objects S and a probability measure* $p(0 \leq p \leq 1)$*, the p-skyline is the subset of objects in S each of which takes a probability of at least p to be in the skyline. That is:*

$$Sky(p) = U \in S/\ Pr(U) \geq p.$$

Example 23. *We illustrate an example from [Pei et al., 2007]. Consider objects modelled in Figure 2.3, each instance of object A is dominated by no other instances of objects B, C or D. Therefore, the probability that A is dominated by no other object is 0 and the probability that A is in the skyline set is 1. Also for object B, the probability that it is in the skyline set is 1. Meanwhile, for instances of object D,* d_1 *is dominated by* a_1*,* d_2 *is dominated by* a_2*,* b_1 *and* b_2*, and* d_3 *is dominated by* a_2*. Thus, the probability that D is dominated by no other object is as follows:*

$\frac{1}{3}($
$(1 - \frac{1}{4}) +$ *case of* d_1
$(1 - \frac{1}{4})(1 - \frac{2}{3}) +$ *case of* d_2

$(1 - \frac{1}{4})) +$ *case of* d_3
$= \frac{7}{12})$

After the work of [Pei et al., 2007] a profusion of research have been done to extend skyline queries over uncertain data such [Lian and Chen, 2009, Jiang et al., 2012, Hyountaek et al., 2014].

2.2.3.2 Possiblistic Skyline Queries

The work of [Bosc et al., 2011] submitted the Skyline queries to a possibilistic database. The concept of dominance over possibilistic database was modified, they compute the extent to which any tuple from a given relation is possibly/certainly not dominated by any other tuple.
Authors in [Bosc et al., 2011] defined the degree of possibility $\Pi(t)$ that a tuple t from the result of the query res be non-dominated by any other tuple t' from res is computed as follows. For each interpretation π_i/t_i of t, one computes the possibility that for every tuple $t' \neq t$, there exists an interpretation t'_j of t' which does not dominate t_i. The final degree $\Pi(t)$ is the maximum of these degrees, computed over all the interpretations of t. They define the degree of possibility as follows:

$$\Pi(t) = \max_{\pi_i/t_i \in int(t)} \Pi(\pi_i/t_i) \tag{2.5}$$

where $int(t)$ denotes the set of interpretations of t and

$$\Pi(\pi_i/t_i) = \min(\pi_i, \min_{t' \in res \setminus \{t\}} \Pi(t_i \not\prec t')) \tag{2.6}$$

With

$$\Pi(t_i \not\prec t') = \begin{cases} 0 & if \{\pi_i/t'_j \in int(t') | t_i \not\prec t'_j\} = \varnothing \\ \max_{\pi_j/t'_j \in int(t') | t_i \not\prec t'_j} \pi_j & otherwise \end{cases} \tag{2.7}$$

The degree of possibility $\Pi'(t)$ that tuple t is dominated by any other tuple t' was defined by [Bosc et al., 2011] as follows:

$$\Pi'(t) = \max_{\pi_i/t_i \in int(t)} \Pi'(\pi_i/t_i) \tag{2.8}$$

Where $\Pi'(\pi_i/t_i) = \min(\pi_i, \max_{t' \in res \setminus \{t\}} \Pi(t_i \prec t'_j))$ and

$$\Pi(t_i \prec t'_j) = \begin{cases} 0 & if\{\pi_i/t'_j \in int(t') | t_i \prec t'_j\} = \varnothing \\ \max_{\pi_j/t'_j \in int(t') | t_i \prec t'_j} \pi_j & otherwise \end{cases}$$

Example 24. *We illustrate an example from [Bosc et al., 2011], let consider a relation of schema (make, category), the preferences* $(VW > Ford > Opel)$ *and* $(SUV > roadster > others)$ *and the tuples:*

$t1 = \langle \{1/Opel, 0.8/VW\}, roadster \rangle$

$t2 = \langle Ford, \{1/SUV, 0.7/sedan\} \rangle$

$t3 = \langle \{1/VW, 0.6/Opel\}, roadster \rangle$.

Let us compute $\Pi(t_1)$. *The interpretations of t1 are:* $t_{11} = 1/\langle$ *Opel, roadster* $\rangle$ *and* $t_{12} = 0.8/\langle$ *VW,roadster* $\rangle$, *We detail here the operationhere:*
$\Pi(t_{11} \not\prec t_2)$*=0.7 (corresponding to the interpretation* $\langle$ *Ford, sedan*$\rangle$ *of* t_2*) and* $\Pi(t_{11} \not\prec t_3)$*=0.6 (corresponding to the interpretation* $\langle Opel, roadster \rangle$ *of* t_3*).*
For the second interpretation of t_1, *we get* $\Pi(t_{12} \not\prec t_2) = 1$ *(which corresponds to the interpretation* $\langle$ *Ford, SUV* $\rangle$ *of* t_2 *that is completely possible and does not dominate* t_{12}*) and* $\Pi(t_{12} \not\prec t_3) = 1$ *(which corresponds to the interpretation* $\langle VW, roadster \rangle$

of t3).
Finally: $\Pi(t_1) = max(min(1, min(0.7, 0.6)), min(0.8, min(1, 1))) = 0.8$.

Concerning the dominance computation, we get:

- $\Pi(t_1 \prec t_2) = min(1, 1) = 1$ *which corresponds to the pair 1 /*$\langle Opel, roadster \rangle$ *for* t_1 *and 1 Ford, SUV for* t_2.
- $\Pi(t_1 \prec t_3) = min(1, 1) = 1$ *which corresponds to the pair 1 /*$\langle Opel, roadster \rangle$ *for* t_1 *and 1 VW, roadster for* t_3.

Finally, $N(t_1) = 1 - max(\Pi(t_1 \prec t_2), \Pi(t_1 \prec t_3)) = 1 - max(1, 1) = 0$

2.2.3.3 Stochastic Skyline Queries

Lot of applications request a personal trade-off among all optimal solutions. The skyline queries, are extensively studied as a multi-dimensional criteria analysis. The probabilistic skyline model proposed to retrieve uncertain objects based on skyline probabilities. Meanwhile, skyline probabilities cannot capture the preference of monotonic utility functions. Base on that gap, authors in [Kijima and Ohnishi, 1999] [Lin et al., 2011] proposed a stochastic skyline operator that guarantees providing the minimum set of candidates for the optimal solutions over all possible monotonic multiplicative utility functions.

R_+^d is used to denote the points in R^d with non negative coordinate values. In the discrete case, an uncertain object U consists of a set $\{u_1, \ldots, u_m\}$ of instances (points) in R_+^d where for $1 \leq I \leq m, u_i$ is in R_+^d and occurs with probability $p_{ui}(p_{ui} > 0)$ and $\sum_{i=1}^{m} p_{ui} = 1$.
For a point $x \in R_+^d$, the probability mass $U.cdf(x)$ of U is the sum of the probabilities of the instances in $R((0, \ldots, 0), x)$ where $(0, 0, \ldots, 0)$ is the origin in R^d, that is $U.cdf(x) = \sum u \preceq x, u \in UP_u$. The stochastic dominance is presented as follows:

Definition 2.11 (Stochastic Dominance)**.** *Assume two uncertain objects U and V , U stochastically dominates an object V ,denoted by $U \prec_{sd} V$, if $U.cdf(x) \geq V.cdf(y)$ for any point $x \in Rd+$ and $\exists y \in Rd+$ such that $U.cdf(x) \geq V.cdf(x)$.*

Definition 2.12 (Stochastic Skyline)**.** *Given a set of uncertain objects $\mathcal{U}$, an object $U \in \mathcal{U}$ is a stochastic skyline object if there is no object $V \in \mathcal{U}$, such that $V \prec_{sd} U$. The set of stochastic skyline objects is called the stochastic skyline of $\mathcal{U}$.*

2.2.3.4 Evidential Skyline Queries

The evidential skyline queries was introduced by [Sayda et al., 2014] and enriched in [Sayda et al., 2016b, Sayda et al., 2016a]. Given two objects of an evidential database, authors calculated the belief that each object dominates the other.

Definition 2.13 (Dominance belief degree)**.** *Given a set of objects $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$ defined on a set of attributes $\mathcal{A} = \{a_1, a_2, \ldots, a_d\}$, with $o_i.a_k$ denotes the bba of object o_i w.r.t. attribute a_k. The degree of belief that an object o_i is better than or equal (or strictly better) to another object o_j w.r.t. an attribute a_k is as follows:*

$$bel(o_i.a_k \leq o_j.a_k) = \sum_{A \subseteq \theta_{ak}} (m_{ik}(A) \sum_{B \subseteq \theta_{ak}, A \leq^{\forall} B} m_{jk}(B)) \tag{2.9}$$

Where $A \leq^{\forall} B$ stands for $a \leq b, \forall (a, b) \in A \times B$

$$bel(o_i.a_k < o_j.a_k) = \sum_{A \subseteq \theta_{ak}} (m_{ik}(A) \sum_{B \subseteq \theta_{ak}, A \leq^{\forall} B} m_{jk}(B)) \tag{2.10}$$

Based on this dominance relationship, they proposed the notion of b-dominant skyline, which comprises the objects that are not dominated with some belief threshold b.

Definition 2.14 (b-dominance)**.** *Given two objects $o_i, o_j \in \mathcal{O} : o_i 6 = o_j$ and a belief threshold b, o_i b-dominates o_j denoted by $o_i \succ_b o_j$ if and only if $bel(o_i \succ_b o_j) \geq b$.*

Example 25. *We illustrate an example from [Sayda et al., 2014]. Given a set of weight-loss products, defined over two attributes; weight loss per month and repayment (if the user is not satisfied). Each product may have one or more focal elements w.r.t. each attribute. For example, the weight loss per month of product o1 comprises two focal elements* $\langle\{15, 16, 18\}, 0.1\rangle and \langle\{19, 20\}, 0.9\rangle$. *Thus, the attribute value is either 15, 16 or 18 with mass function 0.1 or one of the values 19 or 20 with mass 0.9.*

Product	Weight loss per month (Kilograms)	Repayement (%)
o_1	$\langle\{15, 16, 18\}, 0.1\rangle, \langle\{19, 20\}, 0.9\rangle$	$\langle 90, 0.3\rangle, \langle\{90, 100\}, 0.7\rangle$
o_2	$\langle 7, 0.7\rangle, \langle\{8, 9\}, 0.3\rangle$	$\langle\{70, 80\}, 0.8\rangle, \langle 80, 0.2\rangle$
o_3	$\langle\{1, 4\}0.1\rangle, \langle 5, 0.9\rangle$	$\langle\{70, 80\}, 0.7\rangle, \langle 100, 0.3\rangle$
o_4	$\langle 10, 0.2\rangle, \langle 12, 0.2\rangle, \langle\{13, 14\}, 0.6\rangle$	$\langle 100, 1\rangle$
o_5	$\langle\{12, 13, 14\}, 0.2\rangle, \langle 17, 0.4\rangle, \langle 19, 0.4\rangle$	$\langle\{20, 30\}, 0.6\rangle, \langle 30, 0.4\rangle$

TABLE 2.7 – dominance beliefs

Table 2.8 shows the dominance belief that each object in lines dominates another object in columns.

Objects	o_1	o_2	o_3	o_4	o_5
o_1	1	1	0.7	0.3	0.92
o_2	0	1	0	0	0
o_3	0	0	1	0	0
o_4	0	1	1	1	0
o_5	0	0	0	0	1

TABLE 2.8 – dominance beliefs

Indeed, o_1 0.9-dominates both o_2 and o_5. However, it does not 0.9-dominate o_4 since $bel(o1 \succ o4) = 0.3 < 0.9$.
To define the notion of evidential skyline, b-dominance relationship needs to be used. For instance, an object is in the evidential skyline if it is not dominated with some threshold. Thus, [Sayda et al., 2014] defined the notion of b-dominant skyline as follows.

Definition 2.15 (b-dominant skyline)**.** *The skyline of* $\mathcal{O}$*, denoted by b-Sky*$_\mathcal{O}$*, comprise those objects in* $\mathcal{O}$ *that are not b-dominated by any other object, i.e., b-Sky*$_\mathcal{O} = \{o_i \in \mathcal{O} / \nexists o_j \in \mathcal{O}, o_j \succ o_i\}$

We continue with the same example of [Sayda et al., 2014] illustrated in Table 2.8. The 0.4-dominant skyline comprises objects o_1 and o_4, since they are not 0.4-dominated by any other object, while the 0.2 contains only o_1 as o_4 is 0.2 dominated by o_1.

Conclusion

Imperfection, be it imprecision or uncertainty, should be incroporated in today's applications and information systems to provide a complete and accurate model of the real world. During this chapter we pinpointed the utility of using possibility theory to handle one facet of uncertainty, i.e, epistemic uncertainty. This theory will be used to deal with uncertainty in the RDF data context.

On the other hand, skyline preference queries has became an important issue in database research for extracting the most interesting objects from a multi-dimensional dataset. The skyline query processing is used in many applications that demand multi-criteria decision making without using cumulative functions in order to extract the most interesting objects based on users' preferences. An object is considered as interesting, if it is dominated by no other object in all the evaluation criteria. The extensive use of the skyline operator is mainly due the model's simplicity and its applicability on multi-criteria decision decision making with reference to user preferences. We presented a reminder about skyline preference queries, the basic concepts and the main introduced algorithms. Finally, a brief refresher about extending skyline queries over uncertain data was detailed.
In the coming chapter, we are interested in extending the skyline queries over RDF data, particularly when they are associatedwith trust measures [Hartig, 2009a].

Part II

Contributions

Chapter 3

Trust Skyline Model: Semantics and Experimentations

Contents

Introduction 65

3.1 Trust-Skyline model 66

3.1.1 Trust Dominance 67

3.1.2 Trust-Skyline semantics 71

3.1.3 Trust-Skyline computation 72

3.2 Experimental Evaluation 76

3.2.1 Experimental Setup 77

3.2.2 Impact of the trust measure variation 77

3.2.3 Impact of the size of data set 79

3.2.4 Number of used properties in the skyline query 79

3.3 Statistical methods-driven Analysis 79

3.3.1 Trust-Skyline list Analysis 81

3.3.2 Alpha v.s. the distribution of Trust values 83

3.3.3 Central Tendency measures 84

3.3.4 Measures of spread: Quartile measure 85

3.3.5 Trust dependence 85

3.4 Analysis Experimental 86

3.4.1 Experimental Setup 86

3.4.2 Impact of trust threshold variation and Central Tendency measures 87

3.4.3 Impact of trust threshold variation and Quartile measures 88

Conclusion **89**

Introduction

As mentioned in the previous chapter, variety of sources on the Web affects the reliability of collected data. To rate information trustworthiness, new metrics were introduced in RDF representation model such in [Hartig, 2009a, Tomaszuk et al., 2012, Fionda and Greco, 2015].
On the other hand, to reason in presence of trust information, there is a need of new approaches to query RDF data. In this thesis project, we are interested in preference-based queries [Chomicki, 2002, Kiessling, 2002, Chomicki, 2011, Chomicki et al., 2013] to extract and filter the huge amount of data contained in databases. An important kind of preference queries is introduced in [Börzsönyi et al., 2001a], the Skyline operator. It returns the most interesting objects based on the Pareto dominance operator and according to user-defined criteria.
For instance, extending skyline queries over RDF data is proposed in litereture in the work of [Chen et al., 2011]. Authors proposed applying skyline model over RDF data. However, this work did not tackle imperfections of the RDF data model. Although, literature is abundant on works about skyline queries over uncertain relational data, such as [Jiang et al., 2012], [Bosc et al., 2011], [Zhang et al., 2013], the extraction and filter of imperfect RDF data usiong skyline queries has not been advocated, in this proposal we try to tackle this issue.
During this chapter, we present two main contributions as part of our work, we recall that the aim of this thesis project is to model and query imperfect RDF data. Imperfection signifies here the lack of trustworthiness and certainty of RDF data.
First contribution: The first part of our work concerns adapting the skyline operator to trust-weighted RDF data (T-RDF). We begin with redefining the dominance between such type of data. Although, the skyline operator produces a binary result (0/1) in the context of certain data, it produces a degree of dominance in case of trust RDF data. Therefore, we provide semantics for the trust-skyline operator, i.e., the set of objects that are dominated by no other object with a degree greater than a user-defined measure that we denote α. Up to our knowledge the extension of skyline queries over RDF data to extract and filter the massive amount of resources among the data sets have only been advocated in the proposal of [Chen et al., 2011]. Meanwhile, this work did not consider the trust thresholds and deals only with the basic definition of the RDF data model.
To compute the trust-skyline, we propose a new algorithm denoted TRDF-Skyline. This algorithm is based on the checked and proved properties over the redefined dominance relationship. In addition, we compare the proposed solution to the naive method for computing the trust-skyline set, and also with an SQL query that we implemented for which data is stored in a relational table of quadruples. The experiments show interesting and

encouraging results.
Second contribution: It consists in analysing the results presented in the first part and published in [Amna et al., 2017b]. First of all, we distinguish between the trust-Skyline resulting list. While the points with trust measures less than the user-defined threshold α enter directly to the trust-Skyline list, they are not considered as interesting points. This is due to the non-check of Pareto dominance operator. Therefore, we opt for separating the two categories of points (i.e points with less trust and points with greater trust).
To analyze the results we opt for using statistical methods to investigate the trust measures dependence. Indeed we use the central tendency measures (mean and median standards), and the measures of spread (Quartile measure) for such analysis. We checked the impact of the trust measure α and the list of the generated trust measures on the resulting trust-Skyline list. Finally we present the experiments that showed interesting results.
The rest of the chapter is organized as follows: In Section 3.1, we introduce our new model, the Trust-Skyline and we show how it operates over weighted RDF data. Then, Section 3.2 illustrate our experimental study. For Section 3.3, we analyze our trust-Skyline list. Furthermore, we present the proposed methods to check the dependence between trust measures. Finally, we illustrate our experimental study in Section 3.4.

The contributions presented in this chapter are published in:

- **The 19th International Conference on Enterprise Information Systems [Amna et al., 2017a]**
- **The extension of [Amna et al., 2017a] is published as a book chapter in Springer Lecture Notes in Business Information Processing** [Amna et al., 2017b]

3.1 Trust-Skyline model

The only existing work about skyline queries over RDF data is the proposal of [Chen et al., 2011]. The authors introduced a skyline model over RDF data stored in a multiple relations way. Although absence of works on skyline queries over trust-weighted RDF data, the literature is abundant on works about extending skyline queries over uncertain relational data, such as [Bosc et al., 2011, Jiang et al., 2012, Zhang et al., 2013]. The extension of skyline queries over uncertain RDF has not been advocated in the literature, in our work we attempt fill the gap.
During this section, we aim to extend the classic model of skyline queries to deal with the Trust RDF model. Indeed, we introduce the *Trust-Skyline* model in which we extract the

set of objects (resources) that are dominated by no other object (resource) according to user-defined criteria (trust measure).
In the context of trust model the RDF triple $< s, p, o >$ is extended to an RDF quadruple $< s, p, o, t >$ where the trust value t represents the trustworthiness of the triple $< s, p, o >$ and takes its values in the interval [-1, 1] [Hartig, 2009a]. We denote the quadruple a "SPOT".

Definition 3.1 (RDF SPOT)**.** *An RDF SPOT X is a quadruple $< s, p, o, t >$, where o is a value of a property (predicate) p related to a subject (resource) s, with a trust measure t. We denote the triple $< s, p, o >$ by X^*.*

Given that an RDF SPOT describes a unique property of a subject, we need to propose new semantics to compare two resources. Indeed such comparison needs to consider all common properties. Therefore, we propose the notion of *point*. We define a point as the set of SPOTs related to a unique subject (resource). In a multi-dimensional domain, a point is characterized by several dimensions, as well as an RDF subject, characterized by several predicates (properties).

Definition 3.2 (Trust RDF point)**.** *A trust RDF point P_t is the set of SPOTs related to a unique subject s having m properties p_i, the values o_i and the trusts t_i such that $1 \leq i \leq m$. We denote P_t^* the set of SPOs, i.e., the quadruple SPOT without trust measures, related to the subject s.*

Example 26. *Let us consider the RDF data set illustrated in table 3.1. Four hotels are considered, having each one two properties ("HasPrice" and "HasDistance"). The quadruple $< h_1, HasPrice, 21, 0.8 >$ is an RDF SPOT which we denote X. Accordingly, X^* is the RDF SPO $< h_1, HasPrice, 21 >$. Let the pattern matching P be the set of SPOTs related to h_1. $P = \{< h_1, HasPrice, 21, 0.8 >, < h_1, HasDistance, 110, 0.9 >\}$. P^* is the set of SPOs related to h_1:*

$$P^* = \{< h_1, HasPrice, 21 >, < h_1, HasDistance, 110 >\}.$$

As the skyline query is based on the dominance relation (see Definition 2.7), we need to start with redefining this relation in the context of T-RDF data.

3.1.1 Trust Dominance

The dominance relation is based on the comparison between the properties' values (see Definition 2.7). In the context of certain data, the comparison of two values produces a binary

TABLE 3.1 – Example of trust RDF data

Subject	Predicate	Object	Trust
h_1	HasPrice	21	0.8
h_1	HasDistance	110	0.9
h_2	HasPrice	31	0.5
h_2	HasDistance	120	0.2
h_3	HasPrice	21	0.2
h_3	HasDistance	130	0.7
h_4	HasPrice	31	0.8
h_4	HasDistance	60	0.3

result (0/1). Meanwhile, for the uncertain case, comparison is not binary. It is rather quantified with a degree. For example, assume two RDF SPOTs, $p_1 = (H_1,'distance', 50, 0.6)$ and $p_2 = (H_2,'distance', 40, 0.4)$. The distance that separates hotels H_1 and H_2 from the beach are 50 with a trust value 0.6, and 40 with a trust value 0.4, respectively. However, we are not able to conclude that $H_1.distance$ is greater than $H_2.distance$. We only quantify the trustworthiness of this comparison as presented in definition 3.3.

Definition 3.3 (Comparison trust)**.** *Assume two properties' values a and b, having the trusts $Trust(a)$ and $Trust(b)$, respectively. Let λ be an arbitrary value, where, $\lambda \leq -1$. The trust degree of the comparison between a and b, denoted by* ***Trust(a ϕ b)****, such that the operator $\phi \in \{\leq, <, \geq, >, =, \neq\}$, is defined as follows:*

$$\textbf{\textit{Trust(a } } \phi \textbf{\textit{ b)}} = \begin{cases} min(Trust(a), Trust(b)) & if\ a \phi b\ is\ true \\ \lambda & else \end{cases}$$

In the rest of the chapter, λ is arbitrary fixed to λ=-1.

