Université de Paris

Ecole doctorale Pierre Louis de santé publique

Epidémiologie et sciences de l'information biomédicale

INSERM 1138 EQ22 Sciences de l'information au service de la médecine personnalisée

en cotutelle avec

Université de Sherbrooke

Faculté des sciences, Département d'informatique

GRIIS — Groupe de recherche interdisciplinaire en informatique de la santé

Construction de modèles de données relationnels temporalisés guidée par les ontologies

Christina Khnaisser

Thèse de doctorat en informatique biomédicale

Dirigée par Prof. Anita Burgun, Prof. Jean-François Ethier et Prof. Luc Lavoie

Présentée et soutenue publiquement le 4 novembre 2019

Devant un jury composé de :

Mme Anne BERGERON	Professeure	Université du Québec à Montréal	Rapporteur
M. Rémi BASTIDE	Professeur	Université Champollion	Rapporteur
Mme Angela BONIFATI	Professeure	Université de Lyon 1	Examinateur
M. Marc CUGGIA	Professeur	Université de Rennes 1	Examinateur
Mme Anita BURGUN	Professeure	Université Paris Descartes	Directrice
M. Jean-François ETHIER	Professeur	Université de Sherbrooke	Directeur
M. Luc LAVOIE	Professeur	Université de Sherbrooke	Directeur

J'ai commencé mes études supérieures en 2014 avec ma maitrise grâce à deux professeurs, Luc Lavoie et Jean-François Ethier.

Je les remercie énormément de m'avoir passionnément soutenue dans mes projets de recherche et d'enseignement. Merci d'avoir travaillé très dur pour créer le GRIIS, ma deuxième maison.

C'est très émouvant pour moi d'être la première doctorante de ce groupe de recherche.

Je dédie cette thèse au GRIIS.

Remerciements

Je remercie chaleureusement et j'exprime ma gratitude à toutes les personnes qui ont contribué de près ou de loin à mes travaux de recherches, en particulier :

Ma directrice Anita Burgun pour m'avoir donné l'opportunité d'effectuer une partie de ma thèse dans son laboratoire de recherche et de m'avoir incitée à publier. Merci pour ta confiance, ton temps et tes précieux conseils. J'espère pouvoir continuer notre collaboration en recherche.

Mon directeur Luc Lavoie pour m'avoir encouragée à entreprendre cette thèse et m'avoir transmis sa passion pour l'enseignement. Merci pour ta confiance, ta patience et tes précieux conseils. J'espère pouvoir poursuivre ta mission d'enseignement.

Mon directeur Jean-François Ethier pour m'avoir présentée à Anita, m'avoir poussée à faire la cotutelle et d'avoir été là quand certaines décisions étaient difficiles à prendre. Merci pour ta confiance, ta persévérance et tes précieux conseils. J'espère pouvoir continuer notre collaboration en recherche.

Merci à vous trois, sans vous la thèse n'aurait pas existé.

Merci à vous, mes collègues du GRIIS :

Adrien Barton pour tes conseils et ta contribution dans mes publications dans le domaine des ontologies appliquées.

Benoit Fraikin pour tes conseils et ta contribution dans mes publications dans le domaine de la logique et les méthodes formelles.

Samuel Dussault pour tes discussions et les échanges sur divers sujets informatiques.

Mélanie Bélanger, Anne-Marie Cloutier et Thibaud Ecarot pour vos encouragements et votre soutien psychologique.

Francis Ouellet, Blaisia Kampire et Ryeyan Taseen pour votre contribution au développement d'OntoRela

Virginie Duceppe-Lamarre pour tes conseils et corrections linguistiques et orthographiques du manuscrit de thèse.

Et tous les professionnels de recherche : David Chuet, Maxime Beauchesne-Jolin et Laura Guglielmetti pour avoir rendu les pauses midi agréables.

Merci à vous, les collègues du Département d'informatique et de la Faculté des sciences de l'Université de Sherbrooke :

Chantal Proulx pour la gestion de mon dossier académique.

Aïda Ouangraoua et Martin Beaudry pour avoir accepté d'être membres de mon comité d'encadrement et les évaluateurs de ma proposition de thèse.

Lise Charbonneau pour la gestion de mon dossier de cotutelle.

Merci à vous, mes collègues de l'HEGP et l'équipe 22 :

Eric Zapletal et Abdel-Ali Boussadi pour vos échanges sur divers sujets informatiques et de recherche en informatique médicale.

Vincent Looten pour m'avoir communiqué le cas d'étude sur les organisations hospitalières et pour avoir rédigé avec moi l'article concerné.

Rosy Tsopra pour tes conseils de rédaction d'article en informatique biomédicale et tes encouragements à la fin de ma thèse.

Christine Romney et Florence Bordu pour les tâches administratives.

David Baudoin, Sandrine Boulet, Hector Countouris, William Digan, Bastien Rance, Alice Rogier et Maxime Wack pour avoir rendu les pauses midi agréables.

Merci au personnel de l'École doctorale et l'Institut de formation doctorale :

Lydie Martonara, Magalie Moulie et Gislaine Montebello pour la gestion de mon dossier académique.

Merci à vous, les membres du jury :

Anne Bergeron, Rémi Bastide, Angela Bonifati et Marc Cuggia

Merci beaucoup d'avoir accepté de lire et d'examiner mon manuscrit de thèse et d'assister à ma soutenance. Vos conseils seront grandement appréciés.

Merci aux organismes et aux groupes de recherches qui ont financé mes études : Fonds de recherche du Québec nature et technologies, Mitacs Globalink, Unité de Soutien SRAP, GRIIS, INSERM 1138 équipe 22.

Enfin, je remercie mes parents d'avoir toujours soutenu mes décisions…

Résumé

Au sein d'une organisation, de même qu'entre des organisations, il y a plusieurs intervenants qui doivent prendre des décisions en fonction de la vision qu'ils se font de l'organisation concernée, de son environnement et des interactions entre les deux. Dans la plupart des cas, les données sont fragmentées en plusieurs sources non coordonnées ce qui complique, notamment, le fait de retracer leur évolution chronologique. Ces différentes sources sont hétérogènes par leur structure, par la sémantique des données qu'elles contiennent, par les technologies informatiques qui les manipulent et par les règles de gouvernance qui les contrôlent. Dans ce contexte, un système de santé apprenant (*Learning Health System*) a pour objectif d'unifier les soins de santé, la recherche biomédicale et le transfert des connaissances, en offrant des outils et des services pour améliorer la collaboration entre les intervenants ; l'optique sous-jacente à cette collaboration étant de fournir à un individu de meilleurs services qui soient personnalisés.

Les méthodes classiques de construction de modèle de données sont fondées sur des règles de pratique souvent peu précises, *ad hoc*, non automatisables. L'extraction des données d'intérêt implique donc d'importantes mobilisations de ressources humaines. De ce fait, la conciliation et l'agrégation des sources sont sans cesse à recommencer parce que les besoins ne sont pas tous connus à l'avance, qu'ils varient au gré de l'évolution des processus et que les données sont souvent incomplètes. Pour obtenir l'interopérabilité, il est nécessaire d'élaborer une méthode automatisée de construction de modèle de données qui maintient conjointement les données brutes des sources et leur sémantique.

Cette thèse présente une méthode qui permet, une fois qu'un modèle de connaissance est choisi, la construction d'un modèle de données selon des critères fondamentaux issus d'un modèle ontologique et d'un modèle relationnel temporel basé sur la logique des intervalles. De plus, la méthode est semi-automatisée par un prototype, OntoRelα. D'une part, l'utilisation des ontologies pour définir la sémantique des données est un moyen intéressant pour assurer une meilleure interopérabilité sémantique étant donné que l'ontologie permet d'exprimer de façon exploitable automatiquement différents axiomes logiques qui permettent la description de données et de leurs liens. D'autre part, l'utilisation d'un modèle relationnel temporalisé permet l'uniformisation de la structure du modèle de données, l'intégration des contraintes temporelles ainsi que l'intégration des contraintes du domaine qui proviennent des ontologies.

Abstract

Within an organization, many stakeholders must make decisions based on their vision of the organization, its environment, and the interactions between these two. In most cases, the data are fragmented in several uncoordinated sources, making it difficult, in particular, to trace their chronological evolution. These different sources are heterogeneous in their structure, in the semantics of the data they contain, in the computer technologies that manipulate them, and in the governance rules that control them. In this context, a *Learning Health System* aims to unify health care, biomedical research and knowledge transfer by providing tools and services to enhance collaboration among stakeholders in the health system to provide better and personalized services to the patient. The implementation of such a system requires a common data model with semantics, structure, and consistent temporal traceability that ensures data integrity.

Traditional data model design methods are based on vague, non-automatable best practice rules where the extraction of data of interest requires the involvement of very important human resources. The reconciliation and the aggregation of sources are constantly starting over again because not all needs are known in advance and vary with the evolution of processes and data are often incomplete. To obtain an interoperable data model, an automated construction method that jointly maintains the source raw data and their semantics is required.

This thesis presents a method that build a data model according to fundamental criteria derived from an ontological model, a relational model and a temporal model based on the logic of intervals. In addition, the method is semi-automated by an OntoRelα prototype. On the one hand, the use of ontologies to define the semantics of data is an interesting way to ensure a better semantic interoperability since it automatically expresses different logical axioms allowing the description of data and their links. On the other hand, the use of a temporal relational model allows the standardization of data model structure and the integration of temporal constraints as well as the integration of domain constraints defines in the ontologies.

Liste des principales abréviations

5FN	5e forme normale
6FN	6e forme normale
BCDM	*Bitemporal conceptual data model (C. Jensen et R. Snodgrass)*
BDR	Base de données relationnelle
BFO	*The Basic Formal Ontology*
DDLM	*C. Date, H. Darwen and N. Lorentzos temporal data model*
FNBC	Forme normale de Boyce-Codd
GRIIS	Groupe de recherche interdisciplinaire en informatique de la santé
µOnto	Modèle ontologique
µRel	Modèle relationnel
µRelT	Modèle relationnel temporalisé
OBO	*The Open biomedical ontologies Foundry*
OntoRel	Modèle ontologique relationnel
OntoRelα	L'application de conversion d'une ontologie en modèle relationnel
OntoRelτ	Modèle ontologique relationnel temporalisé
OntoRelDic	Dictionnaire du modèle OntoRel
OWL	*Ontology Web Language*
PARS3	Plateforme apprenante pour la recherche en santé et services sociaux
SGBDR	Système de gestion de bases de données relationnelles
SSA	Système de santé apprenant (*Learning Health System*)
UBHF	*Unified Bitemporal Historicization Framework*

Glossaire

Temporel :

Entité relative au temps. Par exemple : un type temporel est un type qui représente un domaine de valeurs associées au temps (instant, période, durée) ; un attribut temporel est un attribut dont le type est temporel.

Temporalisé :

Auquel est associé un élément temporel (une annotation temporelle) ; exemple : un attribut temporalisé, un attribut dont la valeur est relative au temps (auquel est associé un autre attribut qui relativise la valeur par rapport au temps), une relation temporalisée, une relation dont les attributs sont temporalisés (individuellement [par VT dans DDLM] ou globalement [par TT dans DDLM]).

Historicisé :

Dont la conservation de l'évolution relative au temps est possible et effective.

Sommaire

Introduction

Les systèmes informationnels sont au centre du processus de prise de décisions au sein de diverses organisations. Plus particulièrement, dans le domaine de la santé, le système de santé apprenant (SSA) — *Learning Health System* — gère le processus de production, de partage et d'accès aux données afin d'assurer de meilleures prises de décision. Les données relatives à la santé d'un patient existent dans plusieurs sources qui sont internes à un même établissement (service de radiologie, les laboratoires, les essais cliniques, etc.) et externes à celui-ci (pharmacie, soins à domicile, cabinet du médecin de famille, etc.). Ces données évoluent fréquemment et sont reliées entre elles selon des relations temporelles et sémantiques. Les cliniciens et les chercheurs désirent avoir une vue unifiée et complète des données d'intérêt afin d'en permettre la réutilisation dans différents types d'activité, notamment par la chronologie des évènements des épisodes de soins. Ces attentes se butent à de nombreux défis : (1) la difficile extraction des données d'intérêt ; (2) l'historicisation des données ; et (3) la disponibilité des données. Premièrement, l'extraction des données d'intérêt demeure difficile attendu le grand nombre de sources non interopérables et souvent non documentées. Or, la définition syntaxique et sémantique des données est primordiale afin de permettre l'automatisation de leur collecte. Deuxièmement, l'historicisation est un enjeu crucial afin de garder une trace explicite et cohérente de l'évolution des données et de leurs liens temporels. Une meilleure modélisation temporelle des données est requise pour améliorer le suivi du changement de celles-ci et comprendre les liens qui existent entre elles et cela, en fonction du temps. Troisièmement, les données qui permettent de répondre à certaines requêtes peuvent être partielles ou même non disponibles dans les sources spécifiées. De plus, la gestion des données manquantes est un autre aspect important à prendre en compte pour offrir une image complète et exacte de l'état des données. À ce jour, la mise en place de modèles de données, prenant en compte les trois aspects mentionnés précédemment, demeure en grande partie manuelle, non reproductible et dépendante des besoins spécifiques. Plus précisément, les différents artéfacts nécessaires pour obtenir un modèle complet et fonctionnel sont obtenus par diverses méthodes non intégrées, en se fondant sur des sources rarement documentées et ciblant des besoins diversifiés qui évoluent très rapidement. En plus, les méthodes existantes sont basées sur des règles *ad hoc* avec un minimum d'historicisation et de gestion des données manquantes. Ces méthodes limitent l'automatisation, la réutilisation, l'expressivité des requêtes et l'exploitation de données temporelles. Les connaissances nécessaires pour comprendre (interpréter

correctement) les données générées doivent être modélisées d'une façon complète, concise et non ambigüe. Plus particulièrement, les ontologies ont joué un rôle très important dans le domaine biomédical dans plusieurs contextes surtout pour la définition sémantique des données.

Une ontologie est une représentation formelle des connaissances ; elle est décrite par des définitions de concepts et leurs associations exprimées par des axiomes vérifiables. Ainsi, l'utilisation des ontologies pour définir la syntaxe et la sémantique des données peut être un moyen intéressant pour assurer une meilleure interopérabilité et une contextualisation qui permet d'enrichir la sémantique des données. Le but du projet de recherche dont la présente thèse rend compte est de contribuer à l'élaboration d'une méthode de construction de modèles de données relationnels temporalisés à partir d'une ontologie.

1
État de l'art

Cette section présente l'état de l'art sur l'élaboration des modèles de données, cela incluant les modèles conceptuels, les modèles du temps, les modèles relationnels et ontologiques temporalisés. Puis, les problématiques non résolues dans la conversion d'une ontologie en modèle relationnel sont décrites. Finalement, la proposition de thèse est présentée.

1.1 Modélisation de données en santé

L'expansion de l'informatisation des données en santé (cliniques, biologiques, administratives, etc.) — couplée à l'évolution des structures organisationnelles, des champs de compétence et de responsabilité — a généré un grand nombre de systèmes développés indépendamment. De ce fait, une masse de données est créée quotidiennement et de façon quasi continue par les systèmes de gestion de dossiers cliniques informatisés, les systèmes de gestion de dossiers médicaux électroniques, les dispositifs médicaux, la recherche, etc. L'accès aux données produites et à leur évolution chronologique est devenu indispensable à la réalisation, l'évaluation et la gestion des activités de recherche, de formation, de mesure de qualité et de médecine préventive [Mate et al. 2015]. Toutefois, ces données sont hétérogènes par leur structure (hétérogénéité structurelle), par la nature des données qu'elles contiennent (hétérogénéité sémantique) [Ethier et al. 2013], par les technologies qui les gèrent (hétérogénéité technologique) et par les règles de gouvernance applicables (hétérogénéité de gouvernance). Par ailleurs, l'association d'estampilles temporelles aux données cliniques est nécessaire à l'analyse et l'interprétation des évènements cliniques et des tâches médicales [Combi et al. 2010]. La gestion des données temporalisées est donc un enjeu crucial afin de conserver la trace de l'évolution des données cliniques et de garantir leur intégrité à travers le temps.

Un grand nombre de solutions indépendantes émergent en permanence pour résoudre des problématiques associées aux différents types d'hétérogénéité et de représentations temporelles. En conséquence, un des défis est la disposition de ressources suffisantes (technologique, humaine et financière) pour intégrer ces solutions en garantissant un niveau de

qualité acceptable, une durabilité et une maintenabilité de cette intégration. Afin de répondre à ce défi, la combinaison de ces solutions dans un système intégré peut s'avérer être une solution intéressante. À noter qu'un SSA a pour objectif d'unifier les soins de santé, la recherche biomédicale et le transfert des connaissances en offrant des outils et des services pour améliorer la collaboration entre les soignants et les chercheurs dans l'optique de fournir au patient de meilleurs services et des services qui lui soient personnalisés [Grossmann et al. 2011][1]. De plus, un SSA s'intéresse à l'évaluation continue des pratiques d'où la nécessité de pouvoir évaluer les activités, les structures et les pratiques à travers le temps [Chambers et al. 2016]. La complexité de mise en œuvre de ce genre de système ne cesse d'augmenter à cause des changements fréquents des processus-métier. La modélisation de données (la construction d'un modèle de données) est au cœur de cette complexité à cause de l'hétérogénéité de l'environnement, du grand nombre d'entités et d'associations à modéliser [Jarke et al. 2014] ainsi qu'en raison de l'absence d'une indication claire du modèle temporel et de la sémantique temporelle utilisés.