At this stage, we need to define the dominance between two RDF triples. To consider uncertain context of data, we need to adapt the Pareto dominance presented in definition 2.7. Indeed, we changed the logical connectors $\bigwedge$ and $\bigvee$, that represents the conjunction and disjunction of two binary comparisons, to the minimum and maximum operators, respectively, in order to deal with uncertain data.

Definition 3.4 (Trust dominance degree)**.** *Let P and Q be two subjects having n properties p_i and q_i, respectively with $1 \leq i \leq n$. The degree of dominance between P and Q, denoted by $d(Q \succ P)$ is defined as follows*[2]*:*

$$d(Q \succ P) = min(\min_{1 \leq i \leq n} Trust(q_i \leq p_i), \max_{1 \leq i \leq n} Trust(q_i < p_i))$$

2. We assume in this work, that the smaller value, the more preferable

Example 27. *Let us consider two hotels h_1 and h_2, having the properties price and distance. We illustrate four cases in order to test all scenarios between h_1 and h_2 as presented in Table 3.2.*

TABLE 3.2 – Example of hotels properties.

Hotels	case 1		case 2		case 3		case 4	
	price	distance	price	distance	price	distance	price	distance
h_1	20(0.2)	100(0.4)	20(0.6)	80(0.7)	20(0.3)	100(0.5)	20(0.3)	70(0.5)
h_2	30(0.3)	110(0.5)	25(0.3)	70(0.1)	20(0.4)	100(0.6)	25(0.4)	70(0.5)

Here after, we present the computation of the Trust-Skyline over those four cases:

- *case 1: To compute $d(h_1 \succ h_2)$, we have:* $20 \leq 30$ *is true thus the trust of the comparison is min(0.2, 0.3)=0.2 and* $100 \leq 110$ *is true, thus the trust of the comparison in min (0.4, 0.5)= 0.4. Moreover,* $20 < 30$ *thus, the trust of the comparison is min(0.2, 0.3)=0.2, also for* $100 < 110$*, the trust value is min (0.4, 0.5)= 0.4. To conclude, we have $d(h_1 \succ h_2)$= min(min(0.2,0.4), max(0.2,0.4))=0.2*
- *case 2:*
  $d(h_1 \succ h_2)$= min(min(0.3,-1), max(0.3,-1))=-1
- *case 3:*
  $d(h_1 \succ h_2)$= min(min(0.3,0.5), max(-1,-1))=-1
- *case 4:*
  $d(h_1 \succ h_2)$= min(min(0.3,0.5), max(0.3,-1))=0.3

We introduce the concept of *point trust* in order to simplify the computation of dominance degree between two RDF triples.

Definition 3.5 (Point trust)**.** *Let an RDF point P having m properties p_i, where, $1 \leq i \leq m$. Each property is associated with a trust t_i. The point trust, denoted by $P.t^-$ is the minimum trust degree among all its properties.*

$$P.t^- = \min_{1 \leq i \leq n} (p_i.t)$$

The aim from using the notion of *point trust* is to simplify the computation of the trust dominance as illustrated in Proposition 3.1.

Proposition 3.1. *Given two points P and Q having the trusts $Q.t^-$ and $P.t^-$.*

$$d(Q \succ P) = \begin{cases} min(Q.t^-, P.t^-) & if\ Q* \succ P* \\ -1 & else \end{cases}$$

Proof 1. *For two RDF points P and Q, $d(Q \succ P)$ is the minimum between two measures;* $\min_{1\leq i\leq n} Trust(q_i \leq p_i)$, *and* $\max_{1\leq i\leq n} Trust(q_i < p_i)$.

For the first measure($\min_{1\leq i\leq n} Trust(q_i \leq p_i)$*), we have two scenarios:*

- *if there exists any property i where $q_i \leq p_i$ is false, then the measure is equal to -1.*
- *if for each property i, $q_i \leq p_i$ is true, then the measure is equal to the smallest trust among all properties of P and Q. It is equal to $min(Q.t^-, P.t^-)$*

Concerning the second measure($\max_{1\leq i\leq n} Trust(q_i < p_i)$*), we have two possible scenarios:*

- *if there exists at least one property i such that $q_i < p_i$, then the measure is equal to the greatest value between the trusts of all comparisons $q_i < p_i$ that return true.*
- *if there is no property i such that $q_i \leq p_i$ is true, then the measure is equal to -1.*

We combined the scenarios above in Table 3.3. The only case that returns a value different from λ occurs when $\forall i, q_i \leq p_i$ and $\exists i, q_i < p_i$. In this case, we return:

$min(\min_{1\leq i\leq n} Trust(q_i \leq p_i), \max_{1\leq i\leq n} Trust(q_i < p_i))$. *We are sure that* $\min_{1\leq i\leq n} Trust(q_i \leq p_i)$ *is less or equal than* $\max_{1\leq i\leq n} Trust(q_i < p_i)$.

Hence, the measure $\min_{1\leq i\leq n} Trust(q_i \leq p_i)$ *returns the minimal trust among all properties' values of P and Q (see definition 3.3), that is simply $min(P.t^-, Q.t^-)$. The case*

Table 3.3 – Dominance degree function.

	$\exists i, q_i < p_i$	$\nexists i, q_i < p_i$
$\forall i, q_i \leq p_i$	$min(\min_{1\leq i\leq n} Trust(q_i \leq p_i), \max_{1\leq i\leq n} Trust(q_i < p_i))$	-1
$\exists i, q_i > p_i$	-1	-1

where $\forall i, q_i \leq p_i$ and $\exists i, q_i < p_i$, corresponds in fact to $Q^* \succ P^*$. In this case, $d(Q \succ P)$ is equal to the smallest trust among all properties' trusts of P and Q (see definitions 3.4 and 3.5), which is the minimum value between $Q.t^-$ and $P.t^-$.

Example 28. *If we take the same cases shown in example 27 using proposition 3.1, we obtain:*

- *case 1: $h_1^* \succ h_2^*$ then $d(h_1 \succ h_2) = min(Q.t^-, P.t^-) = min(0.2, 0.3) = 0.2$*
- *case 2: $h_1^* \not\succ h_2^*$ then $d(h_1 \succ h_2)$=-1*
- *case 3: $h_1^* \not\succ h_2^*$ then $d(h_1 \succ h_2)$=-1*
- *case 4: $h_1^* \succ h_2^*$ then $d(h_1 \succ h_2) = min(Q.t^-, P.t^-) = min(0.3, 0.4) = 0.3$*

Note that we obtain the same results as in the example 27.

Proposition 3.2. *The trust dominance is transitive. Given two RDF triples P and Q, and a threshold $\alpha \in [-1, 1]$*

$$\textit{if } d(R \succ Q) > \alpha \textit{ and } d(Q \succ P) > \alpha; \textit{ Then } d(R \succ P) > \alpha$$

Proof 2. $d(R \succ Q) > \alpha$ *and* $d(Q \succ P) > \alpha$ *(1)*
$d(R \succ Q) = min(R.t^-, Q.t^-)$ *and* $d(Q \succ P) = min(Q.t^-, P.t^-)$ *(2)*
(1) and (2) imply $min(R.t^-, Q.t^-) > \alpha$ *and* $min(Q.t^-, P.t^-) > \alpha$ *(3)*
(3) implies $min(R.t^-, Q.t^-, P.t^-) > \alpha$ *(4)*
(4) implies $d(R \succ P) > \alpha$

Proposition 3.3. *The trust dominance is asymmetric. Given two RDF triples* P *and* Q*, and a threshold* $\alpha \in [-1, 1]$

$$d(Q \succ P) > \alpha \Rightarrow d(P \succ Q) = -1 < \alpha$$

Proof 3. *According to Proposition 3.1, we have:*

$$d(Q \succ P) > \alpha \Rightarrow Q^* \succ P^*$$
$$Q^* \succ P^* \Rightarrow P^* \not\succ Q^*$$
$$P^* \not\succ Q^* \Rightarrow d(P \succ Q) = -1$$

3.1.2 Trust-Skyline semantics

In [Börzsönyi et al., 2001a], skyline is defined as the set of database objects dominated by no other object. In such perfect context, dominance is binary. However, in context of trust RDF data, dominance is a quantified relation rather than a boolean one. Therefore, the skyline is defined as the set of points dominated by no other point according to some trust value α.

Definition 3.6 (Trust-Skyline)**.** *Let* $\alpha \in [-1, 1]$ *be a user defined threshold. The T-Skyline of a data set* $\mathcal{H}$*, denoted by* $T - Sky^\alpha$*, contains each point* P *in* $\mathcal{H}$ *such there is no point* Q *that dominates* P *with a trust degree greater than* α*.*

$$T - sky^\alpha = \{P \in \mathcal{H} / \nexists Q \in \mathcal{H}, d(Q \succ P) \geq \alpha\}$$

Example 29. *Let us consider the example of five hotels, with two properties each one (Price and Distance), see Table 3.4 . For each property we specify a trust degree to describe the data trustworthiness.*

We detail below the computation of the T-Skyline of the RDF data set presented in Table 3.4 when α *is fixed to 0.1 (*$\alpha = 0.1$*).*

TABLE 3.4 – Example of hotels candidate list of T-Sky.

Hotel	Price	Distance
h_1	23 (0.5)	5 (0.3)
h_2	50 (0.2)	4 (0.6)
h_3	50 (0.7)	3 (0.5)
h_4	40 (0.1)	1 (0.3)
h_5	50 (0.6)	2 (0.4)

— *h_1 dominates h_2 with a degree equals to 0.2 ($\geq \alpha$). Given that, the trust-dominance is asymetric h_2 does not dominate h_1. We conclude that h_2 could not integrate the skyline. Hence, h_2 is pruned from the Trust-Skyline set.*

— *$d(h_1 \succ h_3) = 0.3$, thus h_3 is also pruned.*

— *$d(h_1 \succ h_4) = -1$ and $d(h_4 \succ h_1) = -1$. We make no pruning.*

— *$d(h_1 \succ h_5) = 0.3$. h_5 is pruned.*

We conclude that the Trust-Skyline list includes h_1 and h_4 which are dominated by no other point.

Remark. One can observe that some points could enter directly the trust-Skyline without being compared with other ones. Indeed, if the trust of a point P (see definition 7) is less than the trust threshold α, then we conclude directly that P is in the skyline because we are sure there is no other point Q able to dominate it with a degree greater than α, even if $Q^* \succ P^*$. We detail the analysis of this remark in Section 3.3.

Proposition 3.4. *Given a data set $\mathcal{H}$ and its T-Skyline $T - Sky^\alpha$ and a point $P \in \mathcal{H}$. If $P.t^- < \alpha$ then $P \in T - Sky^\alpha$.*

Proof 4. *If $P.t^- < \alpha$, then we are sure there exists no point $Q \in \mathcal{H}$ such that $d(Q \succ P) \geq \alpha$ since $d(Q \succ P)$ is equal to $min(Q.t^-, P.t^-)$ or -1. In this case, (there is no $Q \in \mathcal{H}$ such that $d(Q \succ P) \geq \alpha$), we are sure that $P \in T - Sky^\alpha$.*

3.1.3 Trust-Skyline computation

In order to compute the Trust-Skyline, we propose two algorithms; the Naive T-Skyline algorithm and the TRDF-Skyline algorithm. In addition, for the evaluation purpose, we discuss also a non-native solution that consists in representing trust-RDF data in relational table and then extract the trust-Skyline using an SQL query. In particular, we show how the Trust-Skyline can be implemented on a relational database using an SQL query. We illustrate in Table 3.5 the stored functions used in that query.

3.1.3.1 SQL-like Trust-Skyline queries

Trust RDF data could be stored in a relational table as quadruplets (see for instance Table 3.1). To implement our SQL-based method, we created a table named T_{RDF} with four attributes: s, p, o and t that stands for subject, predicate, object and trust respectively. As the comparison between objects is not binary, we implemented two comparison operators to specifically deal with the imperfect context; the *less* and *lessorequal* functions which return a degree between -1 and 1. The two functions are described in Table 3.5.

TABLE 3.5 – Used functions Meaning.

less(v_1,v_2)	returns the least trust degree between two points $v1$ and $v2$ if $v1 < v2$
lessorequal(v_1,v_2)	returns the least trust degree between two points $v1$ and $v2$ if $v1 \leq v2$

Below, we present the SQL query syntax that returns the Trust-Skyline of the table T_{RDF} according to the trust threshold α. This latter selects each subject A such there is no subject B that dominates it. Meanwhile, B dominates A if two conditions are satisfied. First, there exists no predicate (property) whose value for A is better or equal than its value for B. The second, it exists at least one predicate whose value for A is strictly better than its value for B. We recall here that the smaller values, the more preferable are, thus, the use of functions *less* and *lessorequal*.

```
SELECT DISTINCT s FROM T_RDF A WHERE NOT EXISTS(
 SELECT * FROM T_RDF B WHERE B.s!= A.s AND NOT EXISTS(
  SELECT * FROM T_RDF C WHERE C.s= A.s AND NOT EXISTS(
  SELECT * FROM T_RDF D WHERE D.s= B.s AND C.p= D.p
        AND lessorequal(D.o, D.t, C.o, C.t)>=&α))
 AND EXISTS
 (SELECT * FROM T_RDF E WHERE E.s=A.s AND EXISTS(
 SELECT * FROM T_RDF F WHERE F.s=B.s AND F.p= E.p
        AND less( F.o, F.t, E .o, E.t)>=&α)));
```

Below, we provide the SQL code of the two functions **less** and **lessorequal**.

```
CREATE OR REPLACE FUNCTION less(v1 IN NUMBER, t1 NUMBER, v2 IN NUMBER, t2
NUMBER) RETURN NUMBER IS
inferior NUMBER := -1;
BEGIN
  IF (v1<v2) THEN
    inferior := least(t1,t2);
  END IF;
  RETURN inferior;
END;
CREATE OR REPLACE FUNCTION lessorequal(v1 IN NUMBER, t1 NUMBER, v2 IN NUMBER,
t2 NUMBER) RETURN NUMBER IS
inferior NUMBER := -1;
BEGIN
  IF (v1<=v2) THEN
    inferior := least(t1,t2);
  END IF;
  RETURN inferior;
END;
```

3.1.3.2 Naive T-Skyline Algorithm

The Block Nested Loop (BNL) algorithm was presented in many works about the Skyline operator such as [Börzsönyi et al., 2001a] [Deepak et al., 2011]. The main idea of the BNL algorithm is that it reads repeatedly the set of tuples. Once a point is read from the input list, it is compared to all the tuples of the candidate list which makes the execution time exponential. This is considered as a naive approach to solve the problem of extracting the trust skyline.

In our algorithm *Naive T-Skyline* (see Algorithm 1) we propose an optimization of the naive method in order to filter the input list because for large input size, the number of compared tuples will be T × T. For instance, based on proposition 3.4, we optimized the naive method by adding directly all points having trust values less or equal to the threshold α. These points could not be dominated with a degree greater than α. Note that Complexity of this method is $O(n^2)$.

We continue with the same example presented on Table 3.4 we proceed on modifying α to check its impact on the Skyline resulting list. If we fix α to 0.1, to be part of the T-Skyline, each point should be dominated by no other point with a degree greater than 0.1. The T-Skyline resulting list is $\{h_1,h_4\}$. If we increase α, points having trust measures inferior than α are in the T-skyline because no other point could dominate them over this degree. The trust degree α has a big impact on computing the T-Skyline list. Hence, we used this measure in our Naive T-Skyline algorithm to make an earlier filtering of the candidate list.

Algorithm 1: The Naive T-Skyline Algorithm

```
   Input: n RDF triples;
   Output: T − Sky Trust-Skyline points;
1  begin
2      foreach point P ∈ DB do
3          SKY ← true
4          if P.t⁻ < α then
5              Add P to TSky
6          else
7              foreach point Q ∈ DB such that Q ≠ P do
8                  if (dominates(Q,P))>= α  /*Using dominates function*/ then
9                      SKY ← false
10                     Break
11         if SKY = true then
12             Add P to TSky
13     return T − Sky
```

3.1.3.3 TRDF-Skyline Algorithm

We introduce a new algorithm, denoted TRDF-Skyline which uses, in addition to the optimization (based on property 3.4) in the naive T-Skyline method, a second optimization based on the transitivity property (proposition 3.2). Hence, we are not obliged to compare all the pairs of points. If a point A dominates a point B, then B is eliminated and A is added to the trust skyline. Then, when we find that a point C dominates A, then A is eliminated and C is added to the skyline. Indeed, the comparison between C and B is useless because B cannot dominate A. We even know that C dominates B thanks to the transitivity property ($A \succ B$ and $C \succ A$ implies $C \succ B$).

Even if the complexity of this method is $O(n)$, we are sure we pruned useless points thanks

to the transitivity property. The TRDF-Skyline algorithm is presented as follows.

Algorithm 2: The TRDF-Skyline Algorithm

```
   Input: n RDF triples;
   Output: TSky Trust-Skyline points;
1  begin
2      foreach point P ∈ DB do
3          if P.t⁻ < α then
4          Add P to TSky
5          else
6              inSKY ← true
7              foreach point Q ∈ TSky, Q ≠ P do
8                  if dominates(P,Q) >= α   /*Using dominates function*/ then
9                      Remove (Q) from TSky
10                     else
11                         if dominates(Q,P) >= α then
12                         inSKY ← false
13                         Break
14         if inSKY:=true then
15             Add P to TSky
16     return T − Sky
```

3.2 Experimental Evaluation

In this section, we evaluate the methods introduced in subsection 3.1.3, which are considered as exact methods. Therefore, evaluation doesn't deal with the output quality. Indeed, the produced skyline is exactly the same regardless of the used method. Consequently, the experiments we led were about (1) the performance of the methods (time execution), and (2) the size of the skyline (size control). For each measure, we varied (1) the trust threshold, (2) the size of the target database and (3) the number of properties in the skyline query. The aim is to understand the impacts of these parameters on the execution time and the size of the skyline.

3.2.1 Experimental Setup

Due to the lack of trust RDF databases, we generated synthetic data sets according to the parameters in table 3.6. For each experiment, we vary one parameter and set the others to the default values (referred in the above-mentioned table). Note that data are generated following the uniform law. Indeed, we used the triple storage approach [Sakr and Al-Naymat, 2009], extended to a quadruple format to deal with the trust measure. The data generator and the algorithms 2 and 1 were implemented in Java. The SQL query were implemented under Oracle 11g. Stored functions were implemented using PL/SQL. All experiments were conducted under Windows 7 on a 2.10 GHz Intel Core Duo processor computer with 4GB of RAM.

Symbol	Parameter	Default
P	Number of properties	6
D	Number of quadruples	300 K
T	Size of T-Skyline data	-
α	Trust measure	0.2
X	Time execution (ms)	-

TABLE 3.6 – Parameters under investigation.

3.2.2 Impact of the trust measure variation

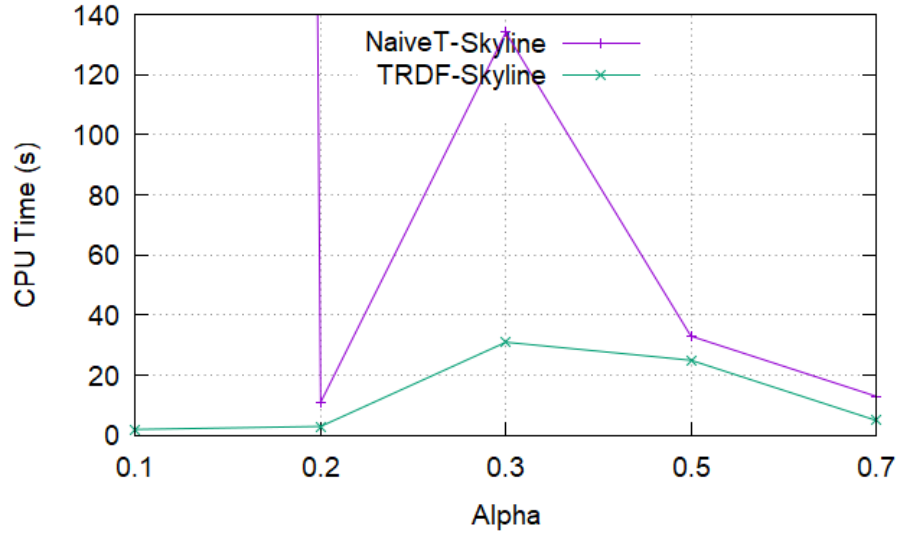
The Trust-Skyline operator is defined as the set of points dominated by no other point according to a trust threshold α. In this experiment, we varied α in order to measure its impact on the execution time and on the size of the trust skyline set, as shown in Figure 3.1.

When α has a great value, both methods perform quickly (Figure 3.1a). This is due to the fact that points' trusts are more probably less than α, and thus enter directly to the skyline without processing. In this case, using Proposition 3.4 leads to considerably pruning the search space. Size of the trust skyline (see figure 3.1b) is important, because it is rare to check a dominance between two points, according to a threshold whose value is important. If there is rare dominance between points, then we obtain a great size of Trust-Skyline set.

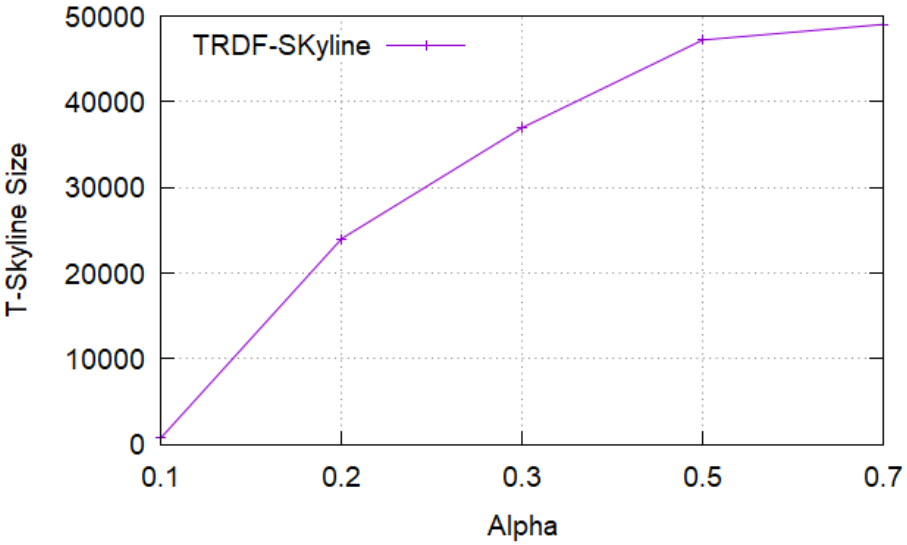
On the other hand, when α has a small value, several points are dominated and thus do not enter to the skyline. Hence, Trust-Skyline size is small in this case. The Naive T-Skyline do not benefit from the Proposition 3.4 pruning method, and execution time

is very important. Meanwhile, for TRDF-skyline, execution time is very acceptable, since pruning based on the transitivity property is always efficient and doesn't depend from α.

Concerning the SQL query, with a database of 6k tuples the execution time exceeds $12.10^3 ms$. SQL query is logically costly since we do not use a native environment, and we don't optimize computation as in the other methods. For instance, the SQL query compares all the points' pairs in the database.



(a) Effect on time execution



(b) Effect on skyline size

FIGURE 3.1 – Effect of α on skyline computation

3.2.3 Impact of the size of data set

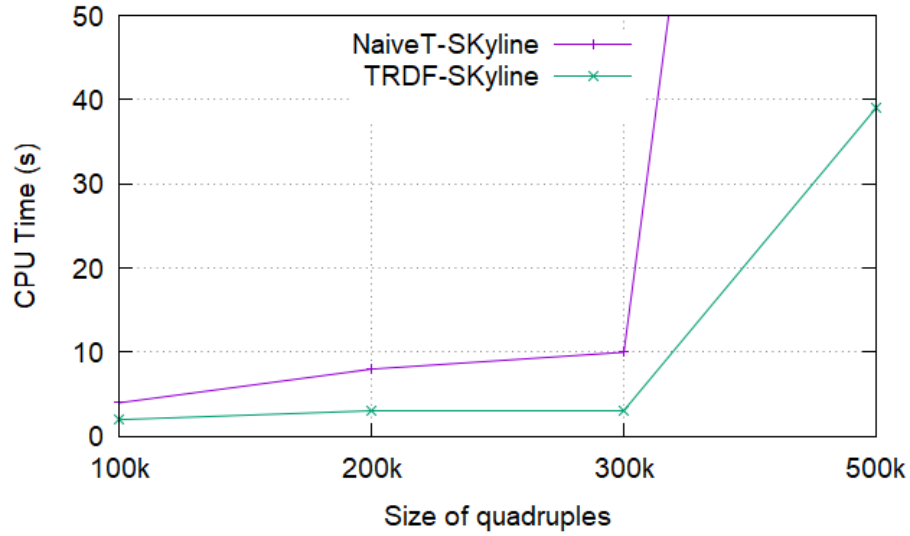
During this experiment, we tackle the impact of the data size on the performance and size of the Trust-Skyline. For instance, we varied the input data size from 100k, to 500k tuples as shown in Figure 3.2. Figure 3.2 depicts a comparison between the two algorithms. As shown in the previous experiment, TRDF-Skyline algorithm outperforms the Naive Trust-Skyline algorithm. When the size of data reaches 300k, the execution time of the Naive T-Skyline becomes exponential. At 500k it exceeded 223s. However, for the TRDF-Skyline it does not exceed 50s. The execution time of the SQL query is the worst, it is very high over a size greater than 12K.
We think that distributed methods are recommended, when data set are very huge. Since Pareto dominance is transitive, data set could be divided. Hence, extraction of trust-Skyline is computed in parallel, and then a smart fusion of the results is operated. An interesting perspective of this work is to model and implement distributed methods to extract trust-Skyline.

3.2.4 Number of used properties in the skyline query

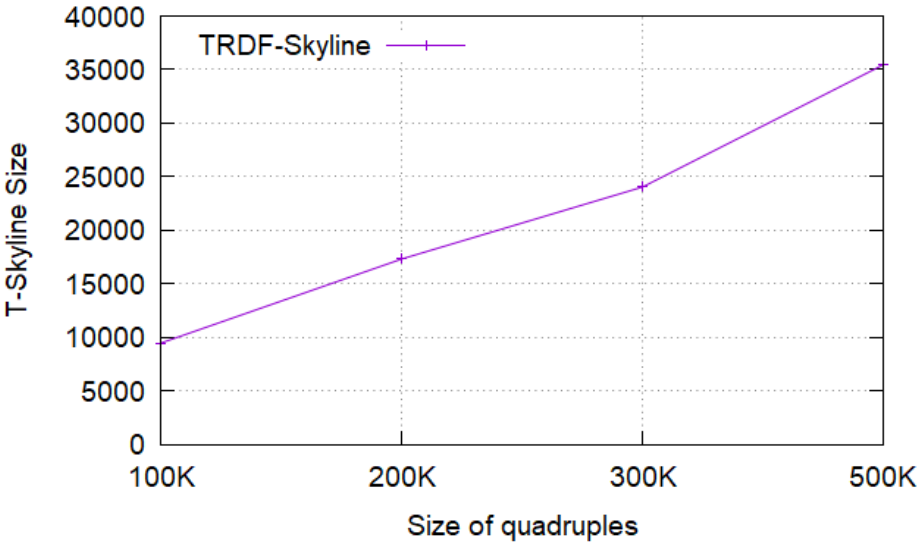
In this experiment, we tackle the impact of properties (criteria) number in the skyline query over the result computation. To this end, we increased the number of skyline query's properties as illustrated in Figure 3.3. The trust skyline size increased with the increase of P (see figure 3.3b), due to the fact that subject has more chance to be not dominated when comparison copes with a high number of criteria. Figure 3.3a illustrates again the performance of the TRDF-Skyline that takes advantage from the transitivity property. The naive algorithm, even worst, performs better than the SQL query which compares all pairs of points, without pruning using the property 3.4.

3.3 Statistical methods-driven Analysis

In this section we are interested in analyzing the results presented in our work [Amna et al., 2017a]. In [Amna et al., 2017a], we proposed to extend the classic model of skyline queries to cope with the trust RDF model. We introduced the *Trust-Skyline* in which we extract the set of most interesting resources in a trust RDF dataset. As the skyline query is based on the Pareto dominance operator, we redefined the dominance relationship in the context of trust RDF data. Then, we proposed an appropriate semantics of the *trust-Skyline* in which we extract the set of most interesting resources in a trust RDF dataset.



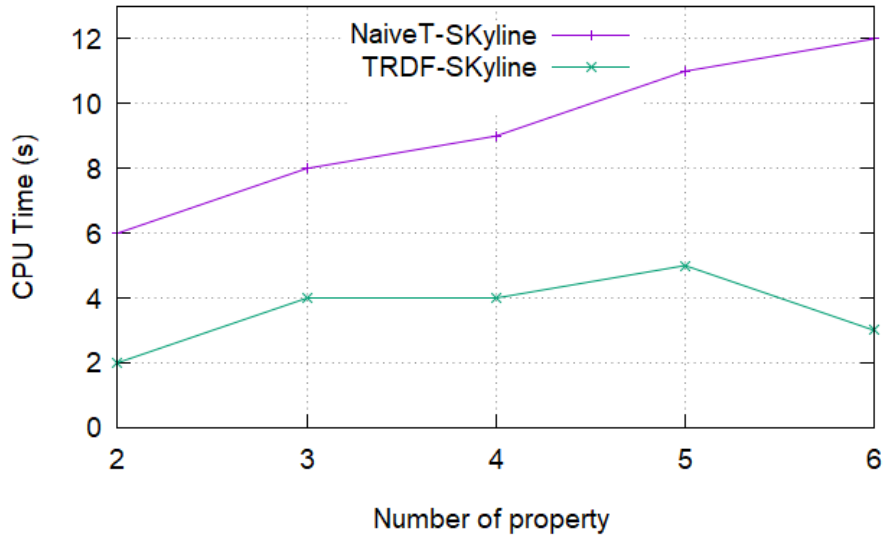
(a) Effect on time execution



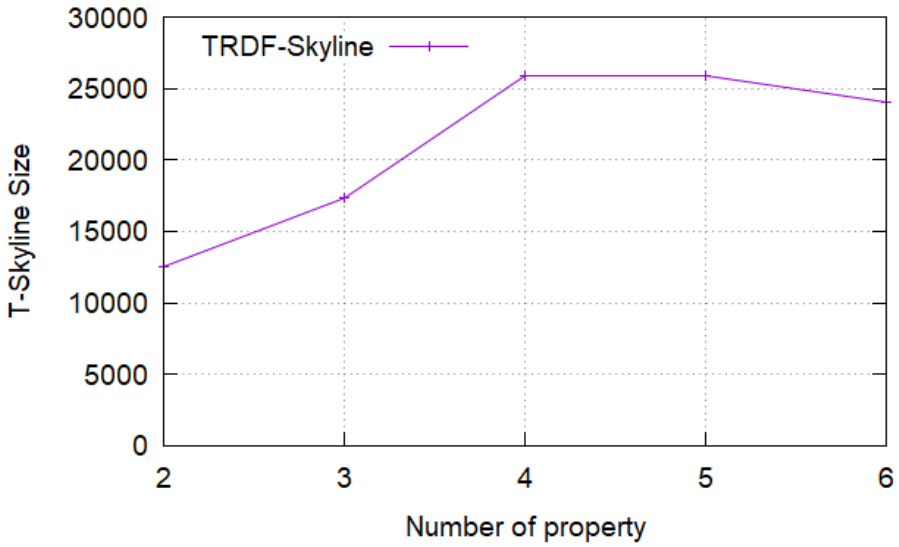
(b) Effect on skyline size

FIGURE 3.2 – Effect of data size on skyline computation

To analyze the results, we distinguish between the points of the computed trust-Skyline. Indeed, points with trust measures less than the user-defined threshold α enter directly to the trust-Skyline list. Therefore, we don't considered as interesting points. This is due to the non-check of Pareto dominance relationship. Thus, we opt for separating the two categories of points. Indeed, we make use of statistical methods to investigate the trust measures dependence. We use the central tendency measures (Mean and Median standards), and the measures of spread (Quartile measure) for such analysis.
For the experiments, we proceed on checking the impact of the trust measure α and the



(a) Effect on time execution



(b) Effect on skyline size

FIGURE 3.3 – Effect of criteria number on skyline computation

list of the generated trust measures on the resulting trust-Skyline list.

3.3.1 Trust-Skyline list Analysis

In [Amna et al., 2017a] a set of properties have been defined and used to optimize the computation of the T-Skyline list. The first property used is adding directly all points having trust measure less or equal to the threshold α. These points could not be dominated

with a degree greater than α. Note that the complexity of this method is $O(n^2)$.
The second method is a combination between the first one and the use of transitivity property. Indeed, there is no need to compare all the pairs of points.
Based on the used methods to compute the T-Skyline list, we can make an analysis of the resulting list. The first category is the list of points having trust values less than α (i.e., points with less trust), the second one is the points that are Pareto-dominated by no other point (i.e., points with more trust).

3.3.1.1 T-Skyline points with less trust

We recall the property 3.4 (see Section 3.1) in which we let the points with trust measure less than α enter directly to the T-Skyline list without processing:
Given a data set D and its T-Skyline $T - Sky^{\alpha}$ and a point $P \in D$. If $P.t^{-} < \alpha$ then $P \in T - Sky^{\alpha}$.
This category of points could be dominated by no other point with a trust degree greater than α. The search space is considerably pruned because the dominance check between points is reduced, we denote this list of points $T - Sky_{<\alpha}$. $T - Sky_{<\alpha}$ points are added to T-Skyline list without considering their property values. Therefore, the list of points is not interested in the T-Skyline final list due to the non-check of Pareto-dominance.
The greater α, the greater the size of the $T - Sky_{<\alpha}$ points. The lower the trust value, the lower the $T - Sky_{<\alpha}$ set and the greater the uncertainty.

3.3.1.2 T-Skyline points with more trust

We denote the list of points added to the T-Skyline list after Pareto-dominance check $T - Sky_{>\alpha}$. The list of $T - Sky_{>\alpha}$ are the most interesting points in the result list since they have the best values among all the properties.
Figure 3.4 presents the difference between the two sets of points. For instance, the greater α, the greater the size of the $T - Sky_{<\alpha}$ points. The lower the trust value, the lower the $T - Sky_{<\alpha}$ set. However, the less value of α the great $T - Sky_{>\alpha}$ and the less skyline size set bacause several points are dominated and so do not enter to the skyline.

3.3.1.3 Behavior of Alpha

Great values
We recall the definition about the trust of a given point (see Definition 3.5). Assume an

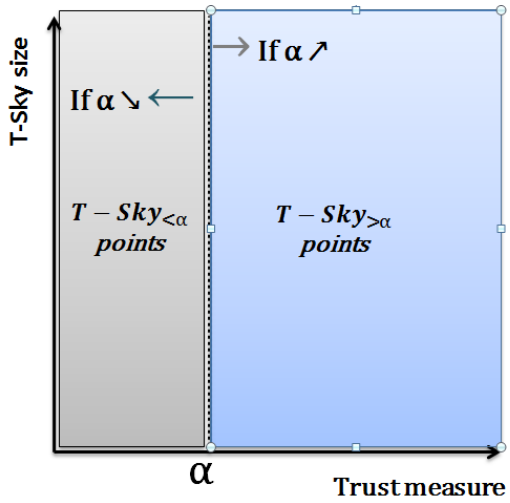


FIGURE 3.4 – Trust-Skyline points analysis.

RDF point P with m properties p_i. The trust of a point, denoted by $P.t^-$ is the minimum trust degree among all its properties and presented as follows:

$$P.t^- = \min_{1 \leq i \leq n} (p_i.t)$$

For a great value of α, the size of the T-Skyline becames greater and the algorithm performs better. This is due to the fact that points' trusts are more probably less than α, and thus enter directly to the skyline without processing.
In this case, thanks to Proposition 3.4 , the search space is considerably pruned. The size of the trust skyline is important, because the dominance check between two points is rare, according to a threshold whose value is important.

Small values

For a small size of α, several points are dominated and thus do not enter to the skyline. Therefore, the skyline size is small in this case. The Naive T-Skyline algorithm do not benefit from the Proposition 3.4. Hence, pruning method, and execution time is very important. However, for TRDF-Skyline algorithm, execution time is very acceptable, since pruning based on the transitivity property is always efficient and doesn't depend from α.

3.3.2 Alpha v.s. the distribution of Trust values

In this section, we aim to study the dependence between α and the rest of trust values $P.t^-$ of the set of T-Skyline candidate points. There are many methods to describe the variability in data set, in this section we choose to use statistical methods to organize and

summarize a set of trust scores, which are: The *measure of Spread* and the *measure of Central Tendency* [Gravetter and Wallnau, 2000].
The purpose of central tendency (or central location) is to determine the single value that identifies the center of the distribution and best represents the entire set of scores. The three standard measures of central tendency are the mean, the mode, and the median, we give more explanations in subsection 3.3.3. While, a measure of spread (or of dispersion), is used to describe the variability in a sample or population. From the set of spread measures, we choose to use the quartile method as introduced in subsection 3.3.4.

3.3.3 Central Tendency measures

Summarizing the set of trust values can help us understand the data, especially when the dataset is large. There exist multiple measures of Central Tendency that summarize the data into a single value.
There are three main measures of central tendency: The mean, the median and the mode. Each of these measures describes a different indication of the typical or central value in the distribution.

- The mean: This measure is the sum of the value of each observation in a dataset divided by the number of observations. This is also known as the arithmetic average. Looking at the trust values below:

  0.2, 0.3, 0.4, 0.2, 0.3, 0.8, 0.3

  The mean is calculated by adding together all the trust values and dividing by the number of observations (Mean of trusts= (0.2+0.3+0.4+0.2+0.3+0.8+0.3)/ (7)) which equals 0.357.

- The median: is the middle value in distribution when the values are arranged in ascending or descending order. The median divides the distribution in half (there are 50% of observations on either side of the median value). In a distribution with an odd number of observations, the median value is the middle value. We illustrate the same example given in the mean measure, after sorting the trust thresholds: 0.2, 0.2, 0.3, 0.3, 0.3, 0.4, 0.8. The median is 0.3.

- The mode: It is the most commonly occurring value in a distribution. Consider same example shown before, the table 3.7 shows a simple frequency distribution of trust measures. The most commonly occurring trust value is 0.3. Therefore the mode of this distribution is 0.3.

Table 3.7 – Trust frequency distribution.

Trust	Frequency
0.2	2
0.3	3
0.4	1
0.8	1

3.3.4 Measures of spread: Quartile measure

In order to study the dependence between α and the trust set points $P.t^-$ we choose to use the quartile statistical measure.
Quartiles tell us about the spread of a data set by breaking the data set into quarters.

Example 30. *For example, consider the price of 11 hotels below, which have been ordered from the lowest to the highest value:*

6, 7, 15, 36, 39, 41, 41, 43, 43, 47, 49

- *The first quartile (Q1) is the value of the middle of the first set, for which 25% of the values are least then it and 75% are greater.*
- *The third quartile (Q3) is the value of the middle of the second set, in which 75% of the values are less than Q3 and 25% are greater.*

The first quartile (Q1) lies between the 3rd and 5th price values and the third quartile (Q3) between the 8th and 10th price values, Hence:

- *First quartile (Q1) = 15*
- *Second quartile (Q2) = 43*

3.3.5 Trust dependence

The use of the quartile measure allow subdividing the generated data set into four parts. Indeed, we can have a global view about the distribution of trust $P.t^-$ of the global set. Such information could be useful for the user in order to know which value of α could fit the best with his query. We illustrate the example below of a set of points with a finite

interval of trust each set:

A brief analysis of the table 3.8, let as know that if we choose α greater or equal to 0.5,

Number of points	Interval of trust
54 points	[0.5 ;1]
26 points	[0.3 ;0.5[
16 points	[0.2 ;0.3[
4 points	[0.1 ;0.2[

TABLE 3.8 – Example of trust distribution.

we will have 46 points (46%) of the database will be included directly in the T-Skyline set. However with an α equals to 0.1, only 4 points will be directly included in the T-Skyline list, etc.

The trust assignment allows users to classify resources as trusted or not. For instance, the user is allowed to give more preferences to get the final result such the size of the returned T-skyline points (according to the trust distribution). In the case where the user wants the most trusted resources, the value of α needs to be not great and vise versa.

Another statistical measure is the interquartile range that describes the difference between the third quartile (Q3) and the first quartile (Q1), telling us about the range of the middle half of the values in the distribution. For the example 30, the interquartile is 43-15=28.

3.4 Analysis Experimental

In this section, we evaluate the dependence between the set of trust measures and the user-defined trust value α. Consequently, the experiments we led were about (1) the performance of the methods (time execution), and (2) the size of the skyline (controle the size). For each measure, we varied (1) the trust threshold, (2) the size of the database. The aim is to understand the effects of these parameters on the execution time and the resulted Trust-Skyline.

3.4.1 Experimental Setup

We have presented in Section 3.1.3 (published in [Amna et al., 2017a]) several methods to compute the Trust-Skyline model. Due to the lack of trust RDF databases, we generated synthetic data sets according to the parameters presented in table 3.9. For each

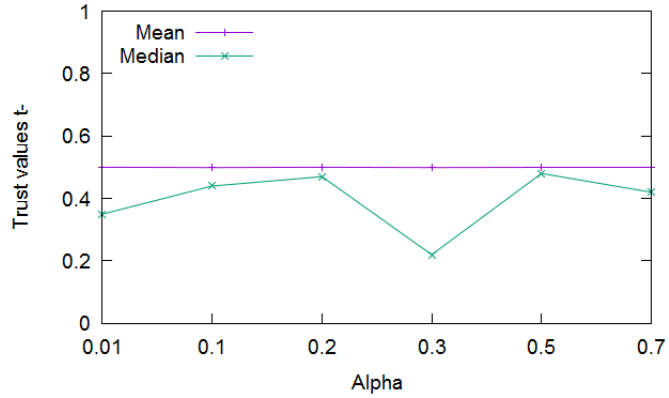
experiment, we vary one parameter and set the others to the default values (see Table 3.9). We extended the 3Store model presented in [Harris and Gibbins, 2003b] [Sakr and Al-Naymat, 2010] [Sakr and Al-Naymat, 2009], using RDF quadruple store representation to deal with the trust measure. The data generator and the algorithms presented in [Amna et al., 2017a] were implemented in Java. Stored functions were implemented using PL/SQL. All the experiments were conducted under Windows 7 on an Intel(R) Core(TM) i5-2410M CPU with 4GB of RAM.

Symbol	Parameter	Default
D	Number of quadruples	300 K
X	Size of T-Skyline data	-
α	Trust measure	0.2
T	Time execution (ms)	-

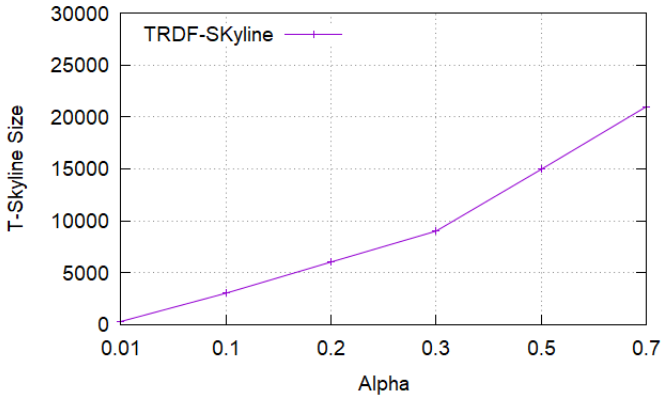
TABLE 3.9 – Parameters under investigation.