1.1.1 Besoins et défis

Bien que chaque établissement puisse avoir un système très efficace d'un point de vue local et pour des besoins précis, le fait de ne pas avoir une image complète du patient crée des difficultés pour fournir des soins optimaux, mener des recherches efficaces et gérer les ressources. Les données de santé sont fortement fragmentées et réparties dans plusieurs établissements de santé en utilisant différentes structures, modèles de connaissances et diverses terminologies pour le même épisode de soins. Dans ce contexte, le résultat net est qu'il est très difficile d'avoir une vision unifiée et complète de l'état clinique et des antécédents d'un individu. À noter que les cliniciens désirent d'ailleurs pouvoir réutiliser des données pour différents types d'analyse et les partager d'une façon exploitable. De plus, le fait de pouvoir recueillir des données temporalisées est un aspect très important pour diverses études cliniques et pour la recherche dans le domaine de santé en général [Combi et al. 2010; Madkour et al. 2016]. Un grand intérêt s'est d'ailleurs développé en ce sens pour l'utilisation secondaire des données de santé provenant de sources multiples afin de réaliser des études cliniques à l'échelle d'une population et développer la médecine personnalisée [Bono et al. 2011; Jensen et al. 2012; Budrionis and Bellika 2016] voire la médecine de précision [Chambers et al. 2016]. D'une part, l'extraction des informations d'intérêt à des fins d'études demeure un grand défi au vu du grand nombre de

[1] "***learning health system*** that is designed to generate and apply the best evidence for the collaborative healthcare choices of each patient and provider; to drive the process of discovery as a natural outgrowth of patient care; and to ensure innovation, quality, safety, and value in health care" [Grossmann et al. 2011] page xi

sources hétérogènes, du grand volume de données, des besoins et des connaissances diversifiées, etc. D'autre part, l'historicisation est devenue un enjeu crucial afin de conserver les traces de l'évolution des données et d'améliorer la prise de décision clinique.

Depuis les années 1980, diverses architectures ont été développées, pour intégrer physiquement ou virtuellement différentes sources de données, cela afin d'uniformiser l'accès tout en permettant de concilier cette hétérogénéité [Smith et al. 1981; Wiederhold 1992; Webb and Will 2001] : bases de données fédérées [Heimbigner and McLeod 1985], entrepôts de données [Kimball 2004; Inmon 2002], accès guidé par les ontologies (*ontology-based data access*) [Calvanese et al. 2007] et *data lake* [Hai et al. 2016].

Cependant, il ne suffit pas de faire l'unification des données des différentes sources en les important dans un même entrepôt ou dans un *data lake* sans s'intéresser simultanément à la sémantique, à la structure, à la technologie [Venot et al. 2013] et à la gouvernance :

- ◊ l'interopérabilité sémantique est nécessaire à l'interprétation unifiée des données selon un sens et une terminologie (encodage) compréhensible par l'humain et la machine ;
- ◊ l'interopérabilité syntaxique (comprenant le modèle logique) est nécessaire à l'accès et au traitement efficace, voire efficient, des données ;
- ◊ l'interopérabilité technologique est requise pour la mise en œuvre de l'acheminement des données entre diverses applications (ou systèmes) selon divers protocoles d'échange ;
- ◊ l'interopérabilité de gouvernance est un préalable au respect des procédures mises en place pour contrôler la collecte de données et leur utilisation ainsi que le respect des obligations éthiques et légales : le protocole d'accès aux données, des mécanismes d'anonymisation, et des techniques sécurisées d'échange de données.

Par conséquent, la construction d'un modèle de données commun est nécessaire à l'intégration de plusieurs sources de données [Chromiak and Stencel 2014]. Les solutions analogues aux *data lake* ne contribuent pas à construire un tel modèle. Quant aux méthodes existantes de construction de modèles de données, elles sont généralement fondées sur de vagues règles de pratique non automatisables [Khnaisser et al. 2015] et impliquent d'importantes mobilisations de ressources humaines pour garantir l'interopérabilité.

Plusieurs défis découlent de ce contexte, notamment :

- ◊ La construction du modèle de données est largement basée sur les processus ou sur les besoins de chaque établissement, ce qui nécessite une conciliation des besoins, la « reconstruction » fréquente d'une grande partie du modèle de données et compromet la pérennisation des applications développées.

- Les connaissances nécessaires pour interpréter correctement les données sont souvent absentes ou difficiles à retracer. La sémantique des données est rarement documentée et est souvent confinée à la couche applicative générant les données, mais surtout confinée aux personnes connaissant de première main lesdites données.
- Le modèle temporel utilisé (et sa sémantique) est rarement documenté et, souvent, il n'est pas appliqué uniformément ni systématiquement. Les modèles formels sont rarement utilisés et la sémantique est fortement liée au contexte et aux processus qui engendrent les données.
- La modélisation et l'intégration des données sont faiblement automatisées. La construction du modèle de données est souvent réalisée en grande partie manuellement, c'est-à-dire sans assistance logicielle qui permette de vérifier la cohérence et la complétude du modèle.
- La traçabilité des données et de leur sémantique en regard de la création, de la modification, de l'utilisation et la transformation est rarement prise en charge de manière systématique. Les utilisateurs primaires[2] sont dépositaires de la sémantique implicite des données ; or cette sémantique est essentielle à la modification, la validation, l'exploration et le traitement adéquat de ces données. Par contre, les utilisateurs secondaires[3] n'ont pas la sémantique implicite, ce qui induit un grand risque de réinterprétation des données.

Donc, un modèle de données commun dans le cadre d'un SSA doit fondamentalement permettre de :

- gérer l'hétérogénéité sémantique, structurelle, technologique et de gouvernance ;
- gérer la traçabilité temporelle des données incluant leurs créations et leurs modifications ;
- gérer l'évolution des sources en minimisant l'impact sur l'existant.

1.1.2 Exigences

Un modèle et une méthode de construction sont requis pour formaliser la structuration et la sémantique des données temporalisées. Le modèle de données doit permettre de partager et communiquer des données entre divers établissements du réseau de la santé afin d'avoir une vue transversale de « l'histoire de santé » du patient ou d'une population. Pour ce faire, il est nécessaire d'établir certaines exigences et de s'assurer de les satisfaire.

[2] Un utilisateur primaire est une personne ou une machine (système d'information) qui utilise les données spécifiquement dans le contexte pour lequel les données ont été créées.

[3] Un utilisateur secondaire est une personne ou une machine (système d'information) qui utilise des données pour une fin autre que celle pour laquelle les données ont été initialement créées.

Garantie d'intégrité

La méthode de construction du modèle doit garantir l'intégrité des données et donc fournir des contraintes d'intégrité et un mécanisme de vérification et validation automatisable.

Les données provenant du modèle seront utilisées par divers algorithmes pour alimenter différents types d'analyse, donc les données doivent être fiables, complètes et cohérentes. L'intégrité des données est obtenue par la définition des contraintes nécessaires et suffisantes pour définir puis maintenir la cohérence des données. Corolairement à cela, l'intégrité traite aussi de la structuration des données comme moyen de définir et d'assurer cette cohérence, et traite donc en définitive de la structure du schéma lui-même.

En regard de la théorie relationnelle, la théorie de la normalisation relationnelle remplit ce rôle avec les formes normales de Boyce-Codd (FNBC) [Codd 1979], de projection-jointure (5FN) [Fagin 1979] et de projection-jointure triviale (6FN) [Date et al. 2003], cela particulièrement pour les modèles temporalisés. Par ailleurs, les solutions dimensionnelles telles que l'étoile (*star schema*) et le flocon (*snowflake schema*) n'apportent pas de solution à cet égard [Adamson 2010; Jiang 2011].

Finalement, il serait souhaitable de distinguer clairement les contraintes intrinsèques à la structure, soit indépendantes du domaine, et celles provenant du domaine, donc nécessitant un apport externe, idéalement en provenance du modèle de connaissance ou de l'analyse assurant la supervision de la construction.

Automatisation de la construction du modèle de données

La méthode de construction du modèle de données doit pouvoir s'effectuer automatiquement sur la base d'algorithmes s'appuyant sur une théorie solide ; elle doit aussi pouvoir être configurée par l'architecte de la base de données qui mettra en œuvre le modèle.

Un niveau d'automatisation est nécessaire pour tenir compte des ressources humaines limitées qui sont disponibles : l'expertise de l'architecte de bases de données, la qualité, le temps et le cout du projet. L'automatisation guidée (supervisée) de la construction d'un modèle de données consiste à fournir des suggestions et des vérifications à l'aide d'algorithmes fondés sur diverses théories afin d'accélérer et de minimiser les erreurs des activités de construction. Pour faciliter l'automatisation de la construction d'un modèle de données, il serait souhaitable de distinguer clairement les contraintes intrinsèques à la structure (donc indépendantes du domaine) de celles provenant du domaine (donc nécessitant un apport externe qui soit idéalement en provenance du modèle de connaissance lui-même ou de l'analyse qui en a été faite).

Sémantique unique selon un modèle formel uniforme

Le modèle de données doit avoir une sémantique unique et explicite selon un modèle formel uniforme afin d'être exploitable par différents processus d'intégration et d'interrogation de données, lesquels étant non établis à l'avance.

Pour éviter les biais dans les analyses des données, il est à noter qu'une sémantique unique et explicite pour une donnée est nécessaire afin de garantir une interprétation uniforme à travers les utilisations qui en sont faites. De plus, cette sémantique doit être documentée et partageable entre les utilisateurs. Ces aspects sont d'ailleurs naturellement associés à la création d'ontologies formelles. Plus spécifiquement, une ontologie fournit suffisamment d'informations sémantiques et contextuelles pour assurer une réutilisation fiable des données en dehors du champ restreint de son application d'origine [Calvanese et al. 2010]. La définition d'un modèle de données à partir d'une ontologie est donc un moyen de modélisation qui peut répondre à ces exigences. D'une part, l'utilisation des ontologies est de plus en plus préconisée pour définir formellement un modèle de connaissance et faciliter le partage de la sémantique [Sugumaran and Storey 2006; Thenmozhi and Vivekanandan 2012]. Plus particulièrement, les ontologies jouent un rôle très important dans le domaine biomédical dans plusieurs contextes, incluant la définition sémantique et l'intégration des données [Mate et al. 2015] de même que la classification de patients [Haendel et al. 2018]. De plus, deux répertoires d'ontologies biomédicales existent (*Bioportal*[4], et *OBO foundry*[5]) et contiennent à eux deux environ 800 ontologies et terminologies. D'autre part, les ontologies permettent de réduire l'effort requis pour effectuer l'arrimage entre une définition sémantique et une ou plusieurs données [Mate et al. 2015] étant donné qu'elle peut exprimer d'une façon exploitable automatiquement différents axiomes logiques permettant la description de concepts.

La méthode doit permettre l'utilisation d'une ontologie pour la modélisation, l'arrimage (*mapping*), l'exploitation (ajout, suppression et modification des données) et l'exploration (interrogation et analyse). Ainsi, l'utilisation des ontologies pour guider l'élaboration d'un modèle de données est un moyen intéressant pour assurer une meilleure interopérabilité et contextualisation [Laleci et al. 2013].

Fiabilité

Le modèle de données doit être mis en œuvre par un système fiable qui permet de garantir intrinsèquement l'intégrité des données dans un contexte transactionnel concurrent.

4 https://bioportal.bioontology.org
5 http://www.obofoundry.org

Un modèle de données repose sur une structure (formée de plusieurs composants) et sur un langage de traitement et d'interrogation qui permet de manipuler (ajouter, supprimer, modifier) les instances de cette structure. Cette structure doit permettre de représenter les données, garantir leur sémantique et leur intégrité afin d'éviter que chaque utilisateur (p. ex. *data scientist*) doive faire le « tri » avant chaque étude. Pour ce faire, la structure doit d'une part être indépendante de l'utilisation (c'est-à-dire être d'un projet spécifique) et des structures des sources (car autant l'utilisation que la structure des sources changent régulièrement et cela parfois de façon imprévisible). D'autre part, la structure doit permettre de gérer les données absentes [Darwen 1998] et les données temporelles d'une façon neutre et uniforme. À la lumière de la technologie récente, le modèle relationnel reste le modèle le plus utilisé pour ce genre de structuration. De même, les systèmes de gestion de bases de données relationnelles (SGBDR) demeurent le moyen le plus efficace pour accéder aux données par l'entremise d'une ontologie [Kontchakov et al. 2011; Spanos et al. 2012].

À noter qu'en général, un SGBDR est un système approprié pour modifier et interroger un grand volume de données de manière fiable : l'intégrité des données est garantie par la gestion des transactions concurrentes grâce aux propriétés ACID[6] et une performance acceptable est garantie grâce aux nombreux algorithmes d'optimisation de requêtes [Spanos et al. 2012]. À ce jour, les implémentations des systèmes de gestion de données RDF (*triple store*) ne sont pas assez matures pour avoir la même efficience qu'un SGBDR en ce qui concerne la gestion de grands volumes de données [Wylot et al. 2018] et le contrôle d'accès [Kirrane et al. 2017].

Traçabilité temporelle uniforme

La méthode de construction du modèle de données doit se baser sur une méthode d'historicisation qui permet une structuration uniforme facilitant la gestion de l'intégrité et l'expressivité des requêtes temporelles.

Une traçabilité temporelle appropriée est nécessaire pour garantir l'intégrité de l'évolution des données, soit l'intégrité temporelle. L'association du temps aux évènements cliniques facilite l'analyse temporelle et l'interprétation des données dans la majorité des départements d'un établissement hospitalier (cardiologie, oncologie, psychiatrie, soins intensifs, médecine interne, etc.) et dans différentes tâches médicales (diagnostics, administration thérapeutique, protocoles cliniques, administratifs, etc.) [Combi et al. 2010]. Il convient de souligner que l'historicisation du modèle de données est un processus de transformation d'un schéma non temporalisé à un

6 Atomicité, cohérence, isolation et durabilité : ensemble de propriétés que garantisse un SGBD lors d'une transaction [Elmasri and Navathe 2016].

schéma temporalisé incluant la définition des contraintes temporelles et les règles de modifications dont le but est d'assurer une traçabilité cohérente et complète des changements des données. Ainsi, l'historicisation doit se baser sur un modèle temporel fondé qui assure une représentation et une sémantique temporelle unifiée pour les types de données et les opérations applicables. Idéalement, ces règles doivent être définies de façon indépendante du domaine et en fonction des dimensions de temps génériques. Plus particulièrement, deux telles dimensions de temps sont déjà bien étudiées dans la littérature : le temps de transaction et le temps de validité [Dyreson et al. 1994].

Mise en œuvre conjointe pour la médiation et l'entreposage

Le mécanisme de construction du modèle de données devra soutenir autant une mise en œuvre par médiation que par entreposage.

L'hétérogénéité des sources imposera encore longtemps une approche par médiation (ne serait-ce qu'en raison de l'hétérogénéité des règles de gouvernance). D'un autre côté, au sein d'une organisation, voire même d'un établissement tel qu'un centre hospitalier, le volume des données en cause nécessite des performances qui ne sont présentement accessibles que par l'entreposage centralisé. Il apparait contreproductif d'imposer l'intermédiaire de la médiation à la structure d'entreposage lorsque des sources multiples, hors organisation et donc vraisemblablement accessibles par médiation, devront être mises à contribution par une requête. Dans le cas de l'entreposage, le plein accès en consultation, en insertion, en retrait et en mise à jour devra pouvoir être mis en œuvre de façon efficace, voire efficiente, cela étant dans un contexte transactionnel qui préserve les propriétés ACID.

Accessibilité des technologies retenues pour une majorité d'établissements

La méthode de construction doit être mise en œuvre par des technologies accessibles à une majorité d'établissements.

Des technologies accessibles à une majorité d'établissements sont nécessaires pour favoriser l'appropriation à moindre cout. Pour ce faire, il s'agit principalement de se baser sur des technologies fiables, connues et libres, cela dans l'optique d'en permettre une adoption plus rapide.

1.1.3 Solutions

Deux types de solutions ont été envisagés : (1) les solutions à base de modèles de données communs dans le domaine de la santé, et (2) les solutions obtenues par construction. Elles sont présentées ci-après, ce qui permettra d'ensuite définir la portée de la présente thèse.