3.4.2 Impact of trust threshold variation and Central Tendency measures

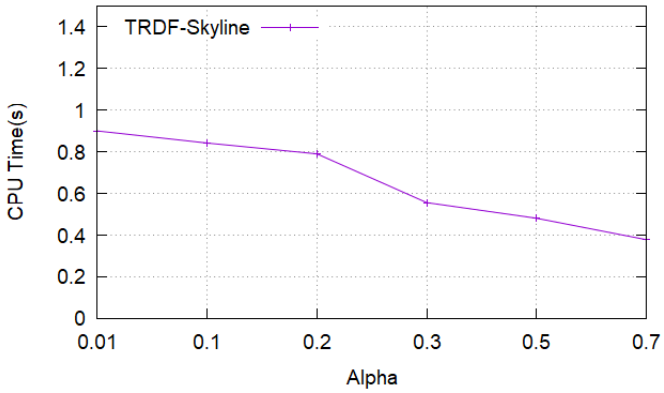
As we presented previously, this work is an analysis of the T-Skyline model (see Section 3.1) and an extension of [Amna et al., 2017a], in which we have introduced the Trust-Skyline as the set of points dominated by no other point according to a trust threshold α. In this experiment, we aim to investigate the relation between the user-defined threshold α and the distribution of the set of trust measures $P.t^{-}$. We start by studying the impact of central tendency measures (we opt for using the mean and the median) and α on the execution time and on the trust skyline, as shown in figure 3.5.
When α has a great value, the TRDF-Skyline algorithm performs quickly. This is due to the fact that $T-Sky_{<\alpha}$ points(defined previously in section 3.3.1) are huge, and thus enter directly to the skyline without processing with respect to the property 3.4. Indeed, the search space is considerably pruned. Size of the trust skyline (figure 3.5a) is important, because it is rare to check a dominance between two points, according to a threshold whose value is important. If we check the central tendency measures, for great values of α we found that the median and mean values are less than α. Therefore, the T-Skyline list is huge, and the algorithm performs well. If there is rare dominance-check between points, then we obtain a great number of skyline points.
On the other hand, when α has a small value, The mod and median of the set of points are greater than *alpha* as shown in figure 3.5. Indeed, the set of $T-Sky_{>\alpha}$ points (defined previously in section 3.3.1) is huge, and several points are dominated and so do not enter to the skyline. That is why skyline size is small in this case.



(a) Dependence of trust measures



(b) Effect on skyline size X



(c) Effect on time execution, T

FIGURE 3.5 – Alpha and Central Tendency measures

3.4.3 Impact of trust threshold variation and Quartile measures

As we presented previously in section 3.3.2, we selected quartile measure to study the spread of trust values. Quartiles are less affected by outliers or skewed data set than the

equivalent measures of mean and standard deviation. This approach sets upper and lower hit selection thresholds based on number of interquartile ranges above or below $Q1$ and $Q3$ quartiles. In our experiments we studied the $Q1$ and $Q3$ quartiles as shown in figure 3.6.
For low values of α, the quartile range (difference between Q3 and Q1) is greater than α, indeed $T - Sky_{<\alpha}$ list is small. The Q1 quartile represents the most likely points to enter the T-Skyline list without dominance-check, and $Q3$ represents the most likely points to enter this list after dominance-check. We can conclude that the user can use those two measure to specify the value of α, if he wants to have an aggressive pruning of the candidate list, the value of *alpha* needs to be small and the value of the points in $Q1$ quartile. Another important point is, if the user wants to only let the most interesting points, that we represented with $T - Sky_{>\alpha}$, enter the T-Skyline, the value of *alpha* needs to be low.

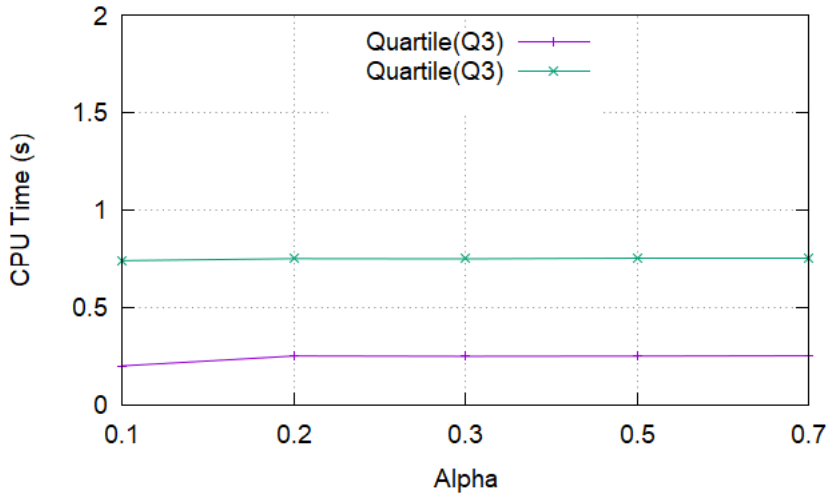


FIGURE 3.6 – Alpha and Quartile measure

Conclusion

During this chapter, we proposed an extension of the skyline operator to the context of trust RDF data. Indeed, a new variant of the skyline, called the trust-Skyline, is introduced. Therefore, semantics of Pareto dominance relationship and (traditional) skyline operator were revised.
To compute the trust-Skyline, we implemented two algorithms that consider the trust thresholds to compute the trust-Skyline set. The Naive T-Skyline algorithm uses points' trust degrees to make an earlier filtering of data. For the TRDF-Skyline algorithm, it is

optimized based on the transitivity property of the trust dominance operator. Furthermore, we presented an SQL query to show how Trust-Skyline can be implemented on a relational database system.
The conducted experiments showed the efficiency of the TRDF-Skyline algorithm. The naive T-Skyline method is acceptable in case the input data size is not huge and the trust threshold is medium or high. Meanwhile, the SQL query showed very limited performance.
As a second contribution, we proposed to analyze the Trust-Skyline resulting list. To this end, statistical methods were used to analyze the dependence between α and the set of generated trust thresholds.
To study the trust-Skyline list, we used two statistical methods that take into account the trust measures to compute the trust-Skyline set. The first method is the central tendency measures, we specifically used the mean and median standards. And the measures of spread that tell us about the spread of a data set by breaking the data set into quarters.
Our experiments showed the impact of the user-defined trust measure α on the T-Skyline list. The T-Skyline list is huge if the trust threshold is medium or high. However, small values of α restrict the entering to this list, only interesting points can be selected.
For the next chapter, we tackle uncertain RDF data in the setting of the rich possibility theory. Indeed, we add a possibility measure to each RDF triple (association between a subject, predicate and object) to model the possibility of such association. Indeed, a new framework is proposed to query possibilistic RDF data denoted Pi-SPARQL. Furthermore, we extend skyline queries to possibilistic RDF data and we present the appropriate experimental evaluations.

Chapter 4

Possibilistic RDF Data

Contents

Introduction 92

4.1 Possibilistic RDF model 93

4.1.1 Possibilistic RDF databases 93

4.1.2 Possibilistic RDF graph 93

4.2 A SPARQL-like language for possibilistic RDF data . . . 96

4.2.1 Possibility-aware Basic Graph Pattern Matching 97

4.2.2 Enhanced SPARQL algebra 99

4.2.3 SPARQL Extension for Possibility distributions Requirements 102

4.3 Possibilistic Skyline over RDF data 105

4.3.1 Comparison of two possibility distributions 105

4.3.2 Possibilistic dominance on RDF data 107

4.3.3 Possibilistic skyline on RDF data 110

4.4 Possibilistic skyline computation 110

4.4.1 Experimental Evaluation 111

4.4.2 Experimental Setup 113

4.4.3 Size of the Skyline on RDF Data 113

4.4.4 Performance and Scalability 114

Conclusion 117

Introduction

A central task in decision-making is to consider the uncertainty associated with data. Thus, a profusion of research work have been conducted to model uncertain RDF data such as in [Huang and Liu, 2009,Lian and Chen, 2011]. Our idea is to deal with uncertain RDF data using possibility theory [Zadeh, 1978], which is a non-classical theory of uncertainty. It constitutes an alternative to capture different kinds of imperfection, such as imprecision, total ignorance, and partial ignorance that are not faithfully representable in probability theory [Zimmermann, 1985].
Thus, we integrated in the structure of RDF data a possibility measure for each subject-property-object triple to reflect the user opinion about the truth of a statement. The possibility measure can be considered as a way to express a source reliability.
On the other hand, the great advance in Web-based information extraction provides an increase of automatic construction of semantic knowledge bases. Indeed, an increase in variety and volume of RDF data format. Furthermore, the process of extraction provides RDF knowledge bases pervaded with some imperfections. In this chapter, we opt for skyline queries [Börzsönyi et al., 2001a, Jiang et al., 2012, Zhang et al., 2013, Chomicki et al., 2013] to exploit the massive amount of imperfect RDF data. Skyline queries have been extended over Graph Data such in [Zou et al., 2010, Zheng et al., 2014]. They aim to make multi-objective decisions over complex data. Given such a multi-criteria preference set, the system should be able to identify all potentially interesting data records according to user preferences [Chomicki, 2002, Chomicki, 2011].
Our main contributions in this Chapter are summarized as follows:

— Modelling uncertain RDF data through possibility theory. We introduce comparison operators between possibility distributions to allow dominance computations in the RDF data set context.

— Extending the skyline operator over possibilistic RDF data. The starting from [Chen et al., 2011], we rethought the dominance operator between two possibilistic RDF points.

The rest of the chapter is organized as follows: Section 4.1 introduces the possibilistic model of uncertain RDF data. Section 4.2 presents the general framework for querying possibilistic RDF data. Then, Section 4.3 extends skyline queries over possibilistic RDF data. Finally, Section 4.4.1 shows the skyline computation and the experimental evaluations.
The contributions presented in this chapter are published in the:
Intenational Journal of Approximate Reasoning [Amna et al., 2018b]
International Conference on Tools with Artificial Intelligence (ICTAI) [Amna et al., 2018a]

4.1 Possibilistic RDF model

In this section, we propose a possibilistic model for uncertain RDF data. In regular database system, the answer to a query q is a crisp set of records. Each record has a binary degree of membership to the result set; 0 if false, and 1 if true. However, in the possibilistic case, the database system returns several possible answers and assigns to each answer a possibility degree that represents the plausibility of such answer to be the solution of a user defined query. In the coming subsections, we introduce the possibilistic database [Prade, 1984, Bosc et al., 2011] model for RDF data. Then, we represent RDF data by possibilistic graph, knowing that RDF data could be equivalently viewed from a graph perspective. We define it as follows.

4.1.1 Possibilistic RDF databases

Due to the uncertainty pervading today's data, databases may have some uncertain predicates or properties. Authors in [Prade, 1984] presented the first version of possibilistic database where the subset of the possible values of an attribute of a given object is supposed to be crisp. Indeed, based on Zadeh theory [Zadeh, 1978], in the case the value of an attribute is unknown, several values being possible. Nevertheless there is some knowledge indicating that among possible values, some of them are more possible than others.
A possibilistic RDF database D is a database of resources whose description may be uncertain. Uncertainty in such databases is modelled and managed through the rich Possibility Theory.

Definition 4.1 (Possibilistic RDF database)**.** *A possibilistic RDF database D is a set of possibilistic triples. Each triple t is associated with a possibility value $\Pi(t)$ indicating its ability to occur. In the possibility distribution we extend the RDF triple $< S, P, O >$ to a quadruple $< S, P, O, \Pi >$ where O is a value of a predicate P related to a subject S, with a possibility degree Π.*

4.1.2 Possibilistic RDF graph

Definition 4.2 (Possibilistic RDF Graph Data)**.** *A possibilistic RDF graph data $\tilde{G}^P = (V, E, \Pi)$ is a graph represented by the triple $(V(G), E(G), \Pi(G))$, where:*

1. *$V(G) = \{V_1 \dots V_m\}$ represents a finite set of vertices,*
2. *$E(G) = \{e_1 \dots e_n\}$ is a finite set of edges that connect pairs of vertices,*

3. $\Pi(G)$ *is the possibility associated to each triple of* G*.*

Example 31. *Assume we collected information about hotels: price, distance (from the beach), address etc., from another source* $R2$ *different than* $R1$ *presented in Example 3. We notice a mismatch between the information gathered from* $R1$ *and the one from* $R2$*.*

1. $(http://hotel.org/H_1, < hasName >, "Gaia")$;
2. $(http://hotel.org/H_1, < hasPrice >, 6)$;
3. $(http://hotel.org/H_1, < hasDistance >, 4)$;
4. ...
5. $(http://hotel.org/H_2, < hasName >, "Via - dei - valeri")$;
6. $(http://hotel.org/H_2, < hasPrice >, 5)$;
7. $(http://hotel.org/H_2, < hasDistance >, 4)$;
8. ...

Accordingly, H_1 has different name, price and distance than the one cited previously which report inconsistent data about H_1. Thus, the uncertainty/inconsistency in the collected RDF data can result from either the extraction accuracy or the reliability of sources. In order to model such uncertainty, we assign each possible triple a possibility degree to indicate its plausibility to occur. Figure 4.1 shows an example of possibilistic RDF database using a graph representation. Below, in Table 4.1 we present the possibilistic RDF database corresponding to Figure 4.1. It is a database of quadruples (subject, preedicate, object,Π). For short, we present only four quadruples related to the hotel H_1.

TABLE 4.1 – Example of possibilistic RDF database

Subject	Predicate	Object	Possibility(Π)
H_1	hasName	Philippos	0.7
H_1	hasName	Gaia	1
H_1	hasPrice	6	1
H_1	hasPrice	5	0.8
...	...	...	...

To query possibilistic RDF data, we need to extend SPARQL to be possibility-aware query language. Indeed, users as well as software agents have to be able to access and use possibility measures associated to triples. Many works focus on improving the performance of answering SPARQL queries such in [Abadi et al., 2007, Hartig, 2009a, Sidirourgos et al., 2008].

Hartig. O [Hartig, 2009a] proposed a trust model that associates RDF statements with trust values and extended the SPARQL semantics to access these trust values in (tSPARQL). Huang et al. [Huang and Liu, 2009] proposed a general framework for supporting

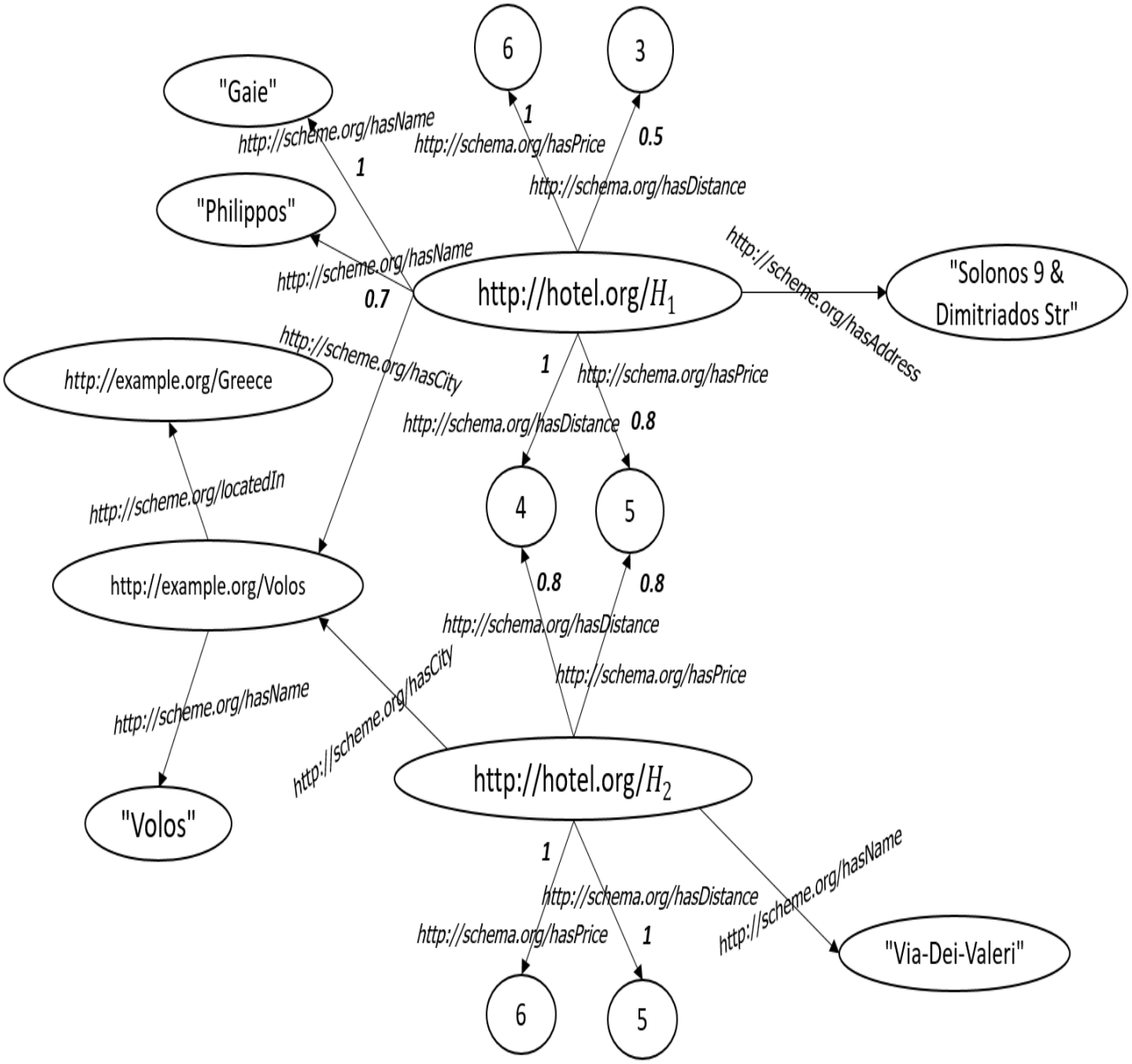


FIGURE 4.1 – Graph representation of uncertain RDF data.

SPARQL queries on probabilistic data model. However as we mentioned previously, we chose possibility theory to deal with uncertain RDF data as it constitutes an alternative to capture different levels of imperfection (uncertainty, imprecision, ignorance, partial ignorance). Therefore, existing techniques for querying uncertain RDF data are not directly applicable to our possibilistic RDF model.

4.2 A SPARQL-like language for possibilistic RDF data

Authors in [Eric and Andy, 2008] presented the official syntax of SPARQL. The proposed operators to construct a graph pattern expression are OPTIONAL, UNION, FILTER, and concatenation via a point symbol (.) (operator AND in traditional algebraic formalism), The syntax also considers { } to group patterns, and some implicit rules of precedence and association. We will give more details about the use of SPARQL graph pattern expression in the coming sections.
Given that the RDF query language SPARQL is of declarative nature, we need to extend SPARQL to a possibility-aware query language. We denote it $Pi-$SPARQL. It extends SPARQL to describe possibilistic requirements and to access the ability of query solutions to occur and to match with a definite query.
Assume the query $Q2$: "Return the price and distance (from the beach) of hotel H_1 ordered by their possibility measure $?\Pi_i$". With Pi-SPARQL users can additionally access possibility measures that represent the possibility of matching subgraphs to occur. This additional expressivity allows for making queries such as Q_2. We introduce a new algebra operator, the **POSSIBLE AS** clause to the query language. The POSSIBLE AS clause permits access to the possibility distributions. Thus, possibility measures can became part of the solutions and can be associated with parts of the query pattern.
For query Q_2, it additionally asks for the occurrence possibility of the price and distance of H_1. Below the Pi-SPARQL representation of Q_2, (we give more details about POSSIBLE AS clause in Section 4.2.3.2):

1. Prefix $ab :< http://hotel.org/example >$
2. SELECT ?distance ?price ?Pi
3. WHERE {H_1 ab:hasDistance ?distance (q1)
4. H_1 ab:hasPrice ?price (q2)
5. POSSIBLE AS ?Pi
6. }
7. ORDER BY ?Pi

FIGURE 4.2 – Pi-SPARQL representation of query Q2

4.2.1 Possibility-aware Basic Graph Pattern Matching

The core of each SPARQL query is a Basic Graph Pattern (BGP) (see Section 1.3.3 for more details).

Example 32. *We illustrate the example of Q2 (see Fig. 4.2) in the certain case. It consists of the conjunction of two atomic queries or triple patterns:*
$Q2 = q_1 \wedge q_2$*:* $q_1(H_1, hasDistance, ?distance)$ *and* $q_2(H_1, hasPrice, ?price)$.
The BGP corresponding to query Q_2 *is modeled in Fig. 4.3 and the result of evaluating the BGP G against the query* Q_2 *is illustrated in Table 4.2.*

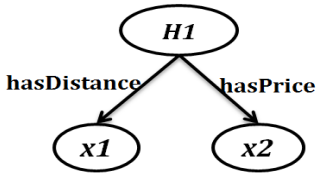


FIGURE 4.3 – Basic Graph pattern G

TABLE 4.2 – The results of evaluating the BGP G against the query Q_2.

$q_1.?distance$	$q_2.?price$
3	5
3	6
4	5
4	6

In order to define the *possibility-aware solution mapping*, we need to introduce some terminology related to semantics of SPARQL graph pattern expressions. As defined in the specification of SPARQL [Eric and Andy, 2008], solutions are defined in the context of BGP matching where each solution basically represents a matching subgraph in the queried RDF graph G.
Intuitively, we define a possibilistic solution mapping (for certain case, see Section 1.3.4]) $\tilde{\mu^p}$ that represents a solution to the BGP G. $\tilde{\mu^p}$ is one matching subgraph; the possibility degree of this solution mapping must represent the possibility of the subgraph to occur. Knowing that, the possibility value can be calculated by a possibility aggregation function (Definition. 4.4).

Definition 4.3 (Possibilistic solution mapping)**.** *Let* $\tilde{\mu^p}$ *be a pair* (μ, Π) *representing of a solution mapping* μ *associated with a possibility degree* Π.

We denote the cardinality of $\tilde{\mu^p}$ in a multiset D of possibilistic solution mappings with $card_D(\tilde{\mu^p})$.

Every solution mapping that is a solution to a BGP represents one matching subgraph. The possibility of a subgraph is an aggregation of the possibility distributions of its triples that are specified by a possibility function. Intuitively, the possibility of the subgraph can be represented by a possibility measure that is calculated from the possibility distributions of its triple using a possibility aggregation function.

Definition 4.4 (Possibility aggregation function)**.** *Given a possibilistic RDF graph $\tilde{G}^P$, the possibility aggregation function Π_a assigns to $\tilde{G}^P$ an aggregated possibility degree $\Pi_a(\tilde{G}^P)$ that represents the possibility of $\tilde{G}^P$. The possibility is calculated using* min *or* max *operators corresponding respectively to conjunctive and disjunctive events.*

Definition 4.5. *Let B be a BGP, Let $\tilde{G^P} = (G, \Pi)$ be a possibilistic RDF graph. The possibilistic solution mapping $\tilde{\mu^p}$ is a solution for B in $\tilde{G^P}$ if there is an RDF instance mapping σ such that:*
(1) $\mu(\sigma(B))$ is a subgraph of G
(2) μ is a mapping for the query variables in B and,
(3) $\Pi = \Pi_a(\tilde{G^P})$ is the aggregated possibility distribution of the possibilistic RDF graph $\tilde{G^P} = (\mu(\sigma(B)), \Pi)$ calculated using the possibility aggregating function Π_a.
For each solution μ for B $card_D\hat{A}(\tilde{\mu^p})$ is the number of distinct RDF instance mapping σ given that $(\mu(\sigma(B))$ is a subgraph of G.