1.1.3.1 Solutions à base de modèles de données communs en santé

Les modèles de données communs comme *i2b2*, *OMOP* et *PCORnet* [Weeks and Pardee 2019] sont très répandus grâce à l'accessibilité des outils qui permettent de les exploiter et de les enrichir de l'extérieur. Cependant, dans plusieurs articles, des préoccupations ont été exprimées au sujet de la perte et de la distorsion des informations qui peuvent survenir lorsque les données sont converties et regroupées dans des entrepôts issus desdits modèles [Ceusters and Blaisure 2017]. De plus, il y a plusieurs problématiques au niveau de la sémantique et la structure qui limitent l'utilisation optimale de ces modèles ; parmi ces problématiques se retrouve l'absence de sémantique explicite unifiée à la structure de données. Par conséquent :

- ◊ Il est difficile de s'assurer que l'instanciation de deux modèles possède la même sémantique. Ainsi, les résultats de deux applications portant sur la même instance ou plusieurs instances sont difficilement comparables. Par exemple, le catalogue des données d'*i2b2* (nommé ontologie) n'est pas un modèle de connaissance qui permet d'établir une sémantique formelle des données. Ce dernier est personnalisé pour chaque installation et présente simplement une annotation terminologique de la structuration des données à l'intérieur de l'entrepôt. De plus, dans le modèle *i2b2*, deux faits (`observation_fact`) d'un même processus (`concept_id`) peuvent avoir différents nombres d'attributs (`modifier_id`). Un autre exemple : la définition du terme « visite » (`encounter`, `visit_occurence`) varie selon le modèle et selon l'établissement ; une explication détaillée à cet effet est présentée dans l'article [Danese et al. 2019].
- ◊ Il est difficile de s'assurer que les mêmes contraintes sont mises en place. L'intégrité des données est compromise par l'absence de contraintes pour valider les données (et leur interprétation). Ainsi, les requêtes et leurs résultats sont rarement reproductibles et sémantiquement vérifiables [Ceusters and Blaisure 2017; Danese et al. 2019]. Dans beaucoup de cas, une requête effectuée sur ces modèles nécessite un prétraitement ou un post-traitement des données résultantes pour qu'elles puissent être réutilisées par des outils d'analyses avancées [Mate et al. 2017]. Dans le modèle *OMOP*, les restrictions sur les cardinalités des attributs d'une entité sont souvent absentes ce qui induit des comptes erronés dans certains contextes [Ceusters and Blaisure 2017].
- ◊ Il est difficile de conserver la trace de l'évolution des données. La structure non normalisée est inadéquate et insuffisante pour reconstruire une chronologie (*timeline*) cohérente et complète de l'état du patient [Defossez et al. 2014]. Par exemple, dans un projet qui

consiste à construire des trajectoires de soins, il est difficile, voire impossible, d'extraire les évènements assez détaillés pour permettre une construction complète et optimale.

En pratique, chaque source alimente l'entrepôt par un processus ETL qui lui est propre. À cause de l'absence d'un modèle de connaissances unique de haut niveau, la sémantique de la structure et des données peut donc être disparate au sein d'un même entrepôt, *a fortiori* entre deux entrepôts.

En résumé, dans la plupart des modèles communs de données, la sémantique est découplée de la structure et l'arrimage sémantique entre les sources et le modèle commun doit alors être réalisé à l'extérieur de la base de données d'une manière *ad hoc*. Par conséquent, il est difficile d'automatiser la vérification et la validation des requêtes de même que d'automatiser l'évolution des données et conséquemment de garantir l'intégrité et la sémantique des données. En conclusion, ces modèles de données (et les outils qui leur sont associés) sont très intéressants pour faciliter, synthétiser et agréger les données pour un projet spécifique, cela bien que ces modèles ne soient pas suffisants dans le cadre d'un SSA multi domaine.

1.1.3.2 Solutions par construction

Une autre gamme de solutions semble s'imposer à la lumière des exigences formulées : un modèle de données relationnel temporalisé (1) dérivé d'une ontologie, (2) construit en respectant la structure uniforme définie par une méthode d'historicisation, et (3) mis en œuvre par un système fiable. Par conséquent, une méthode de construction doit être élaborée pour automatiser et configurer la génération d'un tel modèle en utilisant des technologies accessibles à une majorité d'établissements dans le domaine de la santé.

Nous croyons qu'un SSA tirerait de grands bénéfices d'une telle solution [Ethier et al. 2017]. Dans la perspective de satisfaire ces exigences, le groupe de recherche interdisciplinaire en informatique de la santé (GRIIS) a élaboré PARS3[7] qui est une plateforme apprenante pour la recherche en santé et services sociaux au Québec. PARS3 suit une approche qui est fondée sur les connaissances (les ontologies) pour construire un modèle de données plutôt que de construire (ou proposer) un modèle de données et de tenter de lui associer une interprétation sémantique *a posteriori*.

1.1.4 Portée

La présente thèse s'insère dans le projet PARS3 et sa portée est l'élaboration d'une méthode de construction d'un modèle de données relationnel temporalisé dérivée à partir d'une ontologie.

[7] https://griis.ca/pars3/

La méthode doit être basée sur des théories fondées et supervisées par un architecte de modèle de données pour éviter de recourir à des décisions *ad hoc* pour garantir l'interopérabilité. De plus, la méthode et le modèle générés doivent satisfaire les exigences d'intégrité sémantique et structurelle, de traçabilité temporelle, d'automatisation et d'accessibilité. La méthode sera évaluée relativement au domaine de la santé pour en prouver la pertinence et la faisabilité. D'autres domaines, tels que l'écologie, l'économie, la finance et les télécommunications pourront également tirer bénéfices de la méthode proposée.

1.2 Élaboration d'un modèle de données

L'élaboration d'un modèle de données s'effectue traditionnellement en trois étapes : analyse, spécification et implémentation. À noter que des artéfacts interreliés sont produits à chaque étape, selon une architecture trischématique, en utilisant divers modèles (modèle conceptuel, modèle du temps, modèle relationnel) et différents processus (conversion entre modèles, historicisation).

1.2.1 Architecture trischématique

L'architecture trischématique des modèles de données[8] comporte trois niveaux d'abstraction [Elmasri and Navathe 2016] (Figure 1) :

- Le schéma conceptuel (aussi nommé schéma externe) décrit les concepts d'intérêt du monde réel selon la vision de l'utilisateur ou d'un expert du métier (p. ex. entité-association, orienté-objet, ontologie) [Deputy Chief Information Officer 2015].
- Le schéma logique décrit l'organisation et les contraintes applicables aux données (p. ex. relationnel, réseau, hiérarchique) [Deputy Chief Information Officer 2015].
- Le schéma physique (aussi nommé schéma interne) décrit la structure de stockage et les méthodes d'accès [Deputy Chief Information Officer 2015].

Le schéma conceptuel est construit par un analyste-métier afin de représenter la vision des utilisateurs. Ce dernier est utilisé par un architecte de bases de données pour construire le schéma logique. Par la suite, le schéma logique est utilisé comme modèle de données pour construire et maintenir une base de données ainsi que des applications externes. Le schéma

[8] L'architecture trischématique est définie selon l'architecture ANSI/SPARC (*American National Standards Institute*, *Standards Planning And Requirements Committee)* des SGBD, apparue au début de 1970.

physique est géré par le SGBD(R) et le système d'exploitation. Cette architecture demeure largement utilisée pour la construction de systèmes d'information.

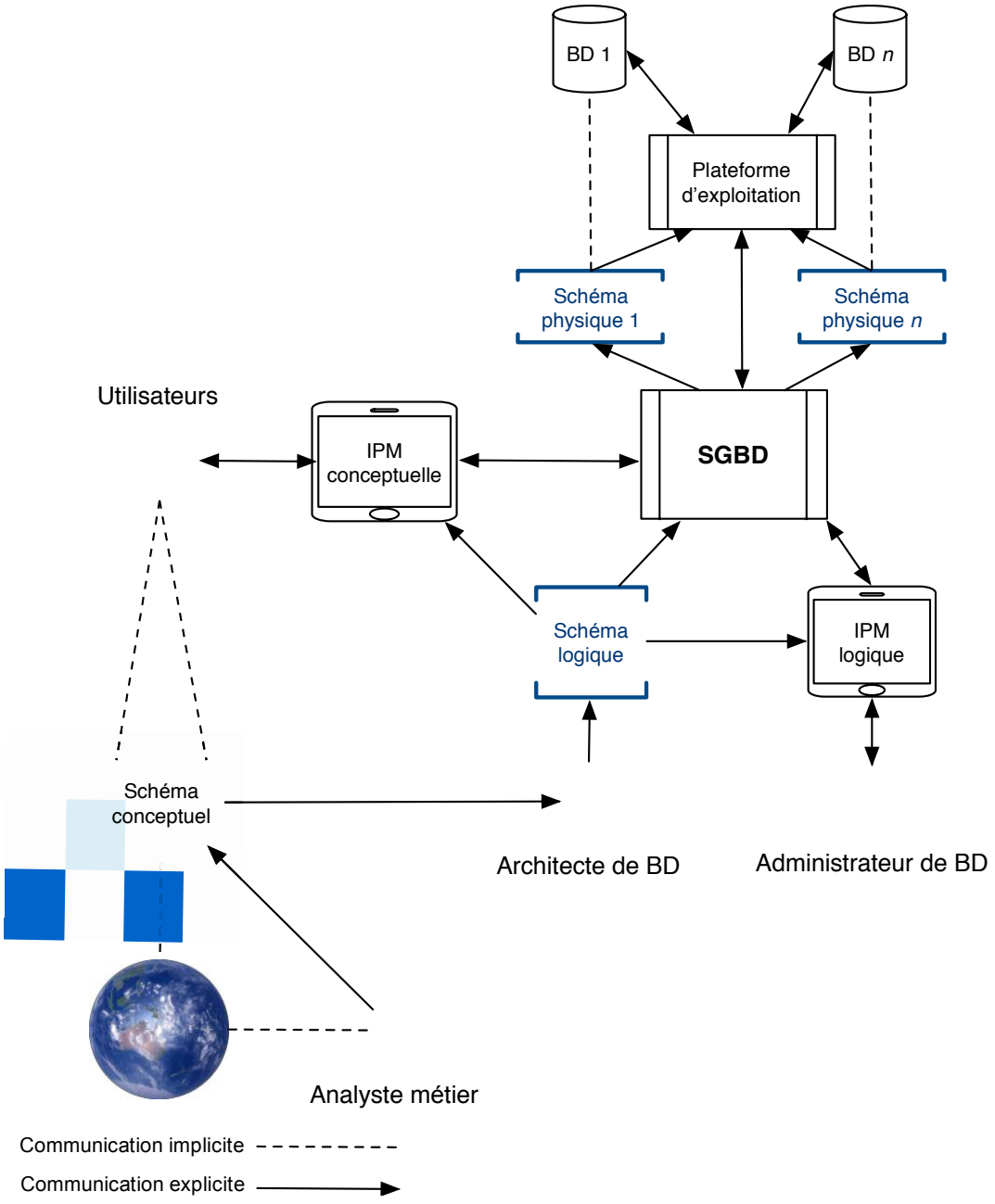


Figure 1. Processus classique de modélisation de données[9]

1.2.2 Approches d'analyse

Dans des environnements où les données sont hétérogènes, la construction du modèle conceptuel est une tâche complexe. Plusieurs approches d'analyse ont été élaborées, en étant essentiellement issues de la communauté des entrepôts de données, pour guider la construction

[9] Adaptation d'une figure présentée dans les notes du cours IFT187 de l'Université de Sherbrooke.

d'un schéma conceptuel commun. Ces approches utilisent différents types d'intrants pour arriver à construire le modèle de données [Romero and Abelló 2009] :

- ◊ L'approche basée sur les fournisseurs des données (*data-driven approach*) consiste à analyser la structure des données produites par les fournisseurs de données (les sources de données).
- ◊ L'approche basée sur les exigences (*requirement-driven approach*) consiste à collecter et analyser les exigences des utilisateurs et des processus.
- ◊ L'approche hybride est une combinaison des deux approches précédentes auxquelles s'ajoute une étape d'arrimage (et réconciliation) entre les sources de données et les exigences.
- ◊ Les approches fondées sur les connaissances (*knowledge-driven approach*) consistent à dériver un modèle de données à partir d'un modèle de connaissances, indépendamment des sources et des exigences.

Chacune des approches ci-dessus présente des avantages et des inconvénients. Dans ce contexte, il convient de souligner que l'utilisation d'une approche basée sur les fournisseurs et les exigences n'est pas très favorable dans le cadre d'un environnement hétérogène et en fréquente évolution. D'une part, l'utilisation d'une approche basée sur les fournisseurs amène une dépendance structurelle par sources, ces dernières étant souvent incompatibles entre elles. De plus, cela peut souvent entrainer des modifications majeures au modèle lors de l'ajout d'une source. Par conséquent, l'arrimage de toutes ces sources — qui est sans cesse à recommencer — est pratiquement impossible à mettre en œuvre dans un contexte de ressources restreintes, voire déclinantes. D'autre part, l'utilisation des exigences n'est pas pratique en raison de la diversité des utilisateurs finaux et de la complexité d'automatisation des exigences [Romero and Abelló 2010]. Avec les besoins continus d'amélioration de la prise de décision, particulièrement dans les domaines cliniques et clinico-administratifs, de nouvelles exigences vont apparaitre et les exigences existantes peuvent être appelées à changer ou même à devenir inappropriées. Par conséquent, le modèle doit contenir toutes les données disponibles, quelles que soient les exigences qui prévalent lors de la création du modèle initial. Ainsi, l'approche hybride réconcilie les structures des données sources avec les exigences, bien qu'elle demeure plus difficile à mettre en place et qu'elle a évolué à cause des compromis *ad hoc* dépendants des besoins du moment. Finalement, aucune de ces trois dernières approches ne donne accès (par elles-mêmes) à la sémantique explicite, car les prédicats associés à un modèle de données sont toujours implicites et ne peuvent être construits ni à partir du schéma source ni à partir des exigences. Contrairement à cela, une approche basée sur la connaissance peut pour sa part

fournir un modèle de données où la sémantique est explicite et indépendante des applications sous-jacentes [Spanos et al. 2012]. Par conséquent, la formulation et le calcul des requêtes sont davantage systématiques et fiables. Cependant, les méthodes actuelles qui utilisent l'approche hybride ne permettent pas encore de résoudre un des inconvénients, soit la non-préservation du lien entre le modèle de connaissance et le modèle de données [Khouri and Ladjel 2010].

Finalement, un modèle de données, et cela particulièrement dans le domaine clinique, ne peut se limiter à offrir une agrégation de données des sources, complétée d'une synthèse. Les exigences d'agrégation et de synthèse ne sont pas toutes connues à l'avance, sont souvent incomplètes, varient dans le temps au gré de l'évolution des processus et des besoins en information [Khnaisser et al. 2015]. Afin d'obtenir un modèle réutilisable et riche, il est recommandé de maintenir un arrimage aux données « brutes » des sources afin de permettre la définition d'autant d'« images » adéquates en fonction des problèmes. Pour cela, une modélisation générale (un modèle de connaissances) de haut niveau est requise afin de représenter adéquatement l'ensemble des données brutes, ou du moins celles qui peuvent être arrimées au modèle de connaissance. Cependant, pour obtenir un arrimage de qualité, il est nécessaire de convenir de certaines exigences et de les satisfaire, plus particulièrement en regard de l'intégrité et de l'historicisation.

1.2.3 Historicisation d'un modèle de données

Dans la littérature, les termes temporalisation et historicisation s'entremêlent et se confondent fréquemment. Par souci de clarté, nous avons retenu les définitions suivantes :

- La temporalisation consiste à définir, selon un modèle du temps, les concepts primitifs comme étant des types temporels, et à définir les opérations fondamentales pour la représentation et la manipulation du temps.
- L'historicisation consiste à définir une méthode de structuration d'un modèle de données pour conserver et gérer adéquatement l'évolution des données dans le temps.

Par exemple, la proposition dans [Date et al. 2014] peut être divisée en deux :

- L'algèbre relationnelle temporelle (*Temporal relational algebra*) [Lorentzos and Johnson 1988] est une temporalisation de l'algèbre relationnelle qui utilise l'algèbre des intervalles et la représentation compacte des relations dont l'intégrité est maintenue par les quatre contraintes de non-contradiction, de non-redondance, de non-circonlocution et de compacité à l'aide des opérateurs *pack (fold), unpack (unfold)* et *using*.

◊ La méthode d'historicisation est déclinée selon la couverture (ouvert sur le futur [*since*], fermé [*during*], ou les deux) et la dimension (validité [*stated*], transaction [*log*] ou les deux).

Un modèle de données non historicisé contient les données courantes sans aucune trace explicite de leur évolution. Par contre, un modèle de données historicisé sauvegarde les données courantes, passées — et voire futures — en garantissant leur cohérence et permet la reconstitution (sans perte) de leur évolution (de leur histoire). Chaque donnée historicisée est associée à un ou plusieurs attributs temporels et chaque contrainte est définie de telle sorte à garantir l'intégrité temporelle des données.

L'historicisation est rapidement devenue un enjeu crucial pour garder les traces de l'évolution des données et améliorer la prise de décision clinique. Afin de permettre une traçabilité appropriée des données, le modèle de données doit être construit sur la base d'un modèle du temps fondé assurant une représentation et une sémantique temporelle unifiées pour des types de données et les opérations applicables. De plus, une méthode de construction d'une telle représentation est nécessaire pour permettre une structuration uniforme et solide facilitant le contrôle de l'intégrité des données et l'expressivité des requêtes temporelles. Cependant, la mise en place et la gestion des schémas historicisés sont complexes et les méthodes existantes possèdent des limites, notamment :

◊ **la représentation est incomplète** relativement aux référentiels temporels et absences de la gestion de l'indétermination du passé et l'indétermination complète ;

◊ **la structure est inadéquate** relativement à la préservation de l'intégrité (les contraintes ne sont pas définies) et l'unification de la sémantique ;

◊ **les procédures de modification ne sont que partiellement définies (quand elles le sont),** et ce faisant la maintenance des données est le plus souvent effectuée au cas par cas par les applications ;

◊ **les outils disponibles sont incomplets**, ils n'offrent pas toutes les fonctionnalités nécessaires à la définition et à la maintenance d'un schéma historicisé.