Example 33. *Let us consider the example of the query in Fig. 4.2. When we apply the BGP in line 5 to our possibilistic RDF graph in Fig. 4.1 we find two matching subgraphs resulting in the four solutions shown in Table 4.3. To explain intuitively this result, we*

TABLE 4.3 – The results of evaluating the BGP G against the RDF graph

S		$?Pi$
$?distance$	$?price$	
3	5	0.8
3	6	1
4	5	1
4	6	1

detail the operation here after. Following the work of [Simon et al., 2018], we used the max operator to compute the possibility measure of the triple $< H_1, 3, 5 >$. Indeed, from the joint possibility distribution of two variables $\Pi(3, 5)$, the marginal possibility distributions is computed by projection.

$$\Pi(3,5) = \max(\Pi(3), \Pi(5)) = \max(0.5, 0.8) = 0.8$$

The solution S_1 maps ?distance to 3 and ?price to 5; S_2 maps ?distance to 3 and ?price to 6; S_3 maps ?distance to 4 and ?price to 5 and S_4 maps ?distance to 4 and ?price to 6;

After defining the notion of solutions in the context of Pi-SPARQL, in the following subsection we take a closer look at Pi-SPARQL query processing and query evaluation. Apart from BGPs, the SPARQL specification introduces other graph patterns CGPs. CGPs supports all complex graph pattern features. Indeed, it extends BGPs with further traditional relational operations (projection, union, difference, optional (left-outer-join) and filter) [Angles et al., 2017]. During query evaluation, CGPs are represented by algebra operators which operate on multisets of solution mappings. To access possibility distributions, Pi-SPARQL needs new types of operators. To this end, Pi-SPARQL redefines the conventional SPARQL algebra operators to operate on multisets of possibilistic solution mappings.

4.2.2 Enhanced SPARQL algebra

In the following subsections, Pi-SPARQL redefines the operators of SPARQL algebra. For a precise redefinition, we introduce the following symbols (inspired from the corresponding symbols in [Eric and Andy, 2008]). With $card_{\tilde{\Omega}}(\tilde{\mu^p})$ denote the cardinality of the possibilistiy solution mapping $\tilde{\mu^p}$ in a multiset of possibilistic solution mappings $\tilde{\Omega}$.

4.2.2.1 Join(Θ)

A join operator (see Section 1.3.5.2 for certain case) that operates on possibilistic solution mappings has to consider the possibility measure Π while merging solutions. To get a precise redefinition of the *Join* operator, we followed the corresponding definition for SPARQL query evaluation in [Eric and Andy, 2008].

Definition 4.6. *Let $\tilde{\Omega}_1$ and $\tilde{\Omega}_2$ be multisets of possibilistic solution mappings. We define:*

$$Join(\tilde{\Omega}_1, \tilde{\Omega}_2) = \{\ merge(\tilde{\mu_1^p}, \tilde{\mu_2^p}) \mid \tilde{\mu_1^p} = (\mu_1, \Pi) \in \tilde{\Omega}_1 \wedge \tilde{\mu_2^p} = (\mu_2, \Pi) \in \tilde{\Omega}_2 \text{ and } \tilde{\mu_1^p} \wedge \tilde{\mu_2^p} \text{ are compatible } \}$$

$$card_{Join(\tilde{\Omega}_1,\tilde{\Omega}_2)}(\tilde{\mu^p}) = \sum_{\substack{\tilde{\mu_1^p}\in\tilde{\Omega}_1 \\ \tilde{\mu_2^p}\in\tilde{\Omega}_2}} \begin{cases} card_{\tilde{\Omega}_1}(\tilde{\mu_1^p}) * card_{\tilde{\Omega}_2}(\tilde{\mu_2^p}) \\ \quad \textbf{if } \tilde{\mu^p} = (\mu, \Pi) \textbf{ with} \\ \quad \Pi = \Pi_a(\tilde{\mu_1^p}, \tilde{\mu_2^p}) \textbf{ and} \\ \quad \mu = merge(\mu_1, \mu_2) \\ 0 \quad \textbf{else} \end{cases}$$

Example 34. *Let us consider the following query Q3: Return the plausible minimum price and distance (from the beach) values of hotel h_1. The Pi-SPARQL query corresponding to query Q3 is as follows:*

```
1. Prefix ab :< http : //hotel.org/example >
2. SELECT (MIN(?distance) AS ?mindistance) (MIN(?price) AS ?minprice) ?Pi
3. WHERE {?hotel ab:hasDistance ?distance
4.        ?hotel ab:hasPrice ?price
5.         POSSIBLE AS ?Pi
6.         }
7. ORDER BY ?Pi
```

The group graph pattern in query Q3 groups two BGPs that ask for the minimum of price and distance of H_1. The result of the query is shown in Fig. 4.4 where we found two merged mapping solutions.

To explain intuitively this result, we detail the operation here after. We found two solu-

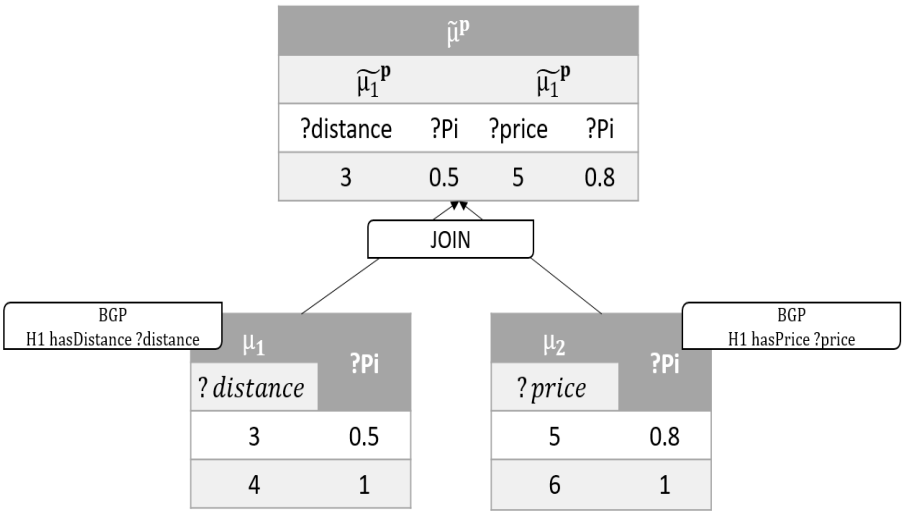


FIGURE 4.4 – Join operator(Θ)

tions for our sample graph in Fig. 4.1. After joining the two solutions, we get the result illustrated in the upper figure. We recall here that we look for the minimum values of the two properties (price and distance).

4.2.2.2 Project(Π)

For certain case, the project(Π) operator allows for selecting a subset of the output variables of a graph pattern as the new output variables. In our case we need to add solutions provided by this operator an additional binding for variable $?Pi$.

Definition 4.7. *Let $\tilde{\Omega}$ be a multiset of possibilistic solution mappings, let **va** be a set of variables. For a given possibilisic solution mapping $\tilde{\mu^p}$, we denote:*
*$Proj^{\pi}(\tilde{\mu^p}, va)$ the restriction of $\tilde{\mu^p}$ to variables in **va**. We define the result of a project possibilistic operator as a multiset of possibilistic solution mapping. A formal definition is given as follows:*

$$Project(\tilde{\Omega}, va) = \left\{(Proj^{\pi}(\tilde{\Omega}[\tilde{\mu^p}], va) \mid \tilde{\mu^p} \in \tilde{\Omega}\right\}$$
$$With\ card_{[Proj^{\pi}(\tilde{\Omega}, va)]}(\tilde{\mu^p}) = card_{[\tilde{\Omega}]}(\tilde{\mu^p})$$

What follows is a list of definitions of some Pi-SPARQL operators, we followed the SPARQL specifications defined in [Eric and Andy, 2008].

4.2.2.3 Filter operator

The Filter operator let for restricting the matches of a CGP over a possibilistic graph database G based on the use of inequalities, or other types of expressions. A formal definition is given as follows:

Definition 4.8. *Let $\tilde{\Omega}$ be a multiset of possibilistic solution mappings and **exp** be an expression. We define the Filter operator as follows:*

$$\textbf{\textit{Filter}}\ (expr, \tilde{\Omega}) = \left\{\tilde{\mu^p} | \tilde{\mu^p} = (\mu, \Pi) \in \tilde{\Omega}\ and\ exp(\mu)\ is\ an\ expression\ that\ has\ an\ effective\ boolean\ value\ of\ true\ \right\}$$
$$With\ card_{[Filter(exp, \tilde{\Omega})]}(\tilde{\mu^p}) = card_{[\tilde{\Omega}]}(\tilde{\mu^p})$$

Let be two graph patterns g_1 and g_2. The difference or Diff of g_1 and g_2 is a CGP whose evaluation is defined as the set of matches in the evaluation of g_1 that do not belong to the evaluation of g_2. We give a formal definition as follows.

Definition 4.9. *Let $\tilde{\Omega}_1$ and $\tilde{\Omega}_2$ be multisets of possibilistic solution mappings, we define the Diff operator as follows:*
$\textbf{\textit{Diff}}(\tilde{\Omega}_1, \tilde{\Omega}_2, exp) = \{ \tilde{\mu_1^p} \mid \tilde{\mu_1^p} \in \tilde{\Omega}_1$
such that: for all $\tilde{\mu_2^p} \in \tilde{\Omega}_2$, either $\tilde{\mu_1^p}$ and $\tilde{\mu_2^p}$ are not compatible or $\tilde{\mu_1^p}$ and $\tilde{\mu_2^p}$ are compatible and $exp(merge(\tilde{\mu_1^p}, \tilde{\mu_2^p}))$ has an effective boolean value equals false $\}$

$$With\ card_{[Diff(\tilde{\Omega_1},\tilde{\Omega_2},exp)]}\ (\tilde{\mu^p}) = card_{[\tilde{\Omega_1}]}\ (\tilde{\mu^p})$$

Diff is used internally for the definition of LeftJoin (see Definition 4.10).

Definition 4.10. *Let $\tilde{\Omega}_1$ and $\tilde{\Omega}_2$ be multisets of possibilistic solution mappings and **exp** be an expression. We define the LeftJoin operator as follows:*

$$\textbf{\textit{LeftJoin}}(\tilde{\Omega}_1, \tilde{\Omega}_2, exp) = Filter(expr, Join(\tilde{\Omega}_1, \tilde{\Omega}_2)) \cup Diff(\tilde{\Omega}_1, \tilde{\Omega}_2, exp)$$

$$With\ card_{[LeftJoin(\tilde{\Omega_1},\tilde{\Omega_2},exp)]}(\tilde{\mu^p}) = card_{[Filter(exp,Join(\tilde{\Omega_1},\tilde{\Omega_2}))]}(\tilde{\mu^p}) + card_{[Diff(\tilde{\Omega_1},\tilde{\Omega_2},exp)]}(\tilde{\mu^p})$$

4.2.3 SPARQL Extension for Possibility distributions Requirements

In this section, we describe formally the different clauses of Pi-SPARQL such: POSSIBLE AS and we explain how we adapt existing clauses to deal with data pervaded with uncertainty.

4.2.3.1 Converting Graph Patterns

Authors in [Eric and Andy, 2008] defined the process of converting graph patterns and solution modifiers in a SPARQL query string into a SPARQL algebra expression. Accordingly, we need to redefine the translation of a graph pattern to algebra expressions to consider the possibility measures in Pi-SPARQL. Based on [Eric and Andy, 2008] the process is defined by specifying a recursive Transform procedure. The input to Transform procedure (see Algorithm 3) is a graph pattern and the result is an algebra expression. To consider the different types of graph patterns, the definition of Transform is subdivided. In this section, we redefine the part which considers the graph patterns of the syntax form

of a GroupGraphPattern. We recall that a SPARQL Abstract Query is a tuple (E, DS, R) where: E is a SPARQL algebra expression, DS is an RDF Dataset and R is a query form. To consider the Pi-SPARQl requirements, we adjusted the Transform procedure as shown in the bold faced parts of Algorithm 3, in lines (3, 7, 8, 18, 19, 20). We used the following symbols Join(Pattern, Pattern), LeftJoin(Pattern, Pattern, expression), Filter(expression, Pattern) and the new algebra symbol: $PA(Pattern, Variable)$ (see Definition 4.11).

Algorithm 3: Adjusted Transform procedure for Pi-SPARQL requirements

```
1  Let FS := ∅;
2  Let G := the empty pattern, /* a basic graph pattern which is the empty set */
3  Let PO := ∅;
4  foreach element E in the GroupGraphPattern do
5      if E is of the form FILTER(expr) then
6          FS := FS ∪ expr;
7      if E is of the form POSSIBLE AS v then
8          PO := PO ∪ v;
9      if E is of the form OPTIONAL {P} then
10         Let A := Transform(P);
11         if A is of the form Filter(F, A2) then
12             G := LeftJoin(G, A2, F);
13         else
14             G := LeftJoin(G, A, true);
15     if E is any other form: then
16         Let A := Transform(E);
17         G := Join(G, A);
18 if PO ≠ 0 ; then
19     foreach variable v in PO do
20         G := PA(G, v);              /*Using the POSSIBLE AS operator*/
21 if FS ≠ ∅: then
22     Let X := Conjunction of expressions in FS
23     G := Filter(X, G)
24 return G.
```

4.2.3.2 Pi-SPARQL Algebra: Project Possibility Operator

The project possibility operator evaluates the POSSIBLE AS clause. For every mapping the operator accesses the possibility measure, creates a new variable binding which maps the specified variable to an RDF literal that represents the possibility measure, and adds

the new binding to the mapping.

Definition 4.11. *Let $\tilde{\Omega}$ be a multiset of possibilistic solution mappings; let $\boldsymbol{v}$ be a query variable which is not bound in any $\tilde{\mu^p} \in \tilde{\Omega}$, let L(t) be a function that returns an RDF literal with the value of π. The result of a project possibility operator is a multiset of possibilistic solution mappings which is defined as:*
$PA(v, \tilde{\Omega}) = \left\{(\mu', \Pi) \mid (\mu, \Pi) \in \tilde{\Omega} \wedge \mu' = \mu \cup \{(v, L(\Pi))\}\right\}$
With $card_{PA(v,\tilde{\Omega})}(\tilde{\mu^p}) = card_{\tilde{\Omega}\hat{\mathrm{A}}}(\tilde{\mu^p})$

Example 35. *We illustrate in Fig. 4.5 the sample solutions for the query pattern of the query Q3. Every solution provided by the project possibility operator contains an additional binding for variable ?Pi that maps ?Pi to a value corresponding to the possibility measure. This possibility measure is associated with the respective solution when the project possibility operator is evaluated (e.g. $\Pi = 1$ for the price property). Note, we used the principle of minimal specificity [Simon et al., 2018] to compute the possibility measure of the resulting solution. The principle of minimal specificity considers the min operator to compute the joint possibility distributions. Thus, the possibility of the resulting solution is 0.5. The*

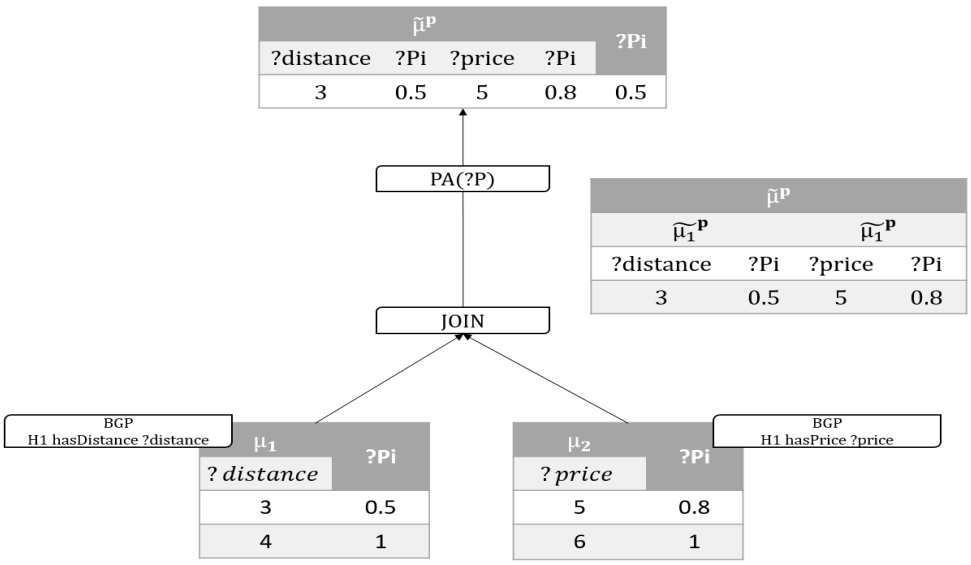


FIGURE 4.5 – Project possibility operator.

POSSIBLE AS clause has been defined for the whole group graph pattern (see example 34). Intuitively, the project possibility operator is applied after joining the solutions.

The extension of SPARQL query language to a possibility-aware query language have been published into the International Conference on Tools with Artificial Intelligence (ICTAI) [Amna et al., 2018a]. In the coming section, we propose to extend skyline queries over possiblisitc RDF data to filter the massive ammount of such data.

4.3 Possibilistic Skyline over RDF data

In this section, we first introduce an extension of the dominance relationship for RDF data modeled by possibility theory. In addition, we define the possibilistic skyline denoted p-Sky$_{\mathcal{H}}$ which retrieves the most interesting subjects over the RDF data set $\mathcal{H}$ according to a subset of properties(predicates).

4.3.1 Comparison of two possibility distributions

The dominance operator is based on comparison between the values of the properties (see Definition 2.7). In the context of certain data, the comparison between two values produces a binary result (0 if false, and 1 if true). However, for the uncertain case, comparison is not binary, it is rather quantified with a degree.

In this subsection, given tow possibilistic values X and Y, we define the degree of possibility that $X < Y$ and $X \leq Y$.

Definition 4.12. ***Comparing possibility distributions***

Let X and Y be two ill-known values represented by two possibility distributions Π_X, Π_Y, the comparison between X and Y is as follows [Dubois and Prade, 1983]:

$$\Pi(X \leq Y) = \begin{cases} \max\Big(\min_{x \leq y}(\pi_X(x), \pi_Y(y))\Big) & if\ \exists\ x \in X\ and \exists\ y \in Y\ \mid\ x \leq y \\ 0 & else \end{cases}$$

$$\Pi(X < Y) = \begin{cases} \max\Big(\min_{x < y}(\pi_X(x), \pi_Y(y))\Big) & if\ \exists\ x \in X\ and \exists\ y \in Y\ \mid\ x < y \\ 0 & else \end{cases}$$

The Definition 4.12 is illustrated as follows. Assume two ill-known values X and Y with two possible values for X; x_1 and x_2 and one value y_1 for variable Y. The possibility that X is less than Y is the possibility that $x_1 < y_1$ or the possibility that $x_2 < y_1$ or both, hence using the *max* function. Now, what does it mean the possibility that $x_1 < y_1$ denoted $\Pi(x_1 < y_1)$? It measures the possibility that a value x_1 is less than y_1. It is zero if $x_1 < y_2$ is false, and $min(\pi(x_1), \pi(y_1))$ if $x_1 < y_1$ is true. We return the minimum because it measures the existence possibility of both x_1 and y_1 (see Subsection 2.1.3.1).

Example 36. *We have the following information about the hotels price h_1 and h_2 defined on the predicate Price*[3]*. We have two possibility distributions $h_1.pr$ and $h_2.pr$ described as follows: $h_1.pr = \{2\backslash 0.2,\ 3\backslash 0.8,\ 7\backslash 1\}$* [4] *and $h_2.pr = \{5\backslash 1, 6\backslash 0.6\}$. In $h_1.pr$, we suppose that the hotel price is 2 with a possibility degree equals to 0.2 and it is 3 with a possibility degree equals to 0.8 and 7 with a possibility degree equal to 1.*
Using the Definition 4.12, let us compute the degrees of possibility that $h_1.pr$ is smaller or equal to $h_2.pr$:
$\Pi(h_1.pr \leq h_2.pr) = \max\Big(\min(0.2,1), \min(0.2,0.6), \min(0.8,1), \min(0.8,0.6), 0, 0\Big) =$
$\max\Big(0.2, 0.2, 0.8, 0.6, 0, 0\Big)$ *=0.8*
*To explain intuitively this result, we detail the operation here. To assess $\Pi(h_1.pr \leq h_2.pr)$, we should compare all combinations of the distributions' elements of $h_1.pr$ and h2.pr. All combinations are mentioned in Table 4.4. In the first line, $h_1.pr$ and $h_2.pr$ are assumed to be equal to 2 and 5, respectively. In this case, it is true that $h_1.pr = 2 \leq h_2.pr = 5$, so $\Pi(h_1.pr \leq h_2.pr)$ is equivalent to $\Pi(h_1.pr = 2 \wedge h_2.pr = 5)$, hence the use of the min operator, inherited from the conjunctive logical operator as noted in line 1, column 3. In the last line (Table 4.4), prices of hotels h_1 and h_2 are assumed to be 7 and 6 respectively. Here it is not possible that $h_1.pr \leq h_2.pr$, so $\Pi(h_1.pr \leq h_2.pr) = 0$. After evaluation of $\Pi(h_1.pr \leq h_2.pr)$ for each combination, we can deduce the overall possibility degree of comparing the two distributions. The couple $(h_1.pr, h_2.pr)$ is either $(2,5)$ or $(2,6)$, or $(3,5)$ etc, so the max operator is used to assess the overall possibility because of the disjunctive nature of the logical operation. Thus, $\Pi(h_1.pr \leq h_2.pr) = max(\Pi(h_1.pr = 2 \leq h_2.pr = 5), \Pi(h_1.pr = 2 \leq h_2.pr = 6), ..., \Pi(h_1.pr = 7 \leq h_2.pr = 6)) = 0.8$

TABLE 4.4 – Comparison of two possibility distributions

$h_1.pr$	$h_2.pr$	$\Pi(h_1.pr \leq h_2.pr)$
2	5	min(0.2,1)=0.2
2	6	min(0.2,0.6)=0.2
3	5	min(0.8,1)=0.8
3	6	min(0.8,0.6)=0.6
7	5	0
7	6	0

3. For short, we use *pr* to denote the price of a given hotel, it is so for the rest of the chapter.
4. $\pi(2) = 0.2$, $\pi(3) = 0.8$, $\pi(7) = 1$

4.3.2 Possibilistic dominance on RDF data

We need now to define the dominance between two possibilistic RDF triples. To consider the uncertain nature of the data, we adapt the Pareto dominance shown in Definition 2.7. The logical connectors $\bigwedge$ and $\bigvee$, that represents the conjunction and disjunction of two binary comparisons, are changed to the minimum and maximum functions, respectively. Given a set of RDF data modeled by possibility theory $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ defined on a set of predicates $\mathcal{A} = \{a_1, a_2, \ldots, a_d\}$, with $h_i.a_k$ denoting the possibilistic value of object h_i w.r.t. predicate a_k.