Les solutions à ces problèmes sont toutes partielles et demeurent en grande partie spécifiques à un domaine d'application en étant difficilement extensibles, difficiles à comprendre (à s'approprier), couteuses en temps et en ressources. Jusqu'à présent, il n'y a aucune étude portant sur la spécification d'une méthode de construction d'un modèle de données à partir d'une ontologie fondée sur une méthode d'historicisation qui a été recensée dans la littérature [Khnaisser et al. 2015; Khnaisser et al. 2018].

1.3 Modèles conceptuels

Plusieurs modèles conceptuels de données ont été proposés pour organiser les entités et leurs associations et définir les restrictions qui s'y appliquent dans un domaine particulier. La modélisation conceptuelle demeure un sujet très actif [Storey et al. 2015]. Un bref résumé est présenté ci-dessous (Figure 2).

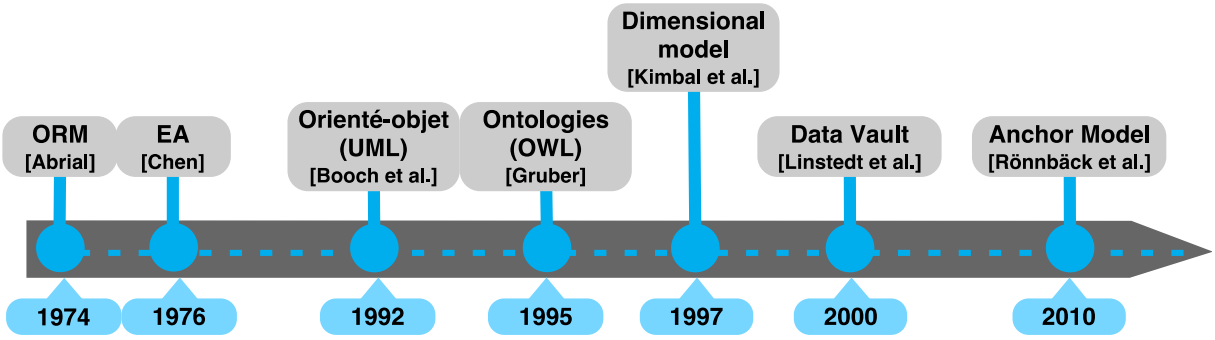


Figure 2. Bref résumé des modèles conceptuels de données

Un des premiers modèles conceptuels de données, le modèle objet-rôle (*Object-role model*) [Abrial 1974; Senko 1975; Falkenberg 1976] est défini pour la modélisation sémantique de système d'information selon le principe suivant [Halpin 1998] (traduction libre) : « Pour être corrects, clairs et adaptables, les systèmes d'information sont mieux spécifiés d'abord au niveau conceptuel, en utilisant des concepts et un langage que les gens peuvent facilement comprendre. ». Il est composé d'objets et de rôles (des relations binaires) pour décrire un univers du discours en utilisant le langage naturel. Le modèle est connu en Europe sous le nom de la méthode d'analyse d'information NIAM (*Natural Language Information Analysis Method*) [Wintraecken 1990]. De plus, ce modèle a fortement influencé le modèle orienté-objet et le modèle ontologique (les ontologies OWL).

Le modèle entité-association (*entity-relationship*) [Chen 1976] est un des modèles les plus utilisés grâce à la simplicité de ses composants et de sa notation graphique. Il est composé d'entités, d'attributs et d'associations (relations munies de participations). Ce modèle est devenu la base de plusieurs méthodes, dont entre autres : MERISE (méthode d'étude et de réalisation informatique pour les systèmes d'entreprise) [Rochfield et al. 1989] et IDEF1X[10] (*Integration Definition for information modeling*) [Bruce 1992]. Cependant, le modèle est utilisé seulement durant la conception et n'est plus réutilisé pour le développement ou la

[10] http://www.idef.com/idef1x-data-modeling-method/

maintenance. Par conséquent, ce modèle fonctionne bien tant que l'environnement opérationnel d'une base de données reste fermé et relativement stable.

Ensuite, pour faciliter la communication les données et les applications conçues, des principes de la programmation orientée-objet des modèles de données orientés-objet ont été créés [Rumbaugh 1991; Booch et al. 1997]. Le langage UML (*unified modelling language*) [Booch et al. 1997], qui est basé sur le paradigme orienté-objet, est un langage graphique utilisé pour définir plusieurs artéfacts d'un système informatique, notamment pour un modèle de données avec le diagramme de classe.

Avec l'augmentation de la complexité des systèmes d'information, la variété des données et des contextes d'utilisation, les modèles EA et UML n'apportent pas de solutions satisfaisantes au point de vue de l'expressivité sémantique et structurelle, cela malgré leur popularité et la présence de divers algorithmes de conversion en schéma relationnel qui sont bien établis [Elmasri and Navathe 2016]. Ainsi, les ontologies ont été proposées [Gruber 2009] dans le but de formaliser la définition des données d'un domaine pour faciliter le partage et l'intégration indépendamment des structures des applications [Bellatreche et al. 2010; Aadil et al. 2016]. Les ontologies sont définies comme suit dans [Guarino 1997] :

> *Ontologies are agreements about shared conceptualizations. Shared conceptualizations include conceptual frameworks for modelling domain knowledge; content-specific protocols for communication among inter-operating agents; and agreements about the representation of particular domain theories.*

Autrement dit, l'utilisation des ontologies permet de définir un modèle commun indépendant des sources des données ou des besoins spécifiques, ce qui favorise l'interopérabilité. Depuis, les ontologies sont utilisées dans plusieurs activités : l'acquisition et le stockage de données [Rodríguez-Muro et al. 2013], ainsi que dans l'accès aux données [Metke-Jimenez et al. 2018]. Par contre, les algorithmes de conversion existants pour les ontologies, depuis 2004 [Gali et al. 2004], demeurent non-consensuels et possèdent des limitations qui empêchent l'utilisation de la richesse de l'expressivité des axiomes [Khnaisser et al. 2018].

Plus tard, avec l'informatisation de divers départements dans une entreprise, la fragmentation des données dans plusieurs sources de données a rendu la prise de décisions plus difficile. Pour répondre à ce besoin, le modèle dimensionnel a été conçu pour faciliter l'analyse des données en agrégeant les données dans une seule structure. Le modèle dimensionnel [Kimball 2013] utilise une approche essentiellement basée sur les processus-métier et sur les exigences de prise de décision ; ce sont ces processus et ces exigences qui sont utilisés pour identifier les « faits »

et les « dimensions » requis par ce modèle ; ce sont également des processus et ces exigences qui sont utilisés pour agréger les données des « mesures » lors des requêtes et des analyses.

Pour répondre aux problématiques du modèle dimensionnel qui concernent l'évolutivité de la structure et la mise à jour des données, deux modèles de modélisation ont été proposés : *Data vault*[11] [Linstedt and Olschimke 2016] et *Anchor modeling*[12] [Rönnbäck et al. 2010]. Ces modèles reposent principalement sur la modélisation agile qui consiste à construire d'une façon itérative un modèle de données en minimisant l'impact des changements sur le modèle existant. Cependant, dans le modèle *Data vault* la sémantique demeure fortement liée au contexte et aux processus qui engendrent les données, donc dépend des sources. Par contre, le modèle *Anchor* reste ouvert sur l'approche prise pour la conception du modèle. De plus, la structure proposée par les deux modèles minimise la différence entre le schéma conceptuel et le schéma logique ce qui est très critiqué [Jovanovic and Bojicic 2012]. Les deux modèles sont normalisés, le modèle *Data vault* normalisé en 3FN le modèle *Anchor* est entièrement en 6FN directement au niveau conceptuel.

De plus, par leur nature, les modèles dimensionnels (*star schema* et *snowflake*) ne peuvent que contenir une sémantique spécifique très souvent encapsulée dans les processus d'alimentation ou dans les processus analytiques sous-jacents. Par contre, étant donné que les ontologies offrent un modèle indépendant des besoins et des sources de données, en plus d'être un excellent support d'intégration de données pour des systèmes disparates, leur utilisation est une approche très prometteuse pour uniformiser la sémantique et la structure d'un schéma relationnel permettant ainsi de garantir l'intégrité des données et la cohérence sémantique des applications.

1.4 Modèles du temps

La perception du temps dépend de plusieurs facteurs (astronomiques, culturels, religieux, etc.). Aussi, la perception du temps est variée : continue, linéaire ou cyclique, bornée ou non, finie ou infinie. La définition du temps et les représentations qui en découlent sont donc souvent très complexes. Plusieurs théories et modèles temporels ont été élaborés depuis 1960 et dont le but était de définir les concepts primitifs requis pour la représentation et le raisonnement. Une étape importante a été franchie lorsqu'Allen [Allen 1983] a défini une algèbre des intervalles avec six relations asymétriques (*before/after, meets/met_by, overlaps/overlapped_by, starts/started_by, during/contains, finishes/finished by*) et une relation symétrique (*equal*) pour

[11] http://danlinstedt.com
[12] http://www.anchormodeling.com

représenter les changements et les interactions entre des évènements et des actions [Allen and Ferguson 1994].

1.4.1 Revues

Une revue de la littérature et une étude comparative intéressante des modèles du temps sont présentées dans [Ermolayev et al. 2014] ; la suite de la présente sous-section en est un résumé.

McDermott [McDermott 1982] a étudié le concept de ligne de temps (*timeline*) comme structure de base pour la représentation informatique du temps. Il définit « ligne de temps » selon plusieurs propriétés :

Discret, dense ou continu

Un modèle temporel discret [Prior 1967; Halpern and Shoham 1991; Sandewall 1995; Ermolayev et al. 2008] est isomorphe aux nombres naturels et est formé de points temporels non décomposables ayant une durée minimale positive (*chronon*). Par contre, les modèles de temps denses [Koubarakis 1992; Sandewall 1995] et continus [Lamport 1978; McDermott 1982; Allen 1983; Pinto 1994] sont isomorphes aux nombres rationnels et réels, respectivement.

Bien que les modèles denses et continus soient plus expressifs, un modèle du temps discret est privilégié parce que les systèmes de stockage et d'analyse de données sont définis selon des modèles discrets pour avoir une représentation concise et une complexité computationnelle qui est utilisable en pratique [Meisen 2016].

Linéaire, ramifié ou circulaire (*linear, branching, circular****)***

Dans une logique temporelle linéaire (ordre total) [Allen 1983; Pinto 1994; Sandewall 1995; Ermolayev et al. 2008], le temps se déplace du passé vers le futur et à chaque point il existe un seul point successeur (un point dans le futur). Dans une logique temporelle ramifiée (ordre partiel) [Bergson 1910; Prior 1967], chaque point peut avoir plusieurs successeurs dans le futur formant ainsi une structure d'arbre infinie permettant, notamment, de définir des lignes de temps parallèles [Halpern and Shoham 1991]. Dans un modèle circulaire, il n'y a ni passé ni futur ; un temps circulaire « tourne » à l'infini, voir [Date et al. 2014]

Borné ou non borné (*bounded, unbounded****)***

Un modèle temporel borné [Prior 1967; Pinto 1994; Sandewall 1995] comporte une borne inférieure (alpha ou origine), supérieure (oméga) ou les deux. Lorsque le modèle est discret et borné aux deux extrémités, l'ensemble de points formant la ligne de temps est fini. Par contre, dans les modèles partiellement bornés (une seule borne) et les modèles non bornés [McDermott

1982; Allen 1983; Halpern and Shoham 1991; Ermolayev et al. 2008] l'ensemble des points est infini.

1.4.2 Synthèse

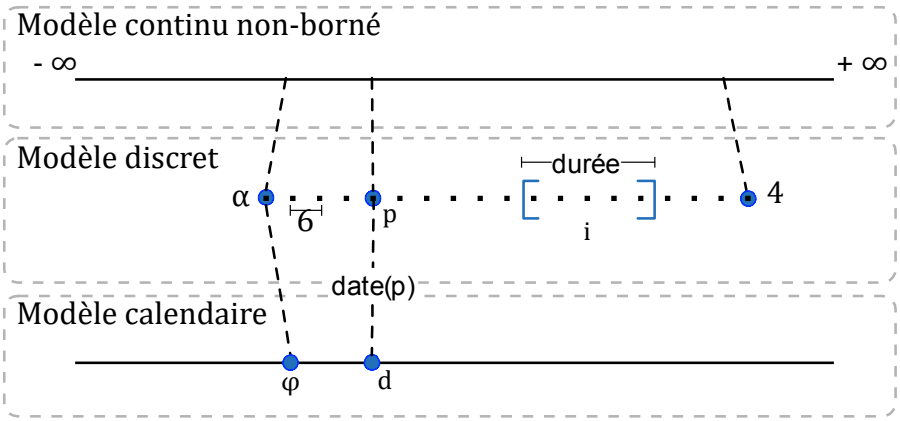
Bien que la perception du temps soit le plus souvent continue, sa représentation dans une base de données ne peut être que discrète et finie. Tous les modèles représentent une portion de la réalité et possèdent des avantages et des inconvénients. Cependant, lorsque les données du problème sont appelées à être stockées et manipulées dans un système informatique, il en découle certaines contraintes incontournables. Un modèle temporel linéaire est privilégié à cause des systèmes de datation [ISO 2004] qui fournissent un algorithme à temps constant pour comparer des dates d'une façon déterministe. Plus spécifiquement, l'ensemble des valeurs ne peut qu'être discret et fini. Les systèmes informatiques ont une limite supérieure qui est contrainte par une implémentation matérielle. Il est donc impossible d'utiliser des modèles du temps dense ou continu [Jensen and Snodgrass 1996]. De ce fait, nous nous intéressons donc uniquement aux modèles discrets, linéaires et bornés aux deux extrémités afin d'avoir une représentation concise, et qui soit calculable de façon efficiente par les systèmes informatiques et en particulier par les SGBD.

Les concepts temporels de base utilisés dans le cadre de cette étude sont définis ci-dessous et illustrés dans la Figure 3. Plusieurs définitions sont librement adaptées de [Manthey 2014] :

- **Axe de temps du modèle discret** : succession finie de points équidistants strictement ordonnés. La borne inférieure de l'axe est nommée alpha (α) et la borne supérieure oméga (ω). Un point temporel est appelé **chronon** ou **instant**. Usuellement l'axe est projeté sur un sous-ensemble fini des entiers. Un axe de temps correspond donc à la définition d'un type en regard de la théorie relationnelle ; un point temporel est une valeur de ce type (une valeur entière si la projection usuelle est utilisée).
- **Granularité** : distance entre deux chronons successifs, notée gamma (γ). Elle est exprimée par une quantité scalaire positive (entier, réel, etc.) et une unité de mesure (typiquement l'année, le jour, la minute ou la seconde).
- **Période** (intervalle temporel) : ensemble non vide de chronons contigus, qui est noté $[d:f]$, avec $\alpha \leq d \leq f \leq \omega$. Un singleton est également appelé **moment**. Il existe donc une correspondance biunivoque entre les instants (p) et les moments ($[p:p]$).
- **Durée** : soit i, une période : la durée de p est égale à, $card(i) \times \gamma$. Si l'axe de temps de référence est projeté sur les entiers, soit $i=[t_1:t_2]$ une période, la durée de p est égale à $(t_2-$

-t1+1) × γ. À remarquer que la durée d'un point n'est pas définie, mais la durée d'un moment l'est.

◊ **Date** : même si les calendriers sont hors de notre propos, il est généralement utile de permettre l'arrimage d'un type point avec un calendrier de référence.



α = premier point (valeur minimale d'un point)
4 = dernier point (valeur maximale d'un point)
6 = distance entre deux points consécutifs
φ = valeur d'un point sur un référentiel temporel (p. ex. calendrier)
p = type de point temporel INSTANT
i = type d'intervalle temporel PERIOD = INTERVAL[INSTANT]

durée(i) = card(i) * 6
date(α) = φ
$p_i < p_j \Rightarrow date(p_i) \leq date(p_j)$
$d_k < d_\ell \Rightarrow date\_inv(d_k) \leq date\_inv(d_\ell)$

Figure 3. Illustration des modèles de temps

Type point

Un type [13] point est tout type discret, ordonné et ordinal [14]. La valeur minimale est conventionnellement désignée par alpha (α) et la valeur maximale par oméga (ω). Un modèle ou un langage pourront, notamment pour des raisons pratiques, ajouter des contraintes à cette définition ; en particulier, un point pourrait être limité aux seuls types scalaires.

Un point est dit « temporel » (nommé **instant** ou chronon) si une durée uniforme est associée à la mesure de distance entre deux points consécutifs. La durée est conventionnellement exprimée en secondes (aux fins du présent document, la définition de la seconde est celle du [Bureau

[13] Un type est un ensemble fini de valeurs. En conséquence, un type point est également fini.

[14] Un type est ordonné s'il est muni d'une relation d'ordre total, une fonction d'ordre partielle n'est pas suffisante. Un type est ordinal s'il est possible d'établir une bijection avec un sous-ensemble des entiers; un type peut être ordonné sans être ordinal, par exemple les réels. Un type ordinal pourrait ne pas être ordonné si la fonction d'ordre total n'est pas précisée.

International des Poids et Mesures 2006]) ; le facteur d'échelle pourrait appartenir aux réels positifs, mais en pratique il sera contraint aux seuls rationnels positifs).

Type intervalle

Soit P un type point, un intervalle de P, noté[15] INTERVAL[P], étant l'ensemble des valeurs d'intervalles légitimes définies sur P où INTERVAL est le constructeur de type intervalle. Un intervalle peut être construit à l'aide de n'importe quel type point (un entier ou un type défini par énumération, par exemple). Un intervalle est dit « temporel » (nommé **période**) si P est un point temporel.