Let us now extend the dominance relationship to uncertain RDF data modeled in the possibilistic theory framework.

Definition 4.13. *Given two subjects h_i and h_j in $\mathcal{H} : h_i \neq h_j$, the degree of possibility that h_i dominates h_j is given by:*

$$\Pi(h_i \succ h_j) = \min\Big(min_{a_k \in \mathcal{A}}(\Pi(h_i.a_k \leq h_j.a_k)), max_{a_k \in \mathcal{A}}(\Pi(h_i.a_k < h_j.a_k))\Big), \forall a_k \in \mathcal{A} \tag{4.1}$$

where $h_i.a_k$ denotes the possibilistic value of object h_i defined on the predicate a_k.

Table 4.5 – RDF data modeled by possibility theory.

Hotels	Price	Distance
h_1	$\{5\backslash 0.8, 6\backslash 1\}$	$\{3\backslash 0.5, 4\backslash 1\}$
h_2	$\{3\backslash 0.6, 5\backslash 1\}$	$\{4\backslash 0.8, 5\backslash 1\}$
h_3	$\{2\backslash 0.2, 5\backslash 1\}$	$\{1\backslash 0.2, 3\backslash 1\}$
h_4	$\{1\backslash 0.3, 4\backslash 1\}$	$\{1\backslash 0.4, 6\backslash 1\}$
h_5	$\{10\backslash 1, 9\backslash 0.2\}$	$\{5\backslash 1, 8\backslash 0.3\}$

Example 37. *Consider the Table 4.5 containing a set of hotels, defined over two predicates: price and distance. The reason behind using the same properties is that by extending skyline queries over possibilistic RDF data, we need to compute the dominance degree between two RDF resources that should have necessarily some common properties. Each hotel may have one or more values provided with possibility degrees w.r.t. each predicate. For example, the price of hotel h_1 comprises the value 5 with a possibility degree equals to 0.8 and the value 6 with a possibility degree equals to 1. Let us compute the degree of possibility that the object h_1 dominates the object h_2. We have:*

$\Pi(h_1.pr \leq h_2.pr) = \max\Big(\min(0.8, 1), 0\Big) = 0.8$
$\Pi(h_1.d \leq h_2.d) = \max\Big(\min(0.5, 0.8), \min(0.5, 1), \min(1, 0.8), \min(1, 1)\Big) = 1$
$\Pi(h_1.pr < h_2.pr) = 0$
$\Pi(h_1.d < h_2.d) = \max\Big(\min(0.5, 0.8), \min(0.5, 1), \min(1, 1)\Big) = 1$

As a result, according to the Equation 4.1,
$\Pi(h_1 \succ h_2) = min\Big(min(0.8, 1), max(0, 1)\Big) = 0.8$

Definition 4.14. ***The p-dominance*** *Given two subjects $h_i, h_j \in \mathcal{H}$ and a possible threshold p, h_i possibly dominates (p-dominates) h_j, denoted by $h_i \succ_p h_j$ if and only if $\Pi(o_i \succ o_j) \geq p$.*

Subject	Predicate	Object	Possibility degree(π)
h_l	HasPrice	3	0.8
h_l	HasPrice	5	1
h_l	HasDistance	13	0.6
h_l	HasDistance	15	1
h_m	HasPrice	1	1
h_m	HasPrice	4	0.8
h_m	HasDistance	21	1
h_m	HasDistance	44	0.8
h_n	HasPrice	2	1
h_n	HasPrice	2.5	0.4
h_n	HasDistance	27	1
h_n	HasDistance	25	0.3

TABLE 4.6 – RDF Data: hotel properties

In order to define the possibilistic skyline over the RDF data, it is essential to investigate some key properties of the possibilistic dominance (p-dominance). Some of these properties are used further to optimize skyline computation.
Note, the physical data model used in our work is an extension of the 3Store model presented in [Harris and Gibbins, 2003b] and [Sakr and Al-Naymat, 2010]. It represents possibility degree in addition to the subject, predicate and object information. More details about the different kinds of RDF storage techniques are available in the excellent surveys [Faye et al., 2012] and [Özsu, 2016].

Property 1. *The p-dominance relationship does not satisfy the property of transitivity.*

Proof 5. *Consider the subjects depicted in Table 4.6, using RDF quadruple store representation [Harris and Gibbins, 2003b] [Sakr and Al-Naymat, 2010], related to the subjects*

defined on the predicates Price and Distance, respectively. Each quadruple $< S, P, O, \Pi >$ *is stored directly in a four-column table. We have* $\Pi(h_l \succ h_m) = 0.8$, $\Pi(h_m \succ h_n) = 1$, *and* $\Pi(h_l \succ h_n) = 0$. *Observe that,* h_l *0.8-dominates* h_m *and* h_m *1-dominates* h_n, *but* h_l *does not 0.8-dominate* h_n. *Thus, the p-dominance relationship is not transitive.*

Given a subject h_i, we denote by $h_i.a_k^-$ and by $h_i.a_k^+$ respectively the minimum value and the maximum value of $h_i.a_k$. For example, in Table 4.6, the possibilistic price value is $h_l.price = \{3\backslash 0.8, 5\backslash 1\}$. One can observe that $h_l.price^- = 3$ and $h_l.price^+ = 5$.

Property 2. *if* $h_i.a_k^+ < h_j.a_k^-$ *then* $\Pi(h_i.a_k \leq h_j.a_k) = 1$.

Proof 6. *if* h_j *does not dominate* h_i, *i.e.,* $h_j \not\succ h_i$ *then* $\Pi(h_j.a_k \leq h_i.a_k) = 0$. *Given a normalized possibility distribution, an ill-known variable* X, $\exists x \in X, \Pi_X(x) = 1$, *as we use* $\max(\min())$ *functions to compare the possibility distributions, the value* 1 *will appear in the maximum value of our possibility comparison.*

Example 38. *Let* $h_i.a_k$ *and* $h_j.a_k$ *be two possibility distributions defined on subjects* h_i *and* h_j, *respectively, and defined on the predicate* a_k *such that* $h_i.a_k = \{1\backslash 0.8, 2\backslash 1\}$ *and* $h_j.a_k = \{3\backslash 1, 4\backslash 0.2\}$.
We have $\Pi(h_i.a_k \leq h_j.a_k) = \max(\min(0.8, 1), \min(0.8, 0.2), \min(1, 1), \min(1, 0.2)) = 1$ *since* $h_i.a_k^+ = 2 < h_j.a_k^- = 3$.

Property 3. *if* $\exists a_k \in \mathcal{A}$ *where* $h_i.a_k^- > h_j.a_k^+$ *then* $\Pi(h_i.a_k \leq h_j.a_k) = 0$

Property 4. *if* $\exists a_k \in \mathcal{A}$ *where* $\Pi(h_i.a_k \leq h_j.a_k) = 0$ *then* $\Pi(h_i \succ h_j) = 0$

Property 5. *Given a possibility threshold* $p \in]0, 1]$, *if* $\exists a_k \in \mathcal{A}$ *where* $\Pi(h_i.a_k \leq h_j.a_k) < p$, *then* $\Pi((h_i \succ h_j) < p$.

Property 6. *Let* X *and* Y *be two ill-known variables defined on the subjects* h_X *and* h_Y, *respectively. Let* p *be the possibility threshold. Let* $\Pi(x)^+$ *be the maximum possibility degree of the possibility distribution* X *where* $\Pi(x)^+ < 1$. *Let* y^+ *be the greatest value (proposition) of variable* Y.
if $x > y^+$ *and* $\Pi(x)^+ < p$ *then* $\Pi(X \leq Y) < p$. *Thus,* $\Pi(h_X \succ h_Y) < p$

Example 39. *Let* $p = 0.4$. *Consider the following possibility distributions;* $X = \{5\backslash 0.3, 7\backslash 1\}$ *and* $Y = \{3\backslash 0.2, 4\backslash 1\}$.
We have $x = 5$ *and* $y^+ = 4$. *One can observe that* $5 > 4$ *and* $\Pi(5) = 0.3 < 0.4$, *then,* $\Pi(X \leq Y) = 0$. *Thus,* $\Pi(h_X \succ h_Y) < p$.

4.3.3 Possibilistic skyline on RDF data

Intuitively, a subject is in the possible skyline if it is not possibly dominated by another subject. Based on the p-dominance relationship, the notion of the p-skyline is defined as follows.

Definition 4.15. *(The possibilistic Skyline (p-Skyline)) The possibilistic Skyline of a data set $\mathcal{H}$, denoted by p-Sky$_\mathcal{H}$, comprises those subjects in $\mathcal{H}$ that are not p-dominated by any other object, i.e.,*
p-Sky$_\mathcal{H}$ $= \{h_i \in \mathcal{H} \mid \nexists\ h_j \in \mathcal{H}, h_j \succ_p h_i\}$.

Example 40. *Let the possibility threshold p where $p = 0.5$. In Table 4.5, we have $\Pi(h_1 \succ h_2) = 0.8$, thus, h_2 can not be in the 0.5-Skyline. In addition, $\Pi(h_2 \succ h_1) = 0.8$ and $\Pi(h_2 \succ h_5) = 1$. As a result, h_1, h_2 and h_5 can not be the 0.5-Skyline since they are 0.5-dominated. However, $\Pi(h_4 \succ h_3) = 0.4$ and $\Pi(h_3 \succ h_4) = 0.2$. Thus, $\{h_3, h_4\}$ form the 0.5-Skyline.*

Property 7. *Given two possibility thresholds p and p', if $p < p'$ then the p'-skyline is a subset of the p-skyline, i.e., $p < p' \Rightarrow p'$-skyline $\subseteq$ p-skyline.*

Proof 7. *Let be a subject h_i such that $h_i \in$ p-skyline and $h_i \notin p'$-skyline. $\nexists h_j, \Pi(h_j \succ h_i) \geq p$. If $p < p' \Rightarrow \nexists h_j, \Pi(h_j \succ h_i) \geq p'$since $h_i \notin p' - skyline$*

Property 7 indicates that the size of the p'-Skyline is smaller than the p-Skyline if $p < p'$. Roughly speaking, from Property 7, we can see that users have the flexibility to control the size of the retrieved possibilistic skyline by varying the possibility threshold p.

4.4 Possibilistic skyline computation

In this section, we present the skyline algorithms that answer the skyline query over uncertain RDF data modeled by possibility theory. Then, we introduce efficient methods in order to reduce the complexity of the possibilistic dominance computation.
A straightforward algorithm to compute the Skyline over Possibilistic RDF data (denoted by BSPR) is to compare each subject h_i against the other subjects. If h_i is not p-dominated, then it belongs to the possibilistic skyline (p-skyline). However, this approach results in a high computational cost (see Section 4.4.1) as it needs to compare each subject with every other subjects in $\mathcal{H}$, it's complexity is $O(n^2)$.
Also, while the p-dominance relationship is not transitive (see Property 1), a subject cannot be eliminated from the comparison even if it is p-dominated since it will be useful

for eliminating other subjects. For this reason, we propose an algorithm (see Algorithm 4) that follows the principle of the two scan algorithm [Chee Yong et al., 2006].
Our proposed algorithm, named *SPR*, computes the possibilistic skyline through two phases. In the first phase (lines 2–12), a set of candidate subjects p-skyline is selected by comparing each subject h_i in $\mathcal{H}$ with those selected in p-skyline. If an object h_j in p-skyline is p-dominated by h_j, then h_j is removed from the set of candidate subjects since it is not part of the possibilistic skyline. At the end of the comparison of h_i with subjects of p-skyline, if h_i is not p-dominated by any subject then, it is added to p-skyline as a candidate subject. After the first phase, the p-skyline comprises a set of subjects that may be part of the p-skyline. The complexity of this method is $O(n)$.
To avoid the situation illustrated by the example in Table 4.6, a second phase is needed (lines 13–16). To determine if a subject h_i in the set p-skyline is indeed a really skyline subject, it is sufficient to compare h_i with those in $\mathcal{H} \setminus \{p\text{-skyline} \cup undom(h_i) \cup \{h_i\}\}$ that occurs earlier than h_i since the other ones have been already compared against h_i, where $undom(h_i)$ is the set of subjects that occurs before h_i and that do not p-dominate h_j. This set is computed in the first phase in order to reduce the dominance checks in the second phase.

Even if *SPR* minimizes the number of dominance checks, it also may result in a high computational cost, in particular when the average number of propositions x_i in a possibility distribution X, and the number of predicates per subject are large. Thus, it is crucial to optimize the method $\succ_p$ (called in lines 6 and 10 in Algorithm 4) in order to reduce the dominance checks and improve the performance of the *SPR* algorithm. We devise an efficient method that overcomes this problem using the minimum and the maximum values of each possibility distribution w.r.t. each predicate, according to Property 1 and Property 2 .
The method p-dominates(h_i, h_j, p) denoted by $\succ_p$ in the Algorithm 4 (line 6) is detailed in Function p-dominates. The details of the p-dominates(h_i, h_j, p) function are as follows. To reduce the complexity of the dominance computation, we essentially rely on the Properties 2 and 3. For each predicate $a_k \in \mathcal{A}$, $h_i.a_k^-$ is compared against $h_j.a_k^+$. If there is any attribute a_k for which $h_j.a_k^+ < h_i.a_k^-$ holds then return false (loop in line 1); since h_i cannot p-dominate h_j according to Property 3.

4.4.1 Experimental Evaluation

In this section, we present an extensive experimental evaluation of our approach. More specifically, we focus on two issues: (i) the size of the skyline; and (ii) the scalability of our proposed methods for computing the possibilistic skyline. For comparison purposes,

Algorithm 4: The Skyline over Possibilistic RDF data SPR

Input: Subjects $\mathcal{H}$; Possibility threshold p;
Output: Possibilistic skyline p-skyline;

```
1  begin
2      foreach h_i in H do
3          isSkyline ← true
4          foreach h_j in p-skyline do
5              if isSkyline then
6                  if h_j ≻_p h_i then
7                      isSkyline ← false
8                  else
9                      undom(h_i) ← undom(h_i) ∪ {h_j}
10             if h_i ≻_p h_j then
11                 remove h_j from p-skyline
12         if isSkyline then
13             insert h_i in p-skyline
14     foreach h_i in p-skyline do
15         foreach h_j in H \ (p-skyline∪undom(h_i) ∪ {h_i}), pos(h_j) < pos(h_i) do
16             if h_j ≻_p h_i then
17                 remove h_i from p-skyline
18     return p-skyline
```

Function p-dominates(h_i, h_j, p)

Input: Objects h_i,h_j; Possibility threshold p;
Output: boolean;

```
1  foreach a_k in A do
2      if h_i.a_k^- > h_j.a_k^+ then
3          return false
4  Dom ← 0
5  foreach a_k in A do
6      pi[a_k] ← Π(h_i.a_k ≤ h_j.a_k)
7      piStrict[a_k] ← Π(h_i.a_k < h_j.a_k)
8  Dom ← min(min(pi[]), max(piStrict[]))
9  return Dom < p
```

we also implemented a baseline algorithm referred to as *BSPR* (baseline algorithm for the Skyline over Possibilistic RDF data).

4.4.2 Experimental Setup

The generation of synthetic sets of RDF data modelled by possibility theory is controlled by the parameters in Table 4.7, which lists the parameters under investigation, their examined and default values. In each experimental setup, we investigate the effect of one parameter, while we set the remaining ones to their default values. The data generator and the algorithms, i.e., *SPR* and *BSPR* were implemented in Java, and all experiments were conducted on a 2.3 GHz Intel Core i7 processor, with 6GB of RAM.

TABLE 4.7 – Parameters and Examined Values.

Parameter	Symbol	Values	Default
Number of subjects	n	1K, 2K, 5K, 8K, 10K, 50K, 100K, 500K	10K
Number of predicates	d	4, 5, 7, 10, 15, 20	4
Max Nbr of propositions/pred	s	2, 3, 5, 8, 10	3
Possibility threshold	p	0.1, 0.3, 0.5, 0.7, 0.9	0.5
Cardinality/pred	c	20, 50, 70, 100	20

4.4.3 Size of the Skyline on RDF Data

Figure 4.6 shows the size (i.e., the number of subjects returned) of the possibilistic skyline w.r.t. n, d, s, p, and c. Recall that when we varry a parameter, the other parameters take the default values.

In Figure 4.6a, the general trend of the curve is decreasing. It is logic because when we compare a subject against 100 subjects, it has more chance to be not dominated than when we compare it against 1000 subjects. That is why the skyline size decreases when the number of subjects increases.

As shown in Figure 4.6b the cardinality of the skyline on RDF data increases significantly with the increase of d. This is because with the increase of d a subject has better opportunity to be not possibly dominated in all predicates.

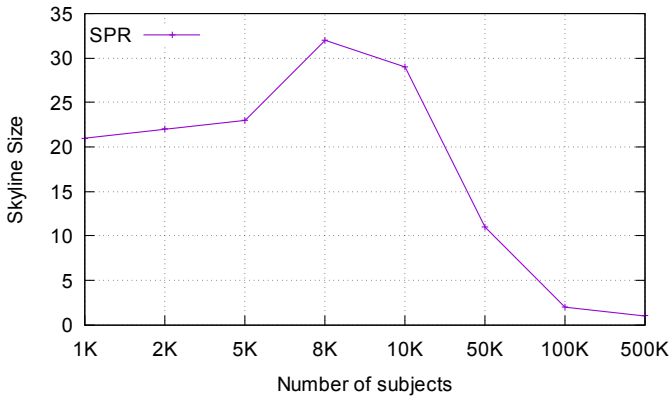
Figure 4.6c shows that the size of the skyline over RDF data increases with the increase of the possibility thresholds p since the possibilistic p-skyline contains the p'-skyline if $p > p'$; see Property 7 – recall that from this property, users have the flexibility to control the size of the returned subjects.

Figure 4.6d shows that the size of the skyline over RDF data decreases with the increase of the number of propositions in a possibility distribution, i.e., the predicate value for each subject. That is because when the comparisons between propositions increase, the

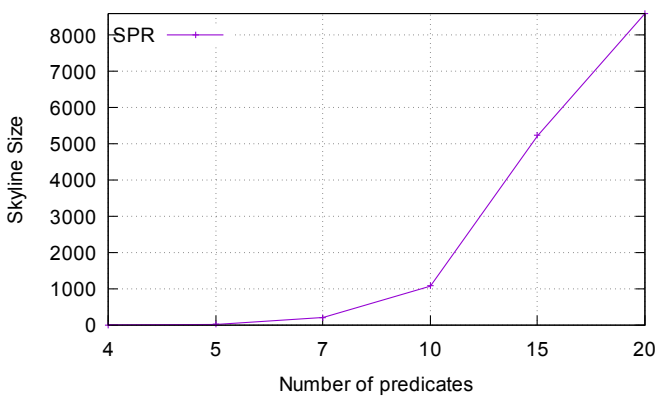
possibility of dominance increases. Thus the skyline size decreases.

4.4.4 Performance and Scalability

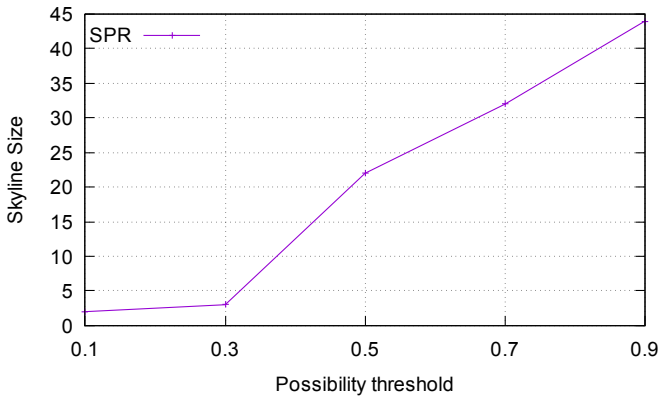
Figure 4.7 depicts the execution time of the implemented algorithms w.r.t. n, d, p and s. Overall, *SPR* outperforms *BSPR*. More specifically, *SPR* is faster than *BSPR*. As expected, Figure 4.7a shows that performance of the algorithms deteriorates with the increase of n. Observe that *SPR* is one order of magnitude faster *BSPR* since it can quickly identifies if a subject is dominated or not. As shown in Figure 4.7b *BSPR* does not scale with d. This is because when d increases the size of the skyline over RDF data becomes larger, thus a large number of subjects will be selected to the second phase. Hence, *BSPR* performs a large number of dominance checks with a basic function. Even if *SPR* performs the same number of dominance checks than *BSPR*, *SPR* is efficient than *BSPR* since it can detect immediately whether a subject dominates or not another. As shown in Figure 4.7c, *SPR* is not affected by p as it computes the possibility dominance between all subjects. However, *BSPR* increases slightly because the size of the skyline increases with the increase of p, thus, less subjects will be eliminated.
Figure 4.7d shows that the execution time of the algorithm slightly decreases with the increase of s. As shown in Figure 4.6d, when the skyline size decreases, the execution time decreases since more the dominance exists, more the search space is early pruned.