Une valeur d'intervalle définie sur P est un ensemble de points $\{x \in P \mid b \leq x \leq e\}$ où b (pour « *begin* ») est le point de début de l'intervalle, e (pour « end ») est le point de fin de l'intervalle avec $b \leq e$. Corolairement, une valeur d'intervalle ne peut être (un ensemble) vide. Un intervalle de cardinalité 1 est appelé singleton ou encore intervalle unitaire.

L'intervalle $\{x \in P \mid b \leq x \leq e\}$ est conventionnellement noté de cinq façons (Tableau 1) qui sont définies au tableau suivant. Soit f pour « *from* » (f=b-1), t pour « to » (t=e+1) et i pour « incrément » (i=e-b+1).

Tableau 1. Notation d'intervalle

Type de bornes	Notation	Définition
fermé-fermé	[b:e]	$b \leq x \leq e$
fermé-ouvert	[b:t)	$b \leq x < t$
ouvert-fermé	(f:e]	$f < x \leq e$
ouvert-ouvert	(f:t)	$f < x < t$
fermé-incrément	(b,i)	$b \leq x \leq b+i-1$

Sauf pour la dernière notation (fermé-incrément), la parenthèse représente une borne exclusive et le crochet une borne inclusive. Pour la dernière notation (fermé-incrément), remarquons l'utilisation de la virgule (plutôt que du deux-points) et que l'incrément i est égal à la cardinalité de l'intervalle et est donc supérieur à 0.

On remarque que, dans les cas discrets, ces notations sont équivalentes :

$$[b:e] = (b-1:e] = [b:e+1) = (b-1:e+1) = (b,e-b+1)$$

sauf pour les valeurs [α:e] qui ne peuvent être dénotées à l'aide des notations ouvert-fermé et ouvert-ouvert, ainsi que les valeurs [b:ω] qui quant à elles ne peuvent être dénotées à l'aide des notations fermé-ouvert et ouvert-ouvert. En conséquence, les notations fermé-fermé et fermé-incrément sont préférables lorsque vient le temps de formuler des règles générales.

15 Le langage TD utilise la notation INTERVAL_P sans crochet, avec un souligné.

Soit p et q deux intervalles notés [p.b:p.e] et [q.b:q.e]. Le Tableau 2 ci-dessous présente la notation de base des intervalles.

Tableau 2. Notation de base des intervalles

Opération	**Définition** [b:e]	**Définition** [b:t)	**Allen**	**TutorialD**
intervalle (b,e)	{x ∈ T | b ≤ x ≤ e}	{x ∈ T | b ≤ x < t}	[b:t)	[b:e]
début	p.b	p.b	p-	first
fin	p.e	p.t-1	p+	last
prédécesseur	p.b-1	p.b-1	pred	pre
successeur	p.e+1	p.e	succ	post
dénombrement	Nombre d'éléments	Nombre d'éléments	?	card
alpha	La plus petite valeur du type T	La plus petite valeur du type T	first	α
oméga	La plus grande valeur du type T	La plus grande valeur du type T	last	ω
appartenance	x ∈ p	x ∈ p	x ∈ p	∈
appartenance-1	p ∋ x	p ∋ x		∋

Opérateurs d'intervalles

La manipulation et la comparaison des intervalles s'effectuent grâce aux opérateurs définis par [Allen 1983] (voir illustration Figure 4).

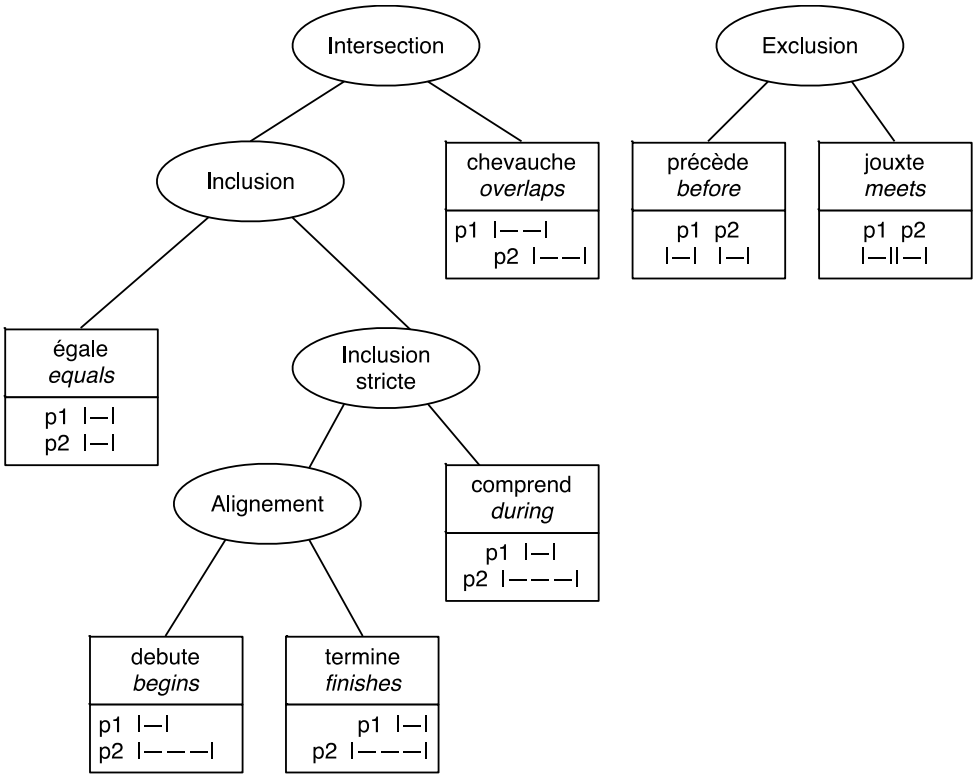


Figure 4. Illustration des opérateurs d'intervalles d'Allen

1.5 Modèles relationnels temporalisés

Un modèle de données temporalisé permet la prise en charge des concepts temporels requis pour la gestion de l'état passé (les données passées), de l'état courant (les données actuelles) et, dans une certaine mesure, de l'état prévu (les données attendues dans le futur). Le modèle relationnel a été étendu pour améliorer la gestion de données temporelles grâce à l'ajout du constructeur de type `INTERVAL`[16] et des opérateurs d'Allen [Allen 1983]. Conséquemment, des opérateurs relationnels (comme `PACK`, `UNPACK` et `USING`) ont également été ajoutés pour le traitement de relations contenant des attributs de type intervalle [Lorentzos and Johnson 1988; Date et al. 2014]. Pour plus de détails, il est possible de consulter les références [Snodgrass 2000; Manthey 2014; Date et al. 2014; Khnaisser 2016].

Il est important de mentionner qu'une récente revue de la littérature [Böhlen et al. 2018] présente les différents modèles et avancées technologiques dans les bases de données temporelles. D'ailleurs, un résumé et un complément de référence sont présentés ci-dessous. La Figure 5, illustre les différents modèles. Les boites grises représentent des méthodes indépendantes. Les boites avec les mêmes couleurs se basent sur les mêmes principes.

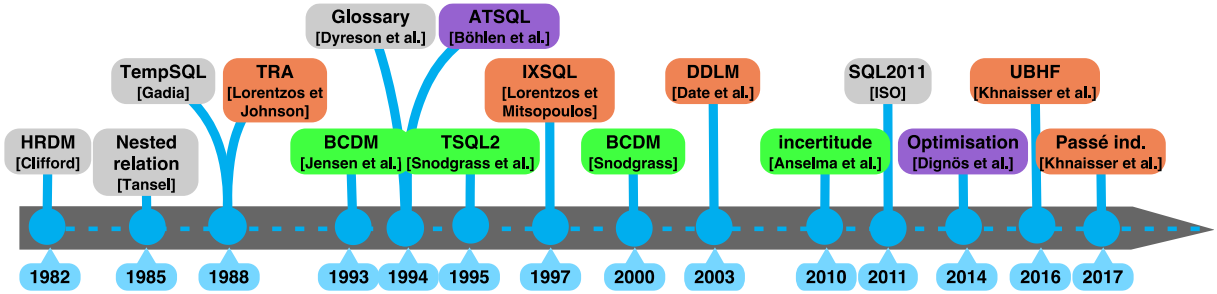


Figure 5. Bref résumé des modèles relationnels temporels

1.5.1 Premières approches

1.5.1.1 Théorie

Les premiers modèles [Tansel 1986; Clifford and Croker 1987; Gadia and Yeung 1988] ont proposé la modélisation du temps basée sur un modèle « relationnel imbriqué » (*nested relational mode*). Ce modèle s'écarte de la théorie relationnelle quant à la notion d'attribut atomique (la première forme normale). Autrement dit, chaque tuple est associé à un attribut temporel non atomique (*temporal element*) contenant l'union des ensembles d'intervalles qui

16 Par commodité, lorsque le contexte s'y prêtera, l'expression *de type intervalle* devra être interprétée comme « dont le type a été construit grâce au constructeur de type `INTERVAL` ».

représente l'histoire complète de l'évolution. Bien que dans ce modèle l'information temporelle était complète, le grand effort qui était requis pour étendre l'algèbre relationnelle et l'introduire dans les SGBDR a freiné son appropriation.

En 1994, un groupe de chercheurs se sont réunis pour définir un glossaire [Dyreson et al. 1994] contenant les principaux concepts temporels utilisés en base de données. Pour notre part, nous définissons certaines constructions ci-dessous (négligeant les composants qui ne sont pas utilisés dans le présent document). Les dimensions du temps sont le plus souvent intuitivement présentées ainsi :

- Temps du domaine (*user-defined time*)[17] : le temps du domaine fait référence à l'horloge associée à l'activité (ou au processus) ayant engendré le fait qui est représenté par la proposition (p. ex. date de naissance, date d'embauche, date de péremption).
- Temps de validité (*valid time*)[18] : le temps de validité d'un fait est le moment où le fait est vrai dans la réalité modélisée (p. ex. la période durant laquelle un patient était considéré comme souffrant d'une pneumonie, cette période pouvant être réévaluée par la suite).
- Temps de transaction (*transaction time*)[19] : le temps de transaction d'un fait est le moment auquel le fait est actuel dans la base de données et peut ne pas être récupéré (p. ex. moment de la dernière mise à jour du dossier patient enregistré dans la base de données).

Toutefois, les modèles n'ont pas tous adopté les mêmes concepts ni les mêmes définitions, ce qui rend difficiles la compréhension et la comparaison des différents modèles. De plus, il est à noter que différentes raisons peuvent justifier la présence de nombreux modèles, comme : l'incomplétude, le non-consensus, le manque de généralité, l'absence de mise en œuvre et d'outils de temporalisation, etc. Depuis les années 1980, plusieurs extensions temporelles au modèle relationnel ont été proposées pour permettre d'améliorer diverses fonctionnalités de traitements des données dans les bases de données relationnelles [Ozsoyoglu and Snodgrass 1995].

Un deuxième groupe de modèle a débuté avec les travaux de Lorentzos [Lorentzos and Johnson 1988]. Au départ, une extension élégante de l'algèbre relationnelle a été définie sur la base des intervalles et des opérations d'Allen [Allen 1983] pour permettre la manipulation des tuples associés à des intervalles. D'ailleurs, sur la base de ces mêmes travaux, une extension SQL a

17 *"User-defined time is an uninterpreted attribute domain of date and time.[…] unlike transaction time and valid time, it has no special query language support. It may be used for attributes such as "birth day" and "hiring date"."*

18 *The valid time of a fact is the time when the fact is true in the modeled reality. […]* [Dyreson et al. 1994]

19 *The transaction time of a database fact is the time when the fact is current in the database and may be retrieved. […]* [Dyreson et al. 1994]

également été proposée [Lorentzos and Mitsopoulos 1997]. Ensuite, cette proposition a été adaptée et raffinée par les travaux de [Date et al. 2003; Date et al. 2014] par la définition des opérations PACK, UNPACK et USING qui consistent à manipuler des relations dont les tuples comportent des attributs de type intervalle (notamment des clés). En plus des opérations sur les intervalles, Date et al. ont proposé une méthode d'historicisation unitemporelle [20] et bitemporelle[21] (nommée DDLM). Cette méthode s'articule selon le principe de base qui est de séparer en premier lieu l'état courant (*since*) de l'historique (*during*), puis de déléguer au SGBD la responsabilité de gérer la traçabilité des transactions (*logged time*) tout en garantissant quatre invariants : non-contradiction, non-redondance, non-circonlocution, compacité.

Un troisième groupe de modèles a débuté avec les travaux de [Jensen et al. 1993]. Un modèle conceptuel (nommé *Bitemporal conceptual data model* - BCDM) a été défini sur la base de 4 modèles [Ben-Zvi 1982; Snodgrass 1987; Gadia and Yeung 1988; McKenzie and Snodgrass 1991] pour permettre plusieurs représentations temporelles dans un même modèle. Puis, Snodgrass a proposé une extension de SQL (TSQL2) [Snodgrass 1995:2] avec des opérations selon trois catégories sémantiques : la sémantique non séquentielle (*non sequenced semantic*), sémantique courante (*current semantic*) et la sémantique séquentielle (*sequenced semantic*). Chacune des sémantiques « rattachées » à une contrainte ou à une requête permet un accès à un état spécifique de la base de données. En plus, Snodgrass a proposé une méthode d'historicisation unitemporelle et bitemporelle [Snodgrass 2000]. Le modèle continue à être raffiné avec plusieurs mécanismes comme la gestion de l'incertitude [Anselma et al. 2010; Anselma et al. 2015], la modélisation du temps volatile (*"now" problem*) [Anselma et al. 2013b; Anselma et al. 2016], les gestions des modifications des données [Anselma et al. 2013a; Yang et al. 2015].

Les plus récents travaux dans la temporalisation et l'historicisation sont proposés par [Dignös et al. 2014; Khnaisser 2016]. Les travaux de Dignös et al. [Dignös et al. 2013; Dignös et al. 2014; Dignös et al. 2019] suivent les principes d'historicisation de BCDM. En plus, ils utilisent implicitement les opérateurs proposés par DDLM pour créer les opérations « d'alignement temporel » (*temporal alignement*) qui permettent la préservation de la sémantique des données temporalisées et l'optimisation des opérations de jointure (*overlap interval partition join*). Les travaux de Khnaisser et al. proposent un cadre de référence unifié — *Unified Bitemporal Historicization Framework* (UBHF) — inspiré par DDLM et BCDM. Le cadre présente une

[20] Une méthode d'historicisation unitemporelle est une méthode qui définit une structure et des contraintes pour représenter le temps de validité .

[21] Une méthode d'historicisation bitemporelle est une méthode qui définit une structure et des contraintes pour représenter le temps de validité et le temps de transaction.

temporalisation basée sur les intervalles et une historicisation bitemporelle d'un modèle relationnel non temporalisé basé sur les principes de la normalisation. UBHF est brièvement présenté dans la section UBHF.

1.5.1.2 Pratique

Au niveau pratique, l'organisation internationale de normalisation [Kulkarni and Michels 2012] s'est en partie inspirée des travaux de [Snodgrass 1995] pour introduire des fonctionnalités temporelles dans ISO\SQL:2011. Depuis, certains SGBDR (Oracle 12c+, DB2 v10+, PostgreSQL 9.2+, Teradata v13+, MSSQL 2016, MariaDB 2019) ont commencé à introduire des fonctionnalités pour définir des attributs temporels (p. ex. *application time, system-time*, *type Range* de PostgreSQL) et des relations temporelles (p. ex. TABLE WITH SYSTEM VERSIONNING de MariaDB) pour gérer les transactions et requêtes temporelles (SEQUENCED, NONSEQUENCED de Teradata).

1.5.1.3 Perspectives

Pour conclure, diverses recherches sur l'extension du modèle relationnel et l'optimisation des opérations relationnelles sont toujours en cours et les SGBDR commencent (en retard) à mettre en œuvre certaines fonctionnalités. La question de l'historicisation n'est pas encore complètement réglée. Les méthodes actuellement proposées définissent les règles de transformation « par exemple » et doivent être en grande partie personnalisées et appliquées manuellement. Dans le prochain chapitre, nous proposerons une contribution à l'avancement de la temporalisation et de l'historicisation.

1.5.2 UBHF

Dans [Khnaisser 2016], nous avons proposé un formalisme pour décrire un modèle relationnel temporalisé qui est inspiré des travaux de [Snodgrass 2000; Date et al. 2014], de même que nous avons proposé une méthode automatisée d'historicisation bitemporelle d'un modèle relationnel non temporalisé. À noter que ce formalisme unifie la représentation syntaxique et sémantique des concepts temporels afin d'automatiser les étapes d'historicisation. Nous définissons certaines constructions ci-dessous (négligeant les composants qui ne sont pas utilisés dans le présent document).

1.5.2.1 Attributs temporels

Un attribut temporel peut être un instant ou une période. Le choix de la granularité de l'instant dépend de l'application (jour, minute, seconde, etc.). Dans les modèles relationnels non temporalisés, il est d'usage de catégoriser les attributs relativement à leur appartenance ou non

à une clé (attributs clé et attributs non-clé). Aux fins des modèles temporalisés, une catégorisation spécifique aux attributs temporels est introduite pour refléter la « nature » du temps utilisé (validité , transaction, domaine) :

Attribut temporel de domaine

Un attribut temporel de domaine (*user-defined time)* représente un instant ou une période ayant une sémantique spécifique dans le domaine d'application (p. ex. date de naissance, date de péremption, etc.). La valeur fait référence à l'horloge associée à l'activité (ou au processus) ayant engendré le fait représenté par la proposition (un tuple). Il peut en outre y avoir plusieurs référentiels de domaine (donc plusieurs attributs temporels de domaine pour une même proposition) dans un contexte où les processus ne partagent pas une même horloge. Ce référentiel est donc externe à la fois au modèle et au SGBD. Les valeurs qui s'y réfèrent sont donc « comme n'importe quelle autre valeur » et, conséquemment, le modèle temporel « n'a rien à en dire ». En pratique, on remarque que l'usage privilégie souvent un seul référentiel de domaine, bien qu'il y ait des exceptions notables comme la logistique du transport.

Attribut temporel de validité

Un attribut temporel de validité [@VT] (*valid time*, *stated time*, *application time*) représente la période durant laquelle une proposition est considérée comme vraie selon un agent. La valeur fait référence à l'horloge de l'agent qui a constaté le fait représenté par la proposition. Une période de validité commence à l'instant où l'agent considère le fait comme avéré et se termine lorsqu'il ne le considère plus comme avéré. Sa valeur est donc fournie par l'agent et modifiable par celui-ci. Il peut en outre y avoir plusieurs référentiels de validité (donc plusieurs attributs temporels de validité pour une même proposition) dans un contexte où il y plusieurs agents indépendants. En pratique, on remarque que l'usage privilégie le plus souvent un seul agent comme étant représentatif d'une organisation dans son ensemble.

Nous distinguons deux sous-catégories d'attribut temporel de validité :

- ◊ Un attribut temporel de validité indéterminé dans le futur [@Vbx] (nommé *since* dans DDLM) où seul le début de la période de validité de la proposition est connu.
- ◊ Un attribut temporel de validité déterminé [@Vbe] (nommé *during* dans DDLM) où le début et la fin de la période de validité de la proposition sont connus.

Attribut temporel de transaction

Un attribut temporel de transaction [@TT] (*transaction time*, *log time*, *system time*) représente la période durant laquelle la proposition est stockée dans la base de données. La valeur fait référence à l'horloge du SGBD qui enregistre une proposition dans la base de données. Sa

valeur est établie par le SGBD sur la base de l'instant de la confirmation (*commit time*) de la transaction (insertion ou retrait des tuples associés). L'utilisateur ne peut donc la modifier qu'indirectement, par l'entremise d'une transaction. Il n'y a qu'un seul référentiel de transaction.

Nous distinguons deux sous-catégories d'attribut temporel de transaction :

- ◊ Un attribut temporel de transaction indéterminé dans le futur [@Tbx] où seulement l'instant d'insertion est connu.
- ◊ Un attribut temporel de transaction déterminé [@Tbe] où l'instant d'insertion et l'instant de retrait sont connus.

Le Tableau 3 ci-dessous résume les attributs temporels du modèle d'historicisation.

Tableau 3. Catégorisation des attributs temporels

Notation	Définition	Type
@V	Attribut temporel de validité	PERIOD
@Vbx	Attribut temporel de validité indéterminé dans le futur	INSTANT
@Vbe	Attribut temporel de validité déterminé	PERIOD
@T	Attribut temporel de transaction	PERIOD
@Tbx	Attribut temporel de transaction indéterminé dans le futur	INSTANT
@Tbe	Attribut temporel de transaction déterminé	PERIOD

1.5.2.2 Relations temporelles

Une relation R est dénotée par R(K, A) où :

- ◊ $K = \{k_1, \ldots, k_n\}$ est l'ensemble d'attributs de la relation formant une clé ($|K| \geq 1$) ;
- ◊ $A = \{a_1, \ldots, a_m\}$ est l'ensemble d'attributs de la relation qui ne forment pas la clé ($|A| \geq 0$).

Une relation historicisée de R est dénotée par $R(K, B, C, D_V, D_T)$ avec $A = B \cup C$ où :

- ◊ $B = \{b_1, ..., b_{|B|}\}$ est un ensemble d'attributs non clés ($|B| \geq 0$), chacun étant associé à un temps de validité ;
- ◊ $C = \{c_1, ..., c_{|C|}\}$ est un ensemble d'attributs non clés ($|C| \geq 0$), chacun étant **non** associé à un temps de validité ;
- ◊ $D_V = \{@V, b_1@V, \ldots, b_m@V\}$ est l'ensemble d'attributs temporels de validité ($|D_V| = 1 + |B|$) où @V est associé à K et, pour tout $i \in [1..\, m]$, $b_i@V$ est associé à l'attribut $b_i \in B$.

◊ D_T = {@T, b_1@T, …, b_m@T} est l'ensemble d'attributs temporels de transaction ($|D_T| = 1 + |B|$) où @T est associé à K et, pour tout i ∈ [1.. m], b_i@T est associé à l'attribut $b_i \in B$.

Comme les attributs, les relations sont catégorisées selon les catégories d'attributs temporels qui forment l'entête. Avec les relations temporelles, les dimensions temporelles sont intrinsèques à la relation, dans le sens où la sémantique du prédicat représenté par la relation est toujours interprétée en fonction du temps.

Relation non temporelle

Une relation R est non temporelle [R@N] si son entête ne contient aucun attribut temporel $|K| \geq 1$, $|D_V| = 0$ et $|D_T| = 0$.

Relation unitemporelle de validité

Une relation R est unitemporelle de validité [R@V] si son entête contient au moins un attribut temporel de validité $|K| \geq 1$, $|D_V| \geq 1$ et $|D_T| = 0$.

Relation unitemporelle de transaction

Une relation R est unitemporelle de transaction [R@T] si son entête contient au moins un attribut temporel de transaction $|K| \geq 1$, $|D_V| = 0$ et $|D_T| \geq 1$.

Relation bitemporelle

Une relation R est bitemporelle [R@VT] si son entête contient au moins un attribut temporel de validité et au moins un attribut temporel de transaction $|K| \geq 1$, $|D_V| \geq 1$ et $|D_T| \geq 1$.

La Figure 6 ci-dessous présente toutes les catégories possibles d'une relation temporelle.

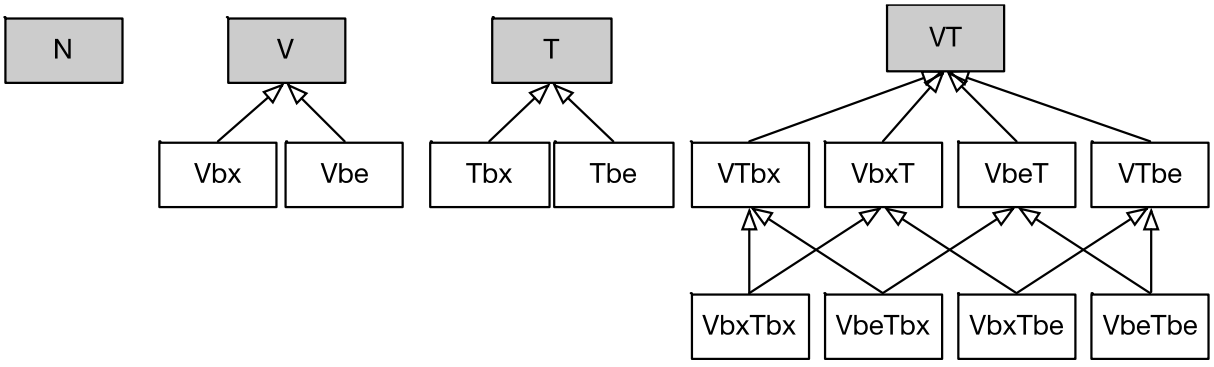


Figure 6. Catégorisation des relations dans un modèle relationnel temporel

Les catégories temporelles à la racine (N, V, T et VT) sont appelées catégories temporelles *primaires*. Les catégories temporelles aux feuilles (Vbx, Vbe, Tbx, Tbe, VbxTbx, VbeTbx, VbxTbe et VbeTbe) sont appelées catégories temporelles *finales*. Les catégories au milieu (VTbx, TbxT, VbeT, VTbe) sont appelées catégories temporelles *intermédiaires*.

1.5.2.3 Décomposition temporelle

La décomposition temporelle a pour but de représenter l'historicisation d'une relation R par un ensemble de relations, appelées *relparts*. En général, les ensembles privilégiés sont des ensembles minimaux à partir desquels toutes les relparts peuvent être calculées. Le choix des ensembles privilégiés varie d'une méthode à une autre. Par exemple, pour la bitemporalité, DDLM utilise {Vbx, Vbe, VbxTbe, VbeTbe} et BCDM {Tbx, Tbe, VTbx, VTbe}, voir [Khnaisser et al. 2017a].

Une *relpart* d'une relation R associée à la catégorie temporelle finale S est une relation dont l'entête comprend un sous-ensemble des attributs de R et les attributs temporels requis par S. Nous distinguons deux catégories de relpart :

- ◊ La relpart clé, notée R_K@S, dont l'entête comprend la clé R et les attributs temporels requis par S.
- ◊ La relpart a_i d'un attribut de R, notée R_a_i@S, dont l'entête comprend la clé de R, un attribut a_i et les attributs temporels requis par S.

Dans une décomposition temporelle de la relation R, un *regroupement clé* est l'ensemble des relparts associées à la clé et un *regroupement* a_i est l'ensemble des relparts associées à a_i.

Dans une décomposition temporelle de la relation R, une *partition* selon la catégorie temporelle S est l'ensemble des relparts associées à S.

1.6 Modèles ontologiques temporalisés

1.6.1 Revues

La temporalisation des ontologies (l'ajout de la dimension temporelle aux ontologies) et plus spécifiquement de OWL [Motik et al. 2012b] requiert un effort considérable [O'Connor and Das 2010]. Le formalisme de base n'est pas conçu pour traiter l'évolution des entités [Krieger 2010]. Plus particulièrement, à cause des prédicats binaires *property (object, subject)*. La temporalisation est souvent modélisée avec des prédicats d'une arité supérieure à 2 (soit une arité par dimension temporelle), p. ex. *validTime_property(object, subject, validtime), bitemporal_property(object, subject, validtime, transactiontime)*.

Il est important de mentionner que la revue de littérature de [Ermolayev et al. 2014] présente les différents modèles ontologiques du temps. Un résumé est présenté ci-dessous et comporte

également des références complémentaires qui ne sont pas incluses dans ladite revue de littérature.

La temporalisation des ontologies

Une ontologie du temps représente des entités temporelles de base (les instants et les intervalles) et un mécanisme de raisonnement. La majorité des ontologies du temps se repose sur le temps linéaire, supporte la représentation relative et absolue du temps et utilise les intervalles et les opérateurs d'Allen (sauf BFO de DOLCE). La liste suivante présente certaines des ontologies du temps qui sont les plus connues (Figure 7) :

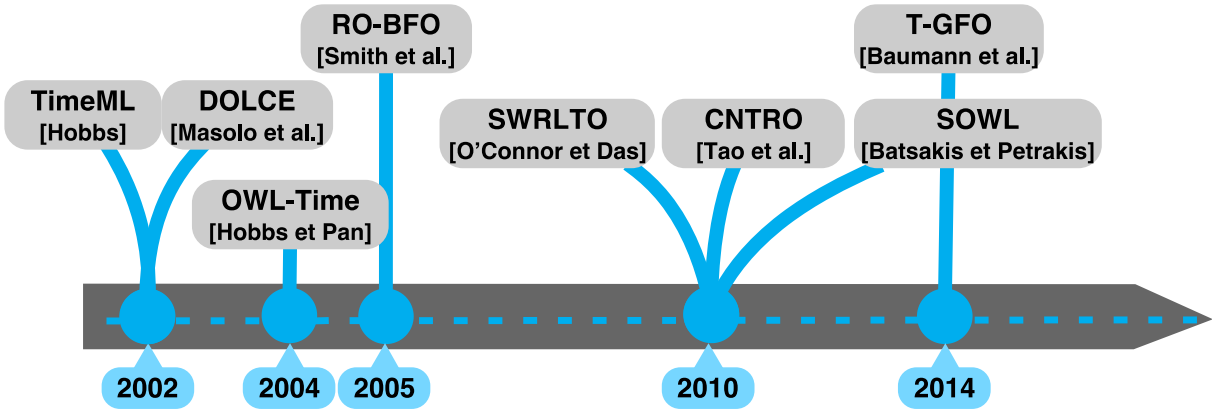


Figure 7. Bref résumé des ontologies du temps

- *DOLCE* [Masolo et al. 2002] représente les entités temporelles comme une qualité[22] (p. ex. les classes *Temporal Region*, *Temporal location*, *Time Intervals*) ; *DOLCE* ne donne aucune indication quant à la logique temporelle devant être utilisée.
- *TimeML* [Hobbs and Pustejovsky 2002], conçue principalement pour l'annotation et le raisonnement temporel de textes [analyse linguistique], permet de représenter les liens temporels entre des expressions temporelles, cela en incluant la négation.
- *OWL-Time Ontology* [Hobbs and Pan 2004], qui est recommandée par le W3C, définit les entités temporelles de base (p. ex. Les classes *Instant*, *Interval*, *Duration*, *TemporalUnit*) et des entités du calendrier (p. ex. les classes *DayOfWeek, MonthOfYear*), mais ne propose pas un mécanisme de raisonnement.

[22] *Qualities can be seen as the basic entities we can perceive or measure: shapes, colors, sizes, sounds, smells, as well as weights, lengths, electrical charges... 'Quality' is often used as a synonymous of 'property', but this is not the case in this upper ontology: qualities are particulars, properties are universals. Qualities inhere to entities: every entity (including qualities themselves) comes with certain qualities, which exist as long as the entity exists.* voir https://www.w3.org/2001/sw/BestPractices/WNET/DLP3941_daml.html#quality%20#5

◊ *RO Ontology* [Smith et al. 2005], basée sur *Basic Formal Ontology* [BFO] [Smith 2015], définit les entités temporelles de base (p. ex. Les classes *Temporal location*, *Temporal instant*, *Temporal part*)sans qu'il y ait toutefois de précision sur la logique temporelle utilisée. Cependant, faute de consensus[23], ces entités temporelles de *RO Ontology* ont été retirées de BFO et de ce fait il y a seulement les classes de haut niveau qui demeurent dans BFO 2.0 (et de haut niveau (les classes *Temporal Region*, *Zero-dimensional temporal region* et *One-dimensional temporal region)*.

◊ *SQWRL* [O'Connor and Das 2010] est basée sur le modèle unitemporel de validité (p. ex. les classes *ValidInstant*, *ValidPeriod*) et le langage de description de règles *SWLR* ; la simplicité de *SWLR* permet une intégration aisée avec l'ontologie à temporaliser.

◊ T-GFO [Baumann et al. 2014], basée sur *General Formal Ontology* GFO [Herre 2010], définit les intervalles [*Chronoids et Time Boundary*] bornés par des points temporels et comparables par les opérateurs d'Allen.

◊ *SOWL* [Batsakis and Petrakis 2011] permet une représentation qualitative et une représentation quantitative ; *SOWL* intègre également les entités temporelles de base et les opérateurs d'Allen en utilisant des règles *SWRL*[24] (*Semantic Web Rule Language*). C'est l'ontologie la plus complète en termes d'entités temporelles et de capacité de raisonnement [Anagnostopoulos et al. 2013].

◊ *Clinical Narrative Temporal Relation Ontology* [CNTRO[25]] [Tao et al. 2010] a été développée pour répondre aux besoins du domaine clinique et mieux tirer parti des données textuelles qui décrivent les activités liées aux épisodes de soins.

Nous tenons à souligner dès à présent que l'examen de l'expressivité et de la complétude des ontologies du temps sort du cadre de notre thèse. Toutefois, plus de détails sur ce sujet sont présentés dans une revue critique [Galton 2018] et dans une revue comparative [Ermolayev et al. 2014]. Une des principales conclusions qui ressort de ces revues est qu'il y a une absence de consensus quant à la représentation du temps dans les ontologies.

L'historicisation des ontologies

Comme c'est le cas pour le modèle relationnel, il y a plusieurs méthodes d'historicisation qui ont été proposées pour les ontologies. Une ontologie historicisée doit pouvoir représenter les

23 Proposition Time-BFO, voir https://github.com/BFO-ontology/BFO/blob/master/docs/OWL-TIME/bfo-owl-time.pdf

24 Voir https://www.w3.org/Submission/SWRL/

25 Renommée Time-Event Ontology (TEO) : https://sbmi.uth.edu/ontology/project/time-event-ontology.htm

liens temporels entre les entités non temporelles et les entités temporelles (préférablement exprimées selon une ontologie du temps).

Nous distinguons deux groupes de méthodes et les présentons à la Figure 8 : (1) les méthodes qui nécessitent la modification et l'extension de OWL ou RDF (rectangles en gris), et (2) les méthodes qui utilisent OWL directement ou avec le langage de règles SWRL (rectangles en orange).