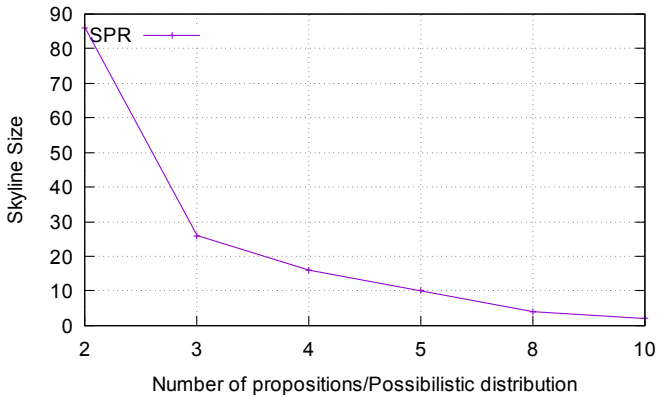
(a) Effect of n



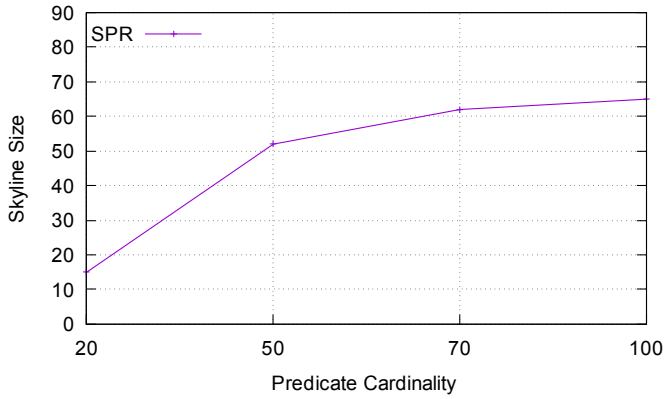
(b) Effect of d



(c) Effect of p

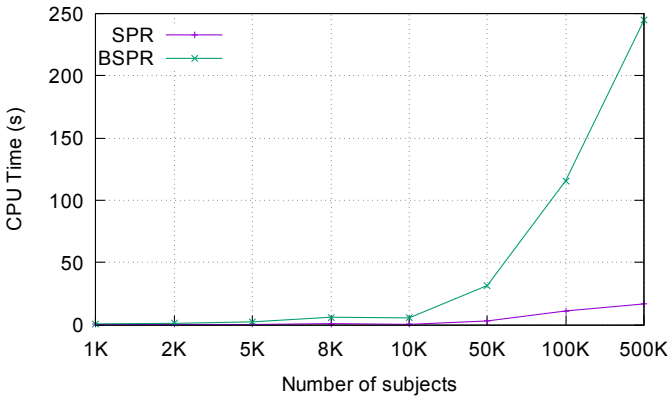


(d) Effect of s

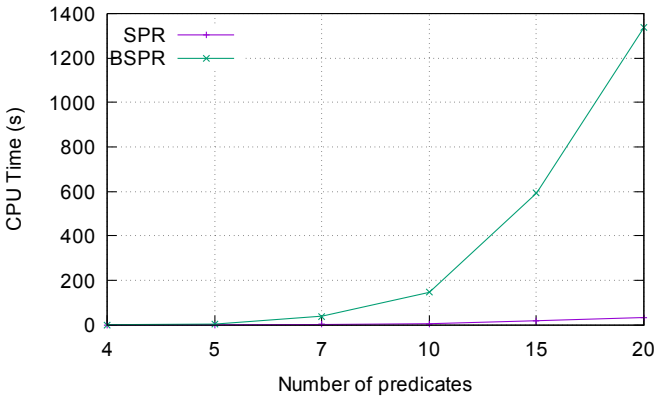


(e) Effect of c

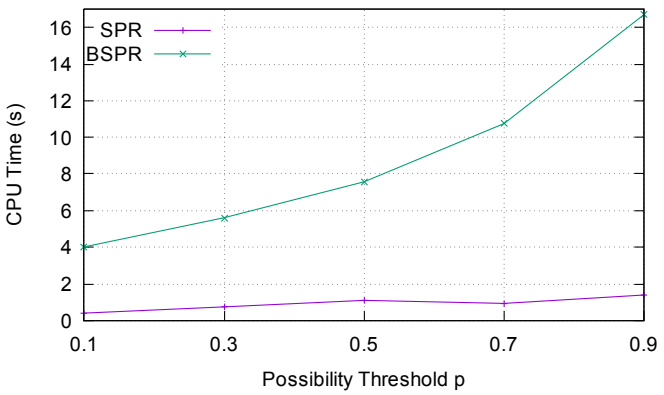
FIGURE 4.6 – Size of the skyline on RDF Data modeled by possibility theory.



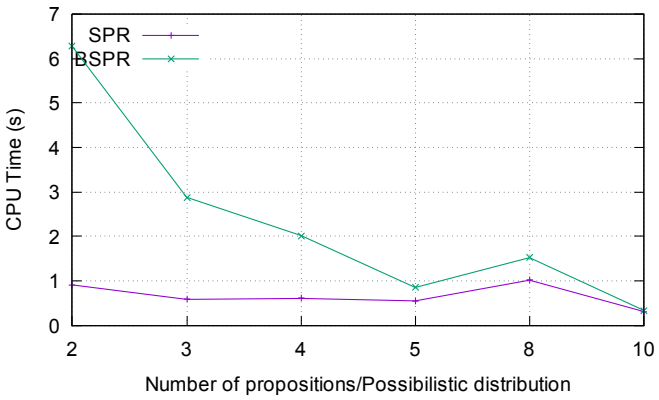
(a) Effect of n



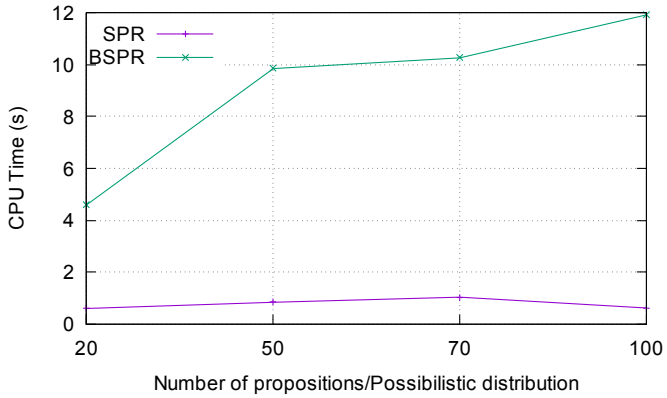
(b) Effect of d



(c) Effect of p



(d) Effect of s



(e) Effect of c

FIGURE 4.7 – Elapsed time to compute the skyline on the RDF data modeled by possibility theory.

Conclusion

In this chapter, we have presented a possibilistic model for uncertain RDF data. In this part, we have proposed a possibilistic model for uncertain RDF data. Indeed, we added a possibility measure to each RDF triple (association between a subject, predicate and object) to model the possibility of such association.
To query possibilistic RDF data, we introduced a general framework denoted Pi-SPARQL. The query evaluation framework is developed based on possibilistic model requirements. Then, appropriate semantics of the solution mappings and evaluation are proposed.
As second part, we extended skyline queries to possibilistic RDF data. The combination aims to filter the massive amount of uncertain resources among the Web according to user preferences. To compute the possibilistic skyline over RDF data, we have implemented two algorithms, the *BSPR*; a naive method to compute skyline queries. Then, we have proposed the *SPR* algorithm to reduce the complexity of *BSPR*. Hence, we have discussed the idea of how to summarize the region of data explored in earlier iterations. Indeed, new candidate skyline points are compared to this summary rather than the rest of the skyline candidates. In addition, as reducing the number of dominance checks is crucial, we proposed to optimize our algorithm by reducing the complexity of the dominance function using the properties that we have established over the possibilistic skyline operator. Experimental evaluations have shown that the *SPR* outperforms the *BSPR* algorithm due to the summarization process of the earlier explored data.

Conclusion

Summary

Semantic Web is a Web of data shared using a set of standards. The large adoption of semantic Web has lead to create huge amount of RDF data. Due to variety of resources and openness of the web, the veracity of collected data is questioned. Uncertainty is an epistemic property that concerns the state of knowledge of an agent about the relationship between the world and a statement about the world. In our case, uncertainty can result from either imprecision/inaccuracy of resources or from inconsistency between them.
In this thesis project, we tackled the issue of uncertain RDF data. We relied on a powerful uncertainty theory, namely possibility theory to extend the RDF data formalism. Hence, we introduced the possibilistic RDF data model. Alongside, to extract data and filter the massive amount of uncertain RDF data, we use the skyline operator to find out a small set of resources that satisfy predefined user preferences.

We summarise below, the main contributions presented in this dissertation:

Notable Findings

— The first part of this work extended the skyline queries over trust-weighted RDF data [Hartig, 2009a]. We introduced a new variant of the skyline, denoted trust-Skyline or $T - Sky^{\alpha}$ (α is a user-defined trust measure). To this end, semantics of Pareto dominance relationship and skyline operator were redefined. Furthermore, new metrics for computing the Trust-skyline were introduced. Indeed, we proposed two algorithms that take into account the trust measures to compute the trust-Skyline set. A naive algotithm and an optimized one based on the transitivity property of the trust dominance operator. We also implemented an SQL query to show how Trust-Skyline can be implemented on a relational database system.

— Second, we introduced a possibilistic model to manage uncertain RDF data. Indeed, we integrated in the structure of RDF data a possibility measure for each subject-property-object triple to reflect the user opinion about the truth of a statement. The possibility measure can be considered as a way to express a source reliability. Furthermore, we described a general framework for supporting SPARQL-like queries on possibilistic RDF data, that we denote Pi-SPARQL. Pi-SPARQL extends SPARQL to allow expressing possibility degrees by associating the solutions for graph patterns with possibility measures. Therefore, Pi-SPARQL proposed appropriate semantics of the solution mappings and evaluation, i.e., it enables users to deal with uncertain RDF data specifications and access the possibility measures associated to the solutions.

— Third, We introduced comparison operators between possibility distributions to allow dominance computation in an RDF data set context. To this aim, we used the skyline operator [Bosc et al., 2011] to extract possibilistic RDF resources that are possibly dominated by no other resources in the sense of Pareto dominance definition. We proposed appropriate semantics of the *possibilistic-skyline* to extract the most interesting resources in a possibilistic RDF database according to user-defined criteria.

— Finally, We provided efficient methods to compute the possibilistic-skyline as well, experiments showed promising results.

Future Work

As for future work, we plan to pursue this study following three lines of research:

— Tackle the issue of the optimization techniques for improving the computation of the possibilistic skyline. Indeed, we plan to extend some existing techniques to summarise the already visited regions, such the Header Point technique presented in [Chen et al., 2011]. It summarizes the region of data explored in earlier join iterations. Meanwhile, we need to extend it to deal with uncertain data.

— Discuss the opportunities to optimize the execution of Pi-SPARQL queries. Indeed, we aim to propose efficient methods for querying Possibilistic RDF data. Such extension opens new challenges linked to filtering query answers, to obtain a set of qualified candidate matches of query Q in a graph G, by adding a possibilistic constraint.

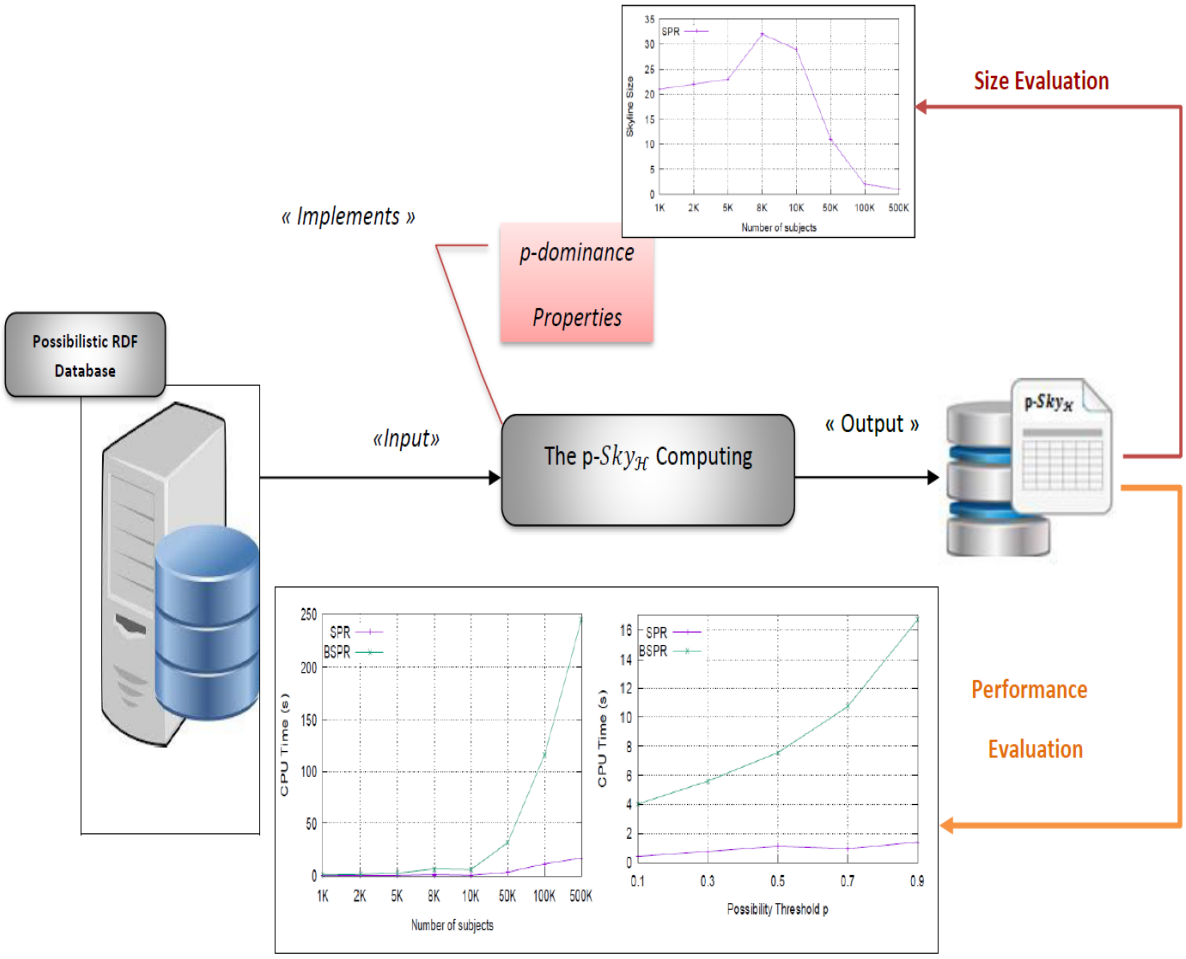
— Another interesting future direction for querying uncertain RDF data is to improve the execution of large SPARQL queries. Indeed, authors in [Huang et al., 2012] distingueshed three kinds of query patterns: the star query patterns (multiple triple

patterns with different properties sharing the same subject), the chain query patterns (sequence of triple patterns where the object of a triple pattern is also the subject of the next triple pattern) and composite query patterns (combination of star and chain patterns).

— Extend possibilistic skyline queries over Graph data. Given that RDF data can be thought in terms of a decentralized directed labeled graph, it is interesting to tackle the issue of uncertain graph data, where uncertainty is modeled thanks to possibility theory.

Appendices

Possibilistic Skyline model



Academic Achievements

Journal paper

[1] **Amna Abidi**, Sayda Elmi, Mohamed Anis Bach Tobji, Allel Hadjali, Boutheina Ben Yaghlane, Skyline queries over possibilistic RDF data, International Journal of Approximate Reasoning (IJAR), VOL.93, page 277-289, 2018.

Book Chapter

[2] **Amna Abidi**, Mohamed Anis Bach Tobji, Allel Hadjali , Boutheina Ben Yaghlane, Statistical Methods for Use in Analysis of Trust-Skyline Sets Revised selected papers), Lecture Notes in Business Information Processing, Enterprise Information Systems, Springer, June, 2018.

International conferences

[3] **Amna Abidi**, Mohamed Anis BACH TOBJI, Allel HADJALI, Boutheina Ben Yaghlane, A General Framework for Querying Possibilistic RDF Data, International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, Volos, Greece, November, 2018, pp. 158-162

[4] **Amna Abidi**, Mohamed Anis Bach Tobji, Allel Hadjali , Boutheina Ben Yaghlane, Skyline Modeling and Computing over Trust RDF Data, Proc. of the 19th International Conference on Enterprise Information Systems (ICEIS'2017), page, 634-643, April, 2017, Porto, Portugal. **Best Paper Award**

Lecture Notes (Springer)

[5] **Amna Abidi**, Nassim Barhri, Mohamed Anis Bach Tobji, Allel Hadjali , Boutheina Ben Yaghlane, First steps towards an electronic meta-journal platform based on crowdsourcing, Proc. of the 2nd International Conference on Digital Economy (ICDEc'2017), Springer-LNBIP, 04-06 May, 2017, Sidi Bou Said, Tunisia.

[6] Rim Jallouli, Mohamed Jallouli, Jihene Rekik, Mohamed Anis Bach Tobji, **Amna Abidi**, Nassim Bahri, The CEVEP medical application for innovative management of Central Venous Port (CVP) Technical issues and research challenges. In proceedings of the International Conference on Digital Economy (ICDEc'2016), Springer-LNBIP, page 68-73, 2016, Carthage, Tunisia.

INSTICC

Institute for Systems and Technologies of Information, Control and Communication

Best Paper Award Certificate

in the Area of
Software Agents and Internet Computing

for the paper entitled:

Skyline Modeling and Computing over Trust RDF Data

authored by:

Amna Abidi, Mohamed Anis Bach Tobji, Allel Hadjali and Boutheina Ben Yaghlane

received at the

19th International Conference on Enterprise Information Systems (ICEIS)

held in Porto - Portugal, April 26 - 29, 2017

On behalf of the Organizing Committee,

Joaquim Filipe
ICEIS Conference Co-chair

e-mail: secretariat@insticc.org

web site: http://www.insticc.org

Bibliography

[Abadi et al., 2007] Abadi, D. J., Marcus, A., Madden, S. R., and Hollenbach, K. (2007). Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 411–422. 4.1.2

[Abadi et al., 2009] Abadi, D. J., Marcus, A., Madden, S. R., and Hollenbach, K. (2009). Sw-store: A vertically partitioned dbms for semantic web data management. *The VLDB Journal*, 18:385–406.

[Agrawal and Wimmers, 2000] Agrawal, R. and Wimmers, E. L. (2000). A framework for expressing and combining preferences. *SIGMOD Rec.*

[Amna et al., 2017a] Amna, A., Mohamed Anis, B. T., Allel, H., and Boutheina, B. Y. (2017a). Skyline modeling and computing over trust RDF data. In *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems, Volume 2*, pages 634–643. 3, 3.3, 3.3.1, 3.4.1, 3.4.2

[Amna et al., 2017b] Amna, A., Mohamed Anis, B. T., Allel, H., and Boutheina, B. Y. (2017b). Statistical methods for use in analysis of trust-skyline sets. In *Enterprise Information Systems*, pages 413–427. Springer. 3

[Amna et al., 2018a] Amna, A., Mohamed Anis, B. T., Allel, H., and Boutheina, B. Y. (2018a). A General Framework for Querying Possibilistic RDF Data. In *International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 158–162. IEEE. 4, 4.2.3.2

[Amna et al., 2018b] Amna, A., Sayda, E., Mohamed Anis, B. T., Allel, H., and Boutheina, B. Y. (2018b). Skyline queries over possibilistic rdf data. *International Journal of Approximate Reasoning*, 93:277 – 289. 4

[Angles et al., 2017] Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J., and Vrgoč, D. (2017). Foundations of modern query languages for graph databases. *ACM Comput. Surv.*, 50:68:1–68:40. 1.3.5, 4.2.1

[Antoniou and vanHarmelen, 2004a] Antoniou, G. and vanHarmelen, F. (2004a). *A Semantic Web Primer*. MIT Press. 1, 1.1, 1.1.1.1, 1.1.2

[Antoniou and vanHarmelen, 2004b] Antoniou, G. and vanHarmelen, F. (2004b). *A Semantic Web Primer*. MIT Press, Cambridge, MA, USA.

[Bailey et al., 2005] Bailey, J., Bry, F., Furche, T., and Schaffert, S. (2005). Web and semantic web query languages: A survey. In *Proceedings of the First International Conference on Reasoning Web*, pages 35–133. Springer-Verlag. 1.2.4.5

[Berners-Lee, 2004] Berners-Lee, T. (2004). Reifying RDF (properly), and N3. https://www.w3.org/DesignIssues/Reify.html. 1.4.3

[Berners-Lee et al., 1998a] Berners-Lee, T., Fielding, R., and Masinter, L. (1998a). Uniform resource identifiers (uri): Generic syntax. 1

[Berners-Lee et al., 1998b] Berners-Lee, T., Fielding, R., and Masinter, L. (1998b). Uniform resource identifiers (uri): Generic syntax.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (May 2001). The semantic web. *Scientific American*. (document)

[Bornea et al., 2013] Bornea, M. A., Dolby, J., Kementsietsidis, A., Srinivas, K., Dantressangle, P., Udrea, O., and Bhattacharjee, B. (2013). Building an efficient rdf store over a relational database. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 121–132. ACM. 1.2.4.5

[Börzsönyi et al., 2001a] Börzsönyi, S., Kossmann, D., and Stocker, K. (2001a). The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering*, pages 421–430. IEEE Computer Society. (document), 1.1, 2, 2.2.1, 2.2.1, 21, 2.2.1, 2.2.2, 2.2.2, 3, 3.1.2, 3.1.3.2, 4

[Börzsönyi et al., 2001b] Börzsönyi, S., Kossmann, D., and Stocker, K. (2001b). The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering*.

[Bosc et al., 2011] Bosc, P., Hadjali, A., and Pivert, O. (2011). On possibilistic skyline queries. In *Proceedings of the 9th International Conference on Flexible Query Answering Systems*, pages 412–423. Springer-Verlag. 2.2.3.2, 2.2.3.2, 24, 3, 3.1, 4.1, 4.4.4

[Buche et al., 2005] Buche, P., Dervin, C., Haemmerle, O., and Thomopoulos, R. (2005). Fuzzy querying of incomplete, imprecise, and heterogeneously structured data in the relational model using ontologies and rules. *IEEE Transactions on Fuzzy Systems*, 13:373–383. 1.4.1

[Ceolin et al., 2014] Ceolin, D., Nottamkandath, A., Fokkink, W., and Maccatrozzo, V. (2014). Towards the definition of an ontology for trust in (web) data. In *Proceedings of the 10th International Conference on Uncertainty Reasoning for the Semantic Web - Volume 1259*, pages 73–78. 1.4.1

[Chee Yong et al., 2006] Chee Yong, C., H. V., J., Kian-Lee, T., Anthony K. H., T., and Zhenjie, Z. (2006). Finding k-dominant skylines in high dimensional space. In *Procee-*

dings of the ACM SIGMOD International Conference on Management of Data, pages 503–514. 4.4

[Chen et al., 2011] Chen, L., Gao, S., and Anyanwu, K. (2011). Efficiently evaluation skyline queries on RDF databases. In *8th Extended Semantic Web Conference, ESWC 2011*, pages 123–138, Heraklion, Crete, Greece. 4, 3, 3.1, 4, 4.4.4

[Chomicki, 2002] Chomicki, J. (2002). Querying with intrinsic preferences. In *Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology*, pages 34–51. Springer-Verlag. (document), 2, 2.2.1, 2.2.1, 3, 4

[Chomicki, 2003] Chomicki, J. (2003). Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28:427–466.