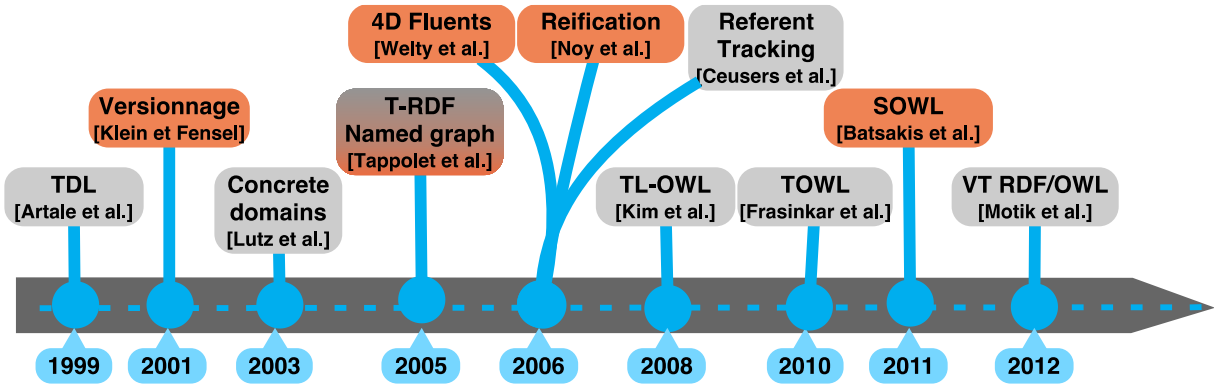


Figure 8. Bref résumé de l'historicisation des ontologies

Les méthodes dont les modèles nécessitent la modification et l'extension de OWL ou RDF sont très intéressantes au niveau théorique en raison de leur apport sémantique. Cependant, l'absence d'outils pour les mettre en pratique ralentit considérablement leur évaluation et leur adoption. La liste suivante présente les méthodes les plus connues :

- Les logiques descriptives temporelles [Artale and Franconi 1999; Lutz et al. 2008] constituent une extension de la logique descriptive [Baader 2010]. Un grand nombre de ces logiques est basé sur le modèle temporel linéaire et discret [Halpern and Shoham 1991; Allen and Ferguson 1994]. Bien que ces logiques offrent une riche expressivité, les capacités de raisonnement demeurent un problème majeur [Artale et al. 2015].
- Le modèle de domaine concret (*concrete domain*) [Lutz 2001; Lutz et al. 2008] concrétise l'historicisation des entités en introduisant des types de données avec leurs opérateurs. Ce modèle a servi comme base pour plusieurs modèles et méthodes, mais la complexité de raisonnement est au moins exponentielle en temps et en espace.
- Le modèle *referent tracking* [Ceusters and Smith 2006] consiste principalement à générer des identifiants artificiels uniques pour référer à un individu de la réalité (p. ex. un patient, une facture, une tumeur) et des prédicats ternaires où la 3e variable est le temps d'assertion de la proposition représentée.

◊ Le modèle TL-OWL [Kim et al. 2008] propose une extension de OWL-DL selon la logique descriptive temporelle basée sur les intervalles [Halpern and Shoham 1991], extension à laquelle s'ajoute une mise en correspondance avec RDF et un raisonneur. Cependant, aucune référence ne montre une mise en œuvre de ce modèle.

◊ Le modèle TOWL construit 3 couches en dessus de OWL, la 1re couche introduit le modèle de domaine concret, la 2e couche introduit les opérateurs d'Allen et la 3e couche permet de gérer l'évolution des individus en utilisant le modèle *4D fluents*. Le principal avantage de TOWL est de permettre de définir le modèle temporel en OWL et de gérer les instances directement depuis le modèle. Cependant, ce modèle n'est pas compatible avec les outils d'édition ni de raisonnement de OWL.

Les modèles et les méthodes qui utilisent directement OWL traitent principalement des problématiques de raisonnement ou d'expression de requête. Cependant, la complexité structurelle et son application qui est seulement faite au niveau RDF (sauf la proposition de SOWL) ralentissent considérablement leurs adoptions. La liste suivante présente les principaux modèles et méthodes :

◊ Le *versionnage* (*versionning*) [Klein and Fensel 2001] est une méthode qui gère l'évolution de l'ontologie en créant des versions pour toute modification. Cependant, le temps n'y est pas représenté explicitement et les requêtes utilisant des intervalles sont pratiquement impossibles à formuler. À noter qu'un système de gestion de version adapté et un mécanisme de requête sont nécessaires pour faire une utilisation optimale de cette solution.

◊ Le modèle RDF temporel [Gutierrez et al. 2005] est une extension de RDF par annotation qui représente le temps de validité du triplet. Une implémentation de ce modèle [Tappolet and Bernstein 2009] utilise les graphes nommés (*named graphs*) [Carroll et al. 2005] où chaque intervalle de temps est représenté par exactement un graphe nommé, tous les triplets appartenant à ce graphe ont la même période de validité. Ce modèle opère seulement au niveau des données et ne permet pas d'utiliser l'expressivité d'OWL, ce qui réduit considérablement la possibilité de valider l'intégrité des données.

◊ La réification [Noy et al. 2006] est une méthode générique pour représenter une association *n*-aire. Elle consiste à créer une entité pour l'association à temporaliser. Cependant, cette méthode engendre beaucoup de redondance et un grand nombre d'entités intermédiaires pour chaque entité à temporaliser (classe et individu), ce qui rend le développement de l'ontologie complexe et limite grandement les possibilités de raisonnement automatisé.

◊ Le modèle *4D fluents* [Welty and Fikes 2006] permet la représentation des entités temporalisées et leurs évolutions. Elle consiste à créer une entité temporelle pour chaque entité impliquée dans une association à temporaliser (p. ex. *time slice*). Plusieurs tentatives d'amélioration et de mise en œuvre de cette méthode ont été proposées [Tappolet and Bernstein 2009; Krieger 2010; Zamborlini and Guizzardi 2010]. Cependant, la mise en application de ce modèle cause une prolifération des entités ontologiques [Artale and Franconi 2000] rendant la maintenance et la formulation des requêtes plus complexe.

◊ Le modèle SOWL [Batsakis and Petrakis 2011] offre des entités temporelles avec des règles SWRL qui permettent de vérifier l'intégrité temporelle. De plus, cette méthode permet de définir des associations temporelles qualitatives (lorsque la valeur temporelle est absente). D'un point de vue pratique, ce modèle est très prometteur : des extensions [Anagnostopoulos et al. 2013; Batsakis et al. 2017] et des outils [Preventis et al. 2012] ont été proposés en pratique.

En conclusion, bien que ces méthodes aient des points positifs, il apparait au final qu'elles comportent de nombreux inconvénients :

◊ L'absence de mécanismes de gestion native des associations plus que binaires (p. ex. ternaire pour des modèles unitemporels) SQWRL [O'Connor and Das 2010].

◊ La limitation à une seule dimension temporelle (le temps de validité ou le temps de transaction).

◊ L'absence de mécanismes de gestion de l'évolution temporelle des individus de l'ontologie [Ermolayev et al. 2014].

◊ Le raisonnement avec des données temporelles appartient à la classe des problèmes NP-difficiles (*NP-Hard*) [Anagnostopoulos et al. 2013; Batsakis et al. 2017].

Finalement, certains inconvénients peuvent être partiellement éliminés en utilisant la représentation des graphes nommés, mais leur mise en application à grande échelle n'a pas encore été évaluée. Actuellement, demeurent d'importants défis de maintenance et d'évolution de la définition de l'ontologie et son instanciation lors d'une mise en application à grande échelle.

1.6.2 Synthèse

Vu l'hétérogénéité et le grand nombre de sources de données, une ontologie formelle est requise pour garantir une sémantique explicite du modèle et des données. L'utilisation des ontologies est un moyen de construction d'un modèle de données qui est mieux adapté pour définir une

sémantique formelle et explicite dans ce genre de situation. De plus, l'interprétation des données cliniques est fortement liée au temps et au contexte. Ainsi, l'utilisation d'un modèle de temps et d'une méthode d'historicisation sur plusieurs axes de temps est nécessaire pour garantir une traçabilité complète de l'évolution des données. Le modèle relationnel est un modèle adapté pour ce genre d'historicisation sur plusieurs axes de temps. De plus, une nouvelle méthode de construction de modèle de données qui permet d'arrimer le modèle relationnel et les ontologies offre le potentiel de répondre à aux exigences présentées à la section Exigences de l'État de l'art.

1.7 Ontologie vers un modèle de données relationnel

Une ontologie, comme une base de données relationnelle, représente des prédicats et des propositions (ou faits) en utilisant différents concepts de modélisation. Une ontologie est définie à l'aide de classes, d'individus, d'axiomes, de propriétés (propriétés d'objet et propriétés de données), de types de données et d'annotations. Pour plus de détails à ce sujet, voir la section µOnto ou les références [Baader 2010; Motik et al. 2012a]. Une base de données relationnelle comprend un schéma relationnel et l'ensemble de variables relationnelles qui en découlent. Pour plus de détails à ce sujet, voir la section µRel ou les références [Codd 1990; Darwen and Date 1995]. Les deux modèles qui sous-tendent µOnto et µRel, le modèle ontologique et le modèle relationnel, partagent des bases communes : la théorie des ensembles et la logique du premier ordre. Ainsi, une conversion de l'un à l'autre est possible, du moins théoriquement. Le principal défi est de maintenir la richesse sémantique des composants ontologiques dans un modèle relationnel.

Dans cet esprit, une revue de la littérature a été réalisée pour explorer les méthodes existantes de génération d'un schéma relationnel à partir d'une ontologie [Khnaisser et al. 2018]. Ces méthodes diffèrent les unes des autres de plusieurs manières : les étapes de conversion, les composants de l'ontologie étant pris en compte lors de la conversion (couverture de conversion), la complétude du schéma relationnel généré, la disponibilité des outils qui implémentent la méthode, etc. Néanmoins, dans le contexte de l'utilisation des ontologies pour générer un schéma relationnel, il reste de nombreux problèmes concernant la conservation de l'expressivité des ontologies, notamment la préservation des cardinalités de propriétés et de l'expressivité des axiomes, le traitement des données absentes, la gestion de la réversibilité et

l'évolution du schéma, de même que la maintenance de la documentation du schéma (à noter que ce qui suit est un extrait traduit et synthétisé de l'article [Khnaisser et al. 2018]).

Préserver les participations de propriétés

Un axiome est une expression qui relie des classes ou des individus à l'aide de propriétés selon une restriction exprimée en termes de participation. Une participation représente le nombre d'individus pouvant participer à une propriété. Elle est représentée comme une participation avec un intervalle ayant une valeur minimale et une valeur maximale telle que [0.. 1], [1..1], [0..n], [1..n], [0..*], [1..*] et [n.. n] où n est un entier positif représentant la valeur exacte de la participation et le symbole « * » (étoile) représente une borne supérieure non définie. Dans un modèle relationnel, une cardinalité peut être représentée sous la forme de clés candidates (clés primaires, clés uniques), de contraintes générales. Un processus de conversion qui gère les contraintes de cardinalité augmente l'intégrité des données et la détection automatique des incohérences. Aucune des méthodes existantes ne permet de préserver explicitement les propriétés des axiomes.

Préserver l'expressivité des axiomes

Les axiomes peuvent être définis sous différentes formes : simple ou complexe. Un axiome simple est défini par des entités atomiques comprenant une propriété, une ou deux classes et un type de données. Plusieurs axiomes simples peuvent être combinés par des opérateurs d'ensemble (intersection et union) et des opérateurs logiques (conjonction et disjonction), formant ainsi des axiomes complexes. Soit l'axiome complexe suivant $A\ op\ qt\ (B \cap C)$ où A, B et C sont des classes, op est une propriété d'objet et qt est une participation ; il est possible de l'exprimer en 3 axiomes simples : $A\ op\ qt\ Z$; $Z \sqsubseteq B \wedge Z \sqsubseteq C$ où Z est une nouvelle classe qui hérite de B et C (qui exprime $(B \cap C)$). Cela peut poser problème lors de l'utilisation des ontologies biomédicales [OBO 2018], car les axiomes complexes sont fréquemment utilisés. Un processus de conversion est nécessaire pour convertir uniformément ces axiomes en préservant la sémantique complète. Aucune des méthodes existantes ne permet de générer des axiomes complexes.

Traiter les données absentes

Les axiomes d'une classe peuvent être facultatifs (comme les attributs annulables dans une base de données relationnelle). C'est-à-dire que la valeur peut être inconnue (dans certains cas) ou inapplicable (dans d'autres cas). Ceci peut être implémenté en utilisant des valeurs NULL, mais cette implémentation n'est pas sémantiquement complète. De plus, les valeurs NULL peuvent introduire une incohérence dans les résultats de la requête. Il est donc recommandé de les éviter

en utilisant d'autres techniques de modélisation. Aucune des méthodes existantes ne permet de traiter adéquatement les données absentes (sans attribut annulable).

Maintenir la compatibilité du type Ontologie-relationnel

Pour certains types d'ontologies (par exemple, `owl:rational, xsd:positiveInteger`), il n'existe pas d'arrimage direct vers un type relationnel. De plus, la compatibilité des types SQL et la manipulation exacte dépendent du SGBDR cible. Ceci doit également être pris en compte lors de la conversion de types. Peu de méthodes définissent une correspondance entre les types de l'ontologie (p. ex. OWL) et les types relationnels (p. ex. SQL), mais ne mentionnent pas si cette correspondance est configurable.

Permettre la réversibilité structurelle et des données

L'un des objectifs de l'utilisation des ontologies est la réutilisation du modèle dans plusieurs tâches. Une fois les données intégrées, un mécanisme efficace qui permet de lier des tuples à des entités peut s'avérer très utile pour la découverte de connaissances et l'accès aux données sous forme ontologique [Poggi et al. 2008]. Dans ce contexte, la réversibilité entre les composants d'une ontologie et ceux d'un modèle relationnel est un point essentiel à prendre en compte lors de la définition du processus de conversion. Il y a deux caractéristiques de réversibilité qui peuvent être définies : (1) la réversibilité de la structure, la possibilité de reconstruire ou d'identifier un composant ontologique à partir d'un composant relationnel et vice-versa ; et (2) la réversibilité des données, la possibilité de reconstruire des individus (c'est-à-dire en tant que triples RDF) à partir des tuples et vice-versa. À noter qu'il y a peu de méthodes qui permettent d'effectuer ces deux types de réversibilité.

Gérer l'évolution de l'ontologie et du schéma relationnel

Comme la connaissance est en constante évolution, cela implique des modifications aux ontologies qui auront vraisemblablement des répercussions sur le schéma relationnel associé. Le schéma doit donc pouvoir y faire face tout en maintenant des interprétations antérieures des connaissances et en préservant l'intégrité des données. L'impact lors de l'intégration d'un changement de connaissance impliquant une évolution du schéma peut être très important. Par conséquent, le processus de conversion doit être défini de manière à faciliter la modification et l'extensibilité de la structure.

Gérer la documentation de schéma

La documentation d'un schéma relationnel est souvent ignorée malgré la valeur ajoutée de nombreuses tâches telles que l'interrogation et l'intégration des données, le développement d'applications, etc. [Curino et al. 2009]. En effet, sans avoir une définition claire des données

ou de la structure de la base de données, plusieurs tâches ne peuvent pas être vérifiées ni même effectuées. Une partie de la sémantique de la classe est portée par les axiomes, mais des informations complémentaires peuvent être trouvées dans les annotations. Une annotation dans l'ontologie représente un texte dans un langage spécifique qui définit divers aspects d'une entité. L'utilisation d'annotations peut être bénéfique pour la génération du schéma et sa documentation. À noter que peu de méthodes permettent l'utilisation des annotations comme moyen de documentation.

Enfin, la conversion d'une ontologie en schéma relationnel a beaucoup de potentiel pour apporter performance et puissance aux opérations de données qui sont basées sur des ontologies. Cependant, il reste encore beaucoup à faire pour réduire l'écart entre les ontologies et les modèles de données relationnelles, mais les travaux réalisés dans le cadre de la présente thèse vont dans ce sens.

1.8 Proposition

Un SSA doit pouvoir s'appuyer sur un modèle de données répondant aux quatre critères suivants : (1) indépendant de la structure des sources, (2) historicisé (donc permettant de tracer l'évolution des données), (3) offrant un modèle d'interprétation sémantique unifié et (4) conforme aux exigences déjà présentées.

Le but principal de la présente thèse est de construire, conformément aux exigences, un modèle de données relationnel historicisé qui soit dérivé d'une ontologie. L'ontologie est utilisée pour décrire objectivement les connaissances d'un univers de discours spécifique, le modèle relationnel est utilisé pour stocker les données correspondantes dans une structure respectant les axiomes de l'ontologie. Un modèle basé sur une ontologie offre une axiomatisation explicite du modèle relationnel où des aspects de la cohérence des données peuvent être évalués à l'aide des axiomes. La Figure 9 représente l'architecture inspirée de l'architecture trischématique. Le schéma conceptuel d'un domaine est une ontologie construite en collaboration entre un ou plusieurs personnes (voire plusieurs établissements) ayant divers rôles : analyste-métier, ontologiste et architecte de base de données. Bien que la construction de l'ontologie soit hors portée de notre thèse, des suggestions (règles de pratique) sont proposées afin de faciliter la conversion optimale et augmenter la qualité générale du schéma. Ensuite, l'ontologie est utilisée par une méthode semi-automatique de construction d'un schéma logique selon un modèle ontologique relationnel temporalisé **OntoRelT** sous la supervision d'un architecte de bases de données. Pour évaluer la faisabilité de l'approche, la méthode est mise en œuvre par

un prototype **OntoRelα** qui consiste à convertir une ontologie OWL en un schéma relationnel SQL selon **OntoRelT**.