[Chomicki, 2011] Chomicki, J. (2011). Logical foundations of preference queries. *IEEE Data Eng. Bull.*, 34:3–10. (document), 2, 3, 4

[Chomicki et al., 2013] Chomicki, J., Ciaccia, P., and Meneghetti, N. (2013). Skyline queries, front and back. *SIGMOD Rec.*, 42:6–18. (document), 2, 3, 4

[Codd, 1990] Codd, E. F. (1990). *The Relational Model for Database Management: Version 2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[D. Dubois, 1982] D. Dubois, H. P. (1982). On several representations of an uncertain body of evidence. *Fuzzy Information and Decision Processes*. 2.1.4.1

[Deepak et al., 2011] Deepak, S., Deepak, A., Rakesh, Kr., P., and K. K., A. (2011). Article: An efficient approach of block nested loop algorithm based on rate of block transfer. *International Journal of Computer Applications*, 21:24–30. 2.2.2, 3.1.3.2

[Dempster, 1967] Dempster, A. P. (1967). Upper and lower probabilities induced by a multivalued mapping. *The Annals of Mathematical Statistics*, 38:325–339. 2.1.2.2

[Dempster, 1968] Dempster, A. P. (1968). A generalization of bayesian inference (with discussion). *Journal of the Royal Statistical Society, Series B: Methodological*, 30:205–247.

[Dragoni and Tettamanzi, 2007] Dragoni, M. and Tettamanzi, A. G. B. (2007). Evolutionary algorithms for reasoning in fuzzy description logics with fuzzy quantifiers. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 1967–1974. ACM.

[Dubois and Prade, 1983] Dubois, D. and Prade, H. (1983). Ranking fuzzy numbers in the setting of possibility theory. *Inf. Sci.*, 30:183–224. 4.12

[Dubois and Prade, 1988] Dubois, D. and Prade, H. (1988). *Possibility theory*. Plenum Press, New-York. 2.1.1, 2.1.3, 2.1.3.1

[Dubois and Prade, 2001] Dubois, D. and Prade, H. (2001). Possibility theory, probability theory and multiple-valued logics: A clarification. *Annals of Mathematics and Artificial Intelligence*, 32:35–66. 2.1.4

[DuCharme, 2011a] DuCharme, B. (2011a). *Learning SPARQL*. O'Reilly Media, Inc. 1.2.1, 1.3

[DuCharme, 2011b] DuCharme, B. (2011b). *Learning SPARQL*. O'Reilly Media, Inc.

[Eric and Andy, 2008] Eric, P. and Andy, S. (2008). Sparql query language for rdf. https://www.w3.org/TR/rdf-sparql-query/ #sparqlSolutions. 1.3.2, 1.3.4, 1.3.4, 1.3.5, 1.3.5, 7, 8, 10, 4.2, 4.2.1, 4.2.2, 4.2.2.1, 4.2.2.2, 4.2.3.1

[Evren and Bijan, 2007] Evren, S. and Bijan, P. (2007). SPARQL-DL: SPARQL query for OWL-DL. In *Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions, Innsbruck, Austria, June 6-7, 2007*. 1.3.1

[Faye et al., 2012] Faye, D. C., Curé, O., and Blin, G. (2012). A survey of RDF storage approaches. *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées*, 15:11–35. 4.3.2

[Fionda and Greco, 2015] Fionda, V. and Greco, G. (2015). Trust models for RDF data: Semantics and complexity. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 95–101, Austin, Texas, USA. (document), 1, 1.4.1, 3

[Francois and Stijn, 2011] Francois, P. and Stijn, V. (2011). What are real sparql queries like? In *International Workshop on Semantic Web Information Management*. ACM. 1.3.1

[Frank and Eric, 2003] Frank, M. and Eric, M. (2003). Rdf primer.

[Golbeck, 2006] Golbeck, J. (2006). Combining provenance with trust in social networks for semantic web content filtering. In *Proceedings of the 2006 International Conference on Provenance and Annotation of Data*, pages 101–108. Springer-Verlag. 1.4.1

[Golbeck et al., 2003] Golbeck, J., Parsia, B., and Hendler, J. A. (2003). Trust networks on the semantic web. In *Cooperative Information Agents VII, 7th International Workshop, CIA*, pages 238–249. 1.4.1

[Graham et al., 2004] Graham, K., Jeremy, J. C., and Brian, M. (2004). Resource description framework RDF: Concepts and abstract syntaxr. https://www.w3.org/TR/rdf-concepts/#section-Graph-syntax.

[Gravetter and Wallnau, 2000] Gravetter, F. and Wallnau, L. (2000). *Statistics for the behavioral sciences*. 5th ed. Belmont: Wadsworth, Thomson Learning. 3.3.2

[Gruber, 1993] Gruber, T. R. (1993). Toward principles for the design of ontologies used for knowledge sharing. In *In Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers. 1.1.1.1

[Guo et al., 2005] Guo, Y., Pan, Z., and Heflin, J. (2005). LUBM: A benchmark for owl knowledge base systems. *Web Semant.*, 3:158–182. 1.1.1.2

[Gutierrez et al., 2004] Gutierrez, C., Hurtado, C., and Mendelzon, A. O. (2004). Foundations of semantic web databases. In *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 95–106, New York, NY, USA.

[Gutierrez et al., 2011] Gutierrez, C., Hurtado, C. A., Mendelzon, A. O., and Pérez, J. (2011). Foundations of semantic web databases. *J. Comput. Syst. Sci.*, 77.

[Harris and Gibbins, 2003a] Harris, S. and Gibbins, N. (2003a). 3store: Efficient bulk RDF storage. In *PSSS1 - Practical and Scalable Semantic Systems, Proceedings of the First International Workshop on Practical and Scalable Semantic Systems, Sanibel Island, Florida, USA, October 20, 2003*. 1.2.4.2

[Harris and Gibbins, 2003b] Harris, S. and Gibbins, N. (2003b). 3store: Efficient bulk RDF storage. In *Practical and Scalable Semantic Systems, Proceedings of the First International Workshop on Practical and Scalable Semantic Systems*. 3.4.1, 4.3.2, 5

[Hartig, 2009a] Hartig, O. (2009a). Querying trust in rdf data with tsparql. In *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC*, pages 5–20, Heraklion, Crete, Greece. (document), 1, 1, 1.4.1, 1.4.1, 11, 2.2.3.4, 3, 3.1, 4.1.2, 4.4.4

[Hartig, 2009b] Hartig, O. (2009b). Towards a data-centric notion of trust in the semantic web (a position statement). In *The Semantic Web: Research and Applications,the 7th Extended Semantic Web Conference , ESWC 2010, Heraklion, Greece, May 2010*, pages 5–20. 1.4.1

[Hartig and Thompson, 2014] Hartig, O. and Thompson, B. (2014). Foundations of an alternative approach to reification in rdf. *CoRR*, abs/1406.3399. 1.4.3

[Hernández et al., 2015] Hernández, D., Hogan, A., and Krötzsch, M. (2015). Reifying RDF: what works well with wikidata? In *Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems*, volume 1457, pages 32–47. 1.4.3

[Hristidis et al., 2001] Hristidis, V., Koudas, N., and Papakonstantinou, Y. (2001). Prefer: A system for the efficient execution of multi-parametric ranked queries. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*.

[Huang and Liu, 2009] Huang, H. and Liu, C. (2009). Query evaluation on probabilistic rdf databases. In *Proceedings of the 10th International Conference on Web Information Systems Engineering*, WISE '09, pages 307–320. (document), 1, 1.4.2, 1.3, 1.4.2, 4, 4.1.2

[Huang et al., 2012] Huang, H., Liu, C., and Zhou, X. (2012). Approximating query answering on rdf databases. *World Wide Web*, 15:89–114. 4.4.4

[Hyountaek et al., 2014] Hyountaek, Y., Jongwuk, L., Jinha, K., and Seung, W. H. (2014). Skyline ranking for uncertain databases. *Information Sciences*, 273:247–262. 2.2.3.1

[Jens et al., 2015] Jens, L., Robert, I., Max, J., Anja, J., Dimitris, K., Pablo, N. M., Sebastian, H., Mohamed, M., Patrick, v. K., Sören, A., and Christian, B. (2015). Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195. 1.1.1.2

[Jiang et al., 2012] Jiang, B., Pei, J., Lin, X., and Yuan, Y. (2012). Probabilistic skylines on uncertain data: Model and bounding-pruning-refining methods. *J. Intell. Inf. Syst.*, 38:1–39. (document), 2, 2.2.3.1, 3, 3.1, 4

[Kalyvas and Tzouramanis, 2017] Kalyvas, C. and Tzouramanis, T. (2017). A survey of skyline query processing. *CoRR*, abs/1704.01788. 2.2.2

[Karvounarakis et al., 2002] Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., and Scholl, M. (2002). Rql: A declarative query language for rdf. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 592–603. 1.2.4.5

[Kiessling, 2002] Kiessling, W. (2002). Foundations of preferences in database systems. In *Proceedings of the 28th International Conference on Very Large Data Bases*. 3

[Kijima and Ohnishi, 1999] Kijima, M. and Ohnishi, M. (1999). Stochastic orders and their applications in financial optimization. *Mathematical Methods of Operations Research*, 50:351–372. 2.2.3.3

[Kung et al., 1975] Kung, H. T., Luccio, F., and Preparata, F. P. (1975). On finding the maxima of a set of vectors. *J. ACM*. 2.2.2

[Lian and Chen, 2009] Lian, X. and Chen, L. (2009). Probabilistic inverse ranking queries over uncertain data. In *Proceedings of the 14th International Conference on Database Systems for Advanced Applications*, DASFAA '09. Springer-Verlag. 2.2.3.1

[Lian and Chen, 2011] Lian, X. and Chen, L. (2011). Efficient query answering in probabilistic RDF graphs. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pages 157–168. (document), 1, 1.4.2, 12, 4

[Lin et al., 2011] Lin, X., Zhang, Y., Zhang, W., and Cheema, M. A. (2011). Stochastic skyline operator. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, pages 721–732. IEEE Computer Society. 2.2.3.3

[Mark, 2009] Mark, F. (2009). A smarter web. *Technology Review, November 2001.* (document), 1, 1.1

[McBride, 2002] McBride, B. (2002). Jena: A semantic web toolkit. *IEEE Internet Computing*, 6:55–59. 1.2.4.3

[Meiser et al., 2011] Meiser, T., Dylla, M., and Theobald, M. (2011). Interactive reasoning in uncertain rdf knowledge bases. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 2557–2560. ACM. 1, 1.4.2

[Miller et al., 2002] Miller, L., Seaborne, A., and Reggiori, A. (2002). Three implementations of squishql, a simple rdf query language. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, ISWC '02, pages 423–435. Springer-Verlag. 1.2.4.5

[Neumann and Weikum, 2008] Neumann, T. and Weikum, G. (2008). Rdf-3x: A risc-style engine for rdf. *Proc. VLDB Endow.*, 1:647–659. 1.2.4.2, 1.2.4.5

[Nguyen et al., 2014] Nguyen, V., Bodenreider, O., and Sheth, A. P. (2014). Don't like rdf reification?: making statements about statements using singleton property. In *WWW*, pages 759–770. ACM. 1.4.3

[Olaf, 2014] Olaf, H. (2014). Reconciliation of rdf* and property graphs. *CoRR*, abs/1409.3288. 1.4.1

[Özsu, 2016] Özsu, M. T. (2016). A survey of RDF data management systems. *Front. Comput. Sci.*, 10:418–432. 1.2.4.5, 4.3.2

[Pei et al., 2007] Pei, J., Jiang, B., Lin, X., and Yuan, Y. (2007). Probabilistic skylines on uncertain data. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 15–26. 2.2.3.1, 22, 2.2.3.1, 23, 2.2.3.1

[Pérez et al., 2009] Pérez, J., Arenas, M., and Gutierrez, C. (2009). Semantics and complexity of sparql. *ACM Trans. Database Syst.*, 34:16:1–16:45. 1.3.1.2

[Prade, 1984] Prade, H. (1984). Lipski's approach to incomplete information data bases restated and generalized in the setting of Zadeh's possibility theory. *Inf. Syst.*, 9. 2.1.5, 4.1, 4.1.1

[Preparata and Shamos, 1985] Preparata, F. P. and Shamos, M. (1985). *Computational Geometry: An Introduction*. Springer-Verlag New York. 2.2.2

[Richardson et al., 2003] Richardson, M., Agrawal, R., and Domingos, P. (2003). Trust management for the semantic web. In *Proceedings of the Second International Conference on Semantic Web Conference*, pages 351–368. Springer-Verlag. 1.4.1

[Robinson et al., 2013a] Robinson, I., Webber, J., and Eifrem, E. (2013a). *Graph Databases*. O'Reilly Media, Inc.

[Robinson et al., 2013b] Robinson, I., Webber, J., and Eifrem, E. (2013b). *Graph Databases*. O'Reilly Media, Inc.

[Sakr and Al-Naymat, 2009] Sakr, S. and Al-Naymat, G. (2009). Relational processing of rdf queries: a survey. *SIGMOD Rec.*, 38:23–28. 1.2.4.1, 1.2.4.2, 1.2.4.3, 1.2.4.4, 3.2.1, 3.4.1

[Sakr and Al-Naymat, 2010] Sakr, S. and Al-Naymat, G. (2010). Relational processing of RDF queries: A survey. *SIGMOD Rec.*, 38(4):23–28. 3.4.1, 4.3.2, 5

[Saleem et al., 2015] Saleem, M., Ali, M. I., Hogan, A., Mehmood, Q., and Ngomo, A.-C. N. (2015). Lsq: The linked sparql queries dataset. In *The Semantic Web - ISWC 2015*, pages 261–269, Cham. Springer International Publishing. 1.3.1

[Sayda et al., 2014] Sayda, E., Karim, B., Allel, H., Mohamed Anis, B. T., and Boutheina, B. Y. (2014). Computing skyline from evidential data. In *Scalable Uncertainty Management - 8th International Conference, SUM*, pages 148–161. 2.2.3.4, 25, 2.2.3.4, 2.2.3.4

[Sayda et al., 2016a] Sayda, E., Mohamed Anis, B. T., Allel, H., and Boutheina, B. Y. (2016a). Efficient distributed skyline over imperfect data modeled by the evidence theory. In *Proceeding of the 28th IEEE International Conference on Tools with Artificial Intelligence, ICTAI*, pages 335–342. 2.2.3.4

[Sayda et al., 2016b] Sayda, E., Mohamed Anis, B. T., Allel, H., and Boutheina, B. Y. (2016b). Efficient skyline maintenance over frequently updated evidential databases. In *Proceeding of the16th International Conference, IPMU*, pages 199–210. 2.2.3.4

[Seaborne, 2004] Seaborne, A. (January 2004). Rdql - a query language for rdf. hhttp://www.w3.org/Submission/2004/SUBM-RDQL-20040109/. 1.2.4.5

[Shafer, 1976] Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, NJ. 2.1.2.2

[Sidirourgos et al., 2008] Sidirourgos, L., Goncalves, R., Kersten, M., Nes, N., and Manegold, S. (2008). Column-store support for rdf data management: Not all swans are white. *Proc. VLDB Endow.*, 1:1553–1563. 4.1.2

[Simon et al., 2018] Simon, C., Weber, P., and Sallak, M. (2018). *Data Uncertainty and Important Measures*. ISTE Ltd and John Wiley & Sons. 33, 35

[Smets, 1996] Smets, P. (1996). Imperfect information: Imprecision and uncertainty. In *Uncertainty Management in Information Systems*, pages 225–254. Springer-US. 2.1.1, 2.1.1.2, 2.1.1.3, 2.1.2.1, 2.1.4

[Smets, 1997] Smets, P. (1997). *Imperfect Information: Imprecision and*. Springer, Boston, MA.

[Steve and Andy, 2013] Steve, H. and Andy, S. (2013). Sparql query language. https://www.w3.org/TR/sparql11-query/#sparqlAlgebra.

[Straccia and Mucci, 2015] Straccia, U. and Mucci, M. (2015). pFOIL-DL: Learning (fuzzy) el concept descriptions from crisp owl data using a probabilistic ensemble estimation. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 345–352. ACM. 1.4.3

[Suchanek et al., 2008] Suchanek, F. M., Kasneci, G., and Weikum, G. (2008). Yago: A large ontology from wikipedia and wordnet. *Web Semant.*, 6:203–217. 1.1.1.2

[Tan et al., 2001] Tan, K.-L., Eng, P.-K., and Ooi, B. C. (2001). Efficient progressive skyline computation. In *Proceedings of the 27th International Conference on Very Large Data Bases*. 2.2.1

[Tomaszuk et al., 2012] Tomaszuk, D., Pak, K., and Rybinski, H. (2012). Trust in RDF graphs. In *Advances in Databases and Information Systems - 16th East European Conference, ADBIS*, pages 273–283, Poznań, Poland. (document), 1, 1.4.1, 3

[van Harmelen, 2004] van Harmelen, F. (2004). The semantic web: what, why, how, and when. *IEEE Distributed Systems Online*, 5.

[W3C, 2004] W3C (2004). RDF semantics. https://www.w3.org/. 1.2

[Weiss et al., 2008] Weiss, C., Karras, P., and Bernstein, A. (2008). Hexastore: Sextuple indexing for semantic web data management. *Proc. VLDB Endow.*, 1:1008–1019. 1.2.4.5

[Wu et al., 2006] Wu, P., Zhang, C., Feng, Y., Zhao, B. Y., Agrawal, D., and El Abbadi, A. (2006). Parallelizing skyline queries for scalable distribution. In *Proceedings of the 10th International Conference on Advances in Database Technology*.

[Zadeh, 1978] Zadeh, L. A. (1978). Fuzzy sets as a basis for theory of possibility. *Fuzzy Sets and Systems*, 1:3–28. (document), 2, 2.1.3, 2.1.3.1, 19, 4, 4.1.1

[Zhang et al., 2013] Zhang, Q., Ye, P., Lin, X., and Zhang, Y. (2013). Skyline probability over uncertain preferences. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 395–405. ACM. (document), 2, 3, 3.1, 4

[Zheng et al., 2014] Zheng, W., Zou, L., Lian, X., Hong, L., and Zhao, D. (2014). Efficient subgraph skyline search over large graphs. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, pages 1529–1538. (document), 2, 4

[Zhu et al., 2013] Zhu, J., Qi, G., and Suntisrivaraporn, B. (2013). Tableaux algorithms for expressive possibilistic description logics. In *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 01*, pages 227–232, Washington, DC, USA. IEEE Computer Society. 1.4.3

[Zimmermann, 1985] Zimmermann, H.-J. (1985). *Possibility Theory vs. Probability Theory*, pages 103–118. Springer Netherlands, Dordrecht. (document), 4

[Zou et al., 2010] Zou, L., Chen, L., Özsu, M. T., and Zhao, D. (2010). Dynamic skyline queries in large graphs. In *Proceedings of the 15th International Conference on Database Systems for Advanced Applications - Volume Part II*, DASFAA'10, pages 62–78. (document), 2, 4

Don't ever lose the dreamer inside, no matter what you witness in life.
Keep that part safe, warm and guarded with all your heart and soul

Jeff Emmerson

Base de Données RDF Imparfaites: Du Modélisation à l'interrogation

Résumé: L'intérêt sans cesse croissant des données RDF disponibles sur le Web a conduit à l'émergence de multiple et importants efforts de recherche pour enrichir le formalisme traditionnel des données RDF à des fins d'exploitation et d'analyse. Le travail de cette thèse s'inscrit dans la continuation de ces efforts en abordant la problématique de la gestion des données RDF en présence d'imperfections (manque de confiance/validité, incertitude, etc.). Les contributions de la thèse sont comme suit: (1) Nous avons proposé d'appliquer l'opérateur skyline sur les données RDF pondérées par des mesures de confiance (Trust-RDF) dans le but d'extraire les ressources les plus confiantes selon des critères définis par l'utilisateur. (2) Nous avons discuté via des méthodes statistiques l'impact des mesures de confiance sur le Trust-skyline.
(3) Nous avons intégré à la structure des données RDF un quatrième élément, exprimant une mesure de possibilité. Pour gérer cette mesure de possibilité, un cadre langagier appropriée est étudié, à savoir Pi-SPARQL, qui étend le langage SPARQL aux requêtes permettant de traiter des distributions de possibilités. (4) Nous avons étudié une variante d'opérateur skyline pour extraire les ressources RDF possibilistes qui ne sont éventuellement dominées par aucune autre ressource dans le sens de l'optimalité de Pareto.

Mots clés:Web sémantique, Ressource Description Framework, SPARQL, Bases de Degré de confiance, Incertitude, Théorie des possibilités, Requêtes à préférences, Opérateur Skyline, données–Interrogation, Conception centrée sur l'utilisateur

Imperfect RDF Databases: From Modelling to Querying

Abstract: The ever-increasing interest of RDF data on the Web has led to several and important research efforts to enrich traditional RDF data formalism for the exploitation and analysis purpose. The work of this thesis is a part of the continuation of those efforts by addressing the issue of RDF data management in presence of imperfection (untruthfulness, uncertainty, etc.). The main contributions of this dissertation are as follows. (1) We tackled the trusted RDF data model. Hence, we proposed to extend the skyline queries over trust RDF data, which consists in extracting the most interesting trusted resources according to user-defined criteria. (2) We studied via statistical methods the impact of the trust measure on the Trust-skyline set.
(3) We integrated in the structure of RDF data (i.e., subject-property-object triple) a fourth element expressing a possibility measure to reflect the user opinion about the truth of a statement. To deal with possibility requirements, appropriate framework related to language is introduced, namely Pi-SPARQL, that extends SPARQL to be possibility-aware query language.
Finally, we studied a new skyline operator variant to extract possibilistic RDF resources that are possibly dominated by no other resources in the sense of Pareto optimality.

KeyWords: Semantic web, Ressource Description Framework, SPARQL, Trust, Uncertainty, Possibility theory, Preference queries, Skyline queries, Querying, User-centered system design