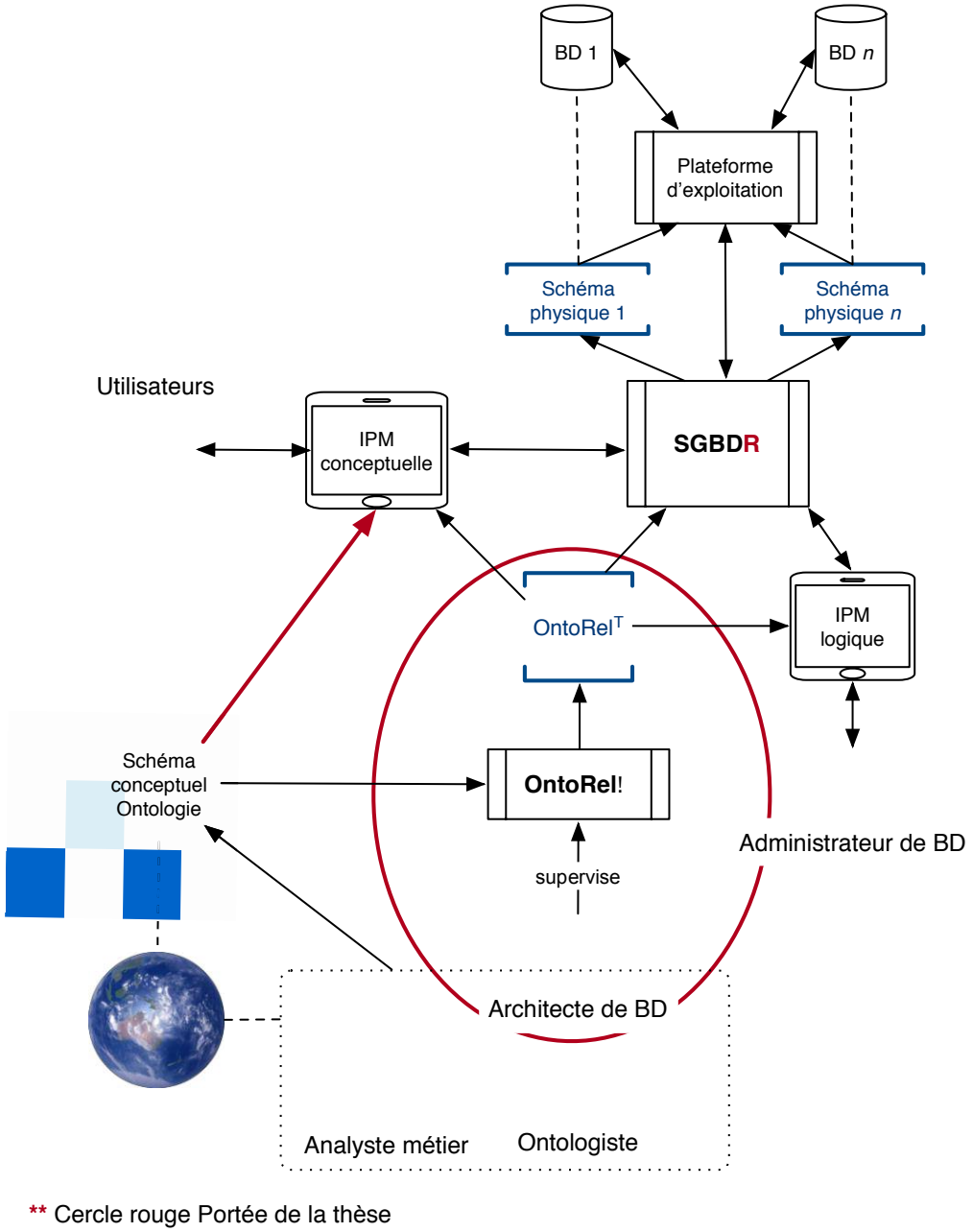


Figure 9. Processus proposé de modélisation de données

En résumé, le travail de cette thèse comporte :

- ◊ la définition d'un modèle ontologique relationnel temporalisé **OntoRelT** incluant :
  - o un modèle ontologique, **µOnto**, basé sur la logique de description, notamment *ALCQI(D)* [Chomicki and Saake 1998:8] ;
  - o un modèle relationnel, **µRel**, basé sur *The Third Manifesto* [Darwen and Date 1995; Date and Darwen 2010] ;

- un modèle relationnel temporalisé, **µRelT**, basé sur les travaux de [Date et al. 2014; Khnaisser 2016] ;

◊ la définition d'une méthode de génération de schéma relationnel selon le modèle OntoRelT incluant :

- le processus de conversion ontologie-relationnel composé (1) d'algorithmes d'analyse des axiomes ontologiques et (2) de règles de conversions des composants ontologiques en composants relationnels ;
- le processus d'historicisation composé des règles de conversion d'un schéma relationnel non temporel à un schéma relationnel temporel ;

◊ la spécification d'un prototype OntoRelα et sa mise en œuvre en utilisant des technologies ouvertes (*open source*) ;

◊ l'évaluation du modèle et du prototype dans le contexte d'une étude de cas des organisations hospitalières en France.

2
Méthode

Cette section présente la méthode utilisée pour générer un schéma relationnel historicisé à partir d'une ontologie : l'ontologie est d'abord analysée, simplifiée puis réduite ; ensuite, chaque composant ontologique est converti en un composant relationnel ; finalement, chaque composant relationnel est historicisé, puis traduit en SQL. Tout au long du processus des messages sont produits afin de consigner les anomalies et d'aider leur éventuel traitement. Préalablement à la présentation de la méthode elle-même, pour faciliter l'expression et la programmation des algorithmes, le modèle ontologique µOnto et le modèle relationnel temporel µRel sont définis.

2.1 µOnto

Une ontologie est formalisée à l'aide de µOnto, un modèle simplifié qui est inspiré de la théorie des ensembles et à certaines parties de la logique de description, notamment $ALCQI^{(D)}$ [Chomicki and Saake 1998:8]. Le but de µOnto est de définir des composants ontologiques et fournir une classification des axiomes adéquate et suffisante pour la conversion en un schéma relationnel tout en préservant l'expressivité ontologique. Nous définissons d'abord le modèle ontologique utilisé, puis un langage associé avant d'aborder la réduction de complexité et la réduction de redondance.

2.1.1 Modèle

Une ontologie est composée d'un ensemble de composants ontologiques : individus, classes, propriétés, types de données, annotations et axiomes.

Type de données

Un type de données [D] est un ensemble de valeurs restreint par une contrainte.

Individu

Un individu (ou instance) [e] est une représentation d'une entité de l'univers du discours.

Classe

Une classe (ou concept) [C] est un ensemble d'individus ayant des caractéristiques communes. Une ontologie contient minimalement deux classes. La classe Thing qui représente l'ensemble

de tous les individus et la classe Nothing qui représente l'ensemble vide. Deux opérateurs internes sur les classes sont définis, l'union (⊔), $C_0 \sqcup C_1 \Leftrightarrow C_0 \cup C_1$, et l'intersection (⊓), $C_0 \sqcap C_1 \Leftrightarrow C_0 \cap C_1$.

Propriété

Une propriété (ou rôle) [p] est une association (relation binaire ordonnée contrainte par une participation) qui associe deux individus ou un individu et une valeur. La définition d'une propriété comprend une restriction sur le premier composant (le domaine), une participation et une restriction sur le deuxième composant (la portée). Nous distinguons deux catégories de propriétés :

- Une propriété de classe [op], notée op : C_0 C_1, associe deux individus — le premier appartenant à C_0 (domaine) et le second à C_1 (portée).
- Une propriété de données [dp], notée dp : C D, associe un individu à une valeur — l'individu appartenant à la classe C (domaine) et la valeur au type D (portée).

Une propriété de classe peut en outre être définie comme l'inverse d'une autre [op_0 inv op_1], à savoir $(e_0, e_1) \in op_0 \Leftrightarrow (e_1, e_0) \in op_1$.

Axiome

Un axiome formalise une règle de l'ontologie. Nous distinguons deux catégories d'axiomes :

- **Axiomes d'héritage**
  - **Axiome d'héritage de classes** [$C_0 \sqsubseteq C_1$] définit un lien d'héritage entre deux classes C_0 et C_1, à savoir $C_0 \sqsubseteq C_1 \Leftrightarrow C_0 \subseteq C_1$.
  - **Axiome d'héritage de propriétés** [$p_0 \sqsubseteq p_1$] définit un lien d'héritage entre deux propriétés p_0 et p_1, à savoir soit $p_0 : C_{d0}\ C_{p0}$ et $p_1 : C_{d1}\ C_{p1}$ alors
    $p_0 \sqsubseteq p_1 \Leftrightarrow (C_{d0} \sqsubseteq C_{d1}) \wedge (C_{p0} \sqsubseteq C_{p1})$.
- **Axiomes d'association**
  - **Axiome d'association de classes** [C_d qt_d op qt_p C_p] associe les individus des classes C_d (le domaine) et C_p (la portée) en regard de la propriété de classe (op) soumise aux participations qt_d (participation de domaine) et qt_p (participation de portée).
  - **Axiome d'association de données** [C_d qt_d op qt_p D_p] associe les individus d'une classe C_d (le domaine) et les valeurs d'un type D_p (la portée) en regard de la propriété de classe (op) soumise aux participations (qt) qt_d (participation de domaine) et qt_p (participation de portée).

Participation

Une (contrainte de) participation [qt] est une contrainte limitant le nombre minimal et le nombre maximal d'associations auxquelles un individu (resp. une valeur) du domaine (resp. portée) d'une propriété doit participer en regard des individus (resp. des valeurs) de la portée (resp. domaine) de la propriété. Une participation est représentée par [min..max] où $min \in \mathbb{N}_{\geq 0}$, $max \in \mathbb{N}_{>0} \cup$ '*' où '*' représente une limite supérieure non bornée à priori[26]. De plus, si $max \neq$ '*' alors $min \leq max$. Nous présentons, ci-dessous, trois opérateurs binaires sur les participations, l'égalité (=), l'inclusion (⊆) et l'intersection (∩) ainsi qu'un opérateur binaire entre un entier et une participation, l'appartenance (∈).

Soit $a, c \in \mathbb{N}_{\geq 0}$; $b, d \in \mathbb{N}_{>0}$; $e \in \mathbb{N}_{>0} \cup$ '*'

(1) $[a..b] = [c..d]$ ssi $a = c \wedge b = d$
(2) $[a..*] = [c..*]$ ssi $a = c$

(3) $[a..b] \subseteq [c..d]$ ssi $c \leq a \wedge b \leq d$
(4) $[a..e] \subseteq [c..*]$ ssi $c \leq a$

(6) $[a..b] \cap [c..d] = [max(a,c)..min(b,d)]$ ssi $max(a,c) \leq min(b,d)$
(7) $[a..b] \cap [c..*] = [c..*] \cap [a..b] = [max(a,c)..b]$ ssi $c \leq b$
(8) $[a..*] \cap [c..*] = [max(a,c)..*]$

(9) $a \in [c..d]$ ssi $a \geq c \wedge (d='*' \vee a \leq d)$

Annotation

Une annotation décrit un composant ontologique par un texte. Une annotation est représentée par un quadruplet <E, N, L, T> où E est une référence à un composant ontologique, N la nature de l'annotation (étiquette ou description), L le code ISO 639 de la langue dans laquelle le texte est rédigé et T est le texte de l'annotation.

Considérations pratiques

Les types de données peuvent être assimilés à des classes prédéfinies, notamment dans le contexte des réductions présentées ultérieurement.

Plusieurs modèles ontologiques (dont OWL) ne permettent pas d'indiquer les participations de domaine et de portée dans un même axiome. Lors de la transposition d'une ontologie exprimée dans l'un de ces modèles vers µOnto, il faut donc réunir les paires d'axiomes correspondants de la forme « C_d op qt_d C_p ; C_p (inv op) qt_p C_d » en un seul axiome µOnto « C_d qt_d op qt_p C_p ».

[26] [0..0] a été exclu du domaine des participations parce que cela voudrait dire que l'axiome exige qu'une telle relation ne soit pas présente. Pour le moment, nous avons exclu la négation dans µOnto.

Finalement, dans la présente version du modèle, la négation et la quantification universelle ne sont pas définies. Le but principal de ce modèle est de décrire des composants ontologiques qui permettent de décrire un processus de conversion ontologique-relationnelle uniforme et automatisable. Or, dans le cas général, la construction (matérialisation) d'une relation comme complément d'une autre n'est pas réalisable en pratique par un SGBD (essentiellement pour des raisons de ressources en temps et en espace).

2.1.2 Langage

La syntaxe abstraite du langage µOnto est présentée ci-après. Un langage concret complet est défini en annexe 1.

```
ontology    ::= instruction*
instruction ::= declaration | annotation | axiom

declaration ::=
   'CL' ID                 // classe
 | 'DT' ID                 // type de données
 | 'OP' ID class0 class1   // propriété de classe
 | 'DP' ID class type      // propriété de données
 | 'INV' ID property       // ID est l'inverse de property

annotation  ::= annCategory element LN TEXT
element     ::= class | property
annCategory ::=
   'LAB'                   // étiquette
 | 'DES'                   // description

axiom       ::= domain operator range
domain      ::= expression
range       ::= expression
expression  ::=
   expression0 '⊓' expression1  //complexe
 | expression0 '⊔' expression1  //complexe
 | operator expression          //complexe
 | class                        //simple

operator    ::= '⊑' | qt0 property qt1
qt          ::= '[' MIN '..' MAX ']'
property    ::= ID
class       ::= ID
type        ::= ID

MIN   ::= /* entier_positif_ou_nul */
MAX   ::= /* entier_positif | '*' */
ID    ::= /* identifant (e.a. un IRI)*/
LN    ::= /* un code langue ISO 649 */
TEXT  ::= /* une suite de caractères */
```

2.1.3 Réduction de la complexité des axiomes

Les expressions de classes engendrent en pratique des classes anonymes. Pour permettre une conversion automatisable et rigoureuse des axiomes en composants relationnels, ces classes doivent être exprimées explicitement et donc déclarées au même titre que les autres.

La réduction de la complexité des axiomes convertit un ensemble d'axiomes complexes en un ensemble d'axiomes simples de manière à garantir la préservation de la sémantique. Plus spécifiquement, un axiome exprimé dans le formalisme de la source est analysé puis représenté selon la grammaire abstraite suivante.

La réduction de la complexité consiste à introduire de nouvelles classes et de nouvelles contraintes[27] de façon à ce que toute expression se réduise à un seul identifiant de classe (voir catégorie `class` dans la grammaire abstraite). Le processus de réduction de la complexité consiste donc en l'application récursive des règles, et qui sont décrites dans la rubrique ci-après, à chacun des axiomes.

Règles de réduction de complexité

Soit f une forme syntaxique (ici l'une des quatre formes de la catégorie `expression`), les règles peuvent être formellement décrites à l'aide des quatre fonctions suivantes :

- id(f) : l'identifiant associé à une forme f ;
- in(f) : l'ensemble des instructions devant être ajoutés à l'ontologie en raison du remplacement de la forme f par id(f) ;
- co(f) : l'ensemble des contraintes devant être ajoutées (dans un contexte de monde ouvert) afin de maintenir l'équivalence entre f et in(f) (et donc la réversibilité des règles).

Ces fonctions sont définies ci-après :

$$
\begin{aligned}
&id(C) = C\\
&in(C) = \{\}\\
&co(C) = \{\}\\
\\
&id(p\ \beta) = Z\\
&in(p\ \beta) = in(\beta) \cup \{\text{« CL Z »}, \text{«\ Z p id}(\beta)\ \text{»}\}\\
&co(p\ \beta) = co(\beta) \cup \{\text{« p } \beta \subseteq Z\ \text{»}\}\\
\\
&id(\beta \sqcap \gamma) = C\\
&in(\beta \sqcap \gamma) = in(\beta) \cup in(\gamma) \cup \{\text{« CL Z »}, \text{« Z} \sqsubseteq id(\beta)\ \text{»}, \text{« Z} \sqsubseteq id(\gamma)\ \text{»}\}\\
&co(\beta \sqcap \gamma) = co(\beta) \cup co(\gamma) \cup \{\text{« } (\beta \cap \gamma) \subseteq Z\ \text{»}\}
\end{aligned}
$$

[27] Une contrainte se présente sous la même forme qu'un axiome, mais n'est requise que pour maintenir la réversibilité de la règle car elle sera assurée structurellement par le schéma relationnel comme nous le verrons par la suite.

$id(\beta \sqcup \gamma) = C$
$in(\beta \sqcup \gamma) = in(\beta) \cup in(\gamma) \cup \{« CL\ Z », « id(\beta) \sqsubseteq Z », « id(\gamma) \sqsubseteq Z »\}$
$co(\beta \sqcup \gamma) = co(\beta) \cup co\ (\gamma) \cup \{« Z \subseteq (\beta \cup \gamma) »\}$

avec :

- ◊ C un identifiant de classe (`class`) déjà défini ;
- ◊ Z un nouvel identifiant de classe (`class`) généré à chaque utilisation de la règle et différent de tout autre au sein de l'ontologie ;
- ◊ p un opérateur (`operator`) — soit ⊑, soit un identifiant de propriété déjà défini ;
- ◊ β et γ des sous-expressions (`expression`).

Exemple 1

Voici un exemple inspiré de l'ontologie PDRO[28] (ontologie des prescriptions de médicaments). Soit les axiomes ontologiques suivants :

```
OP has_part  Thing Thing

DP has_value Thing String

CL Drug_product_specification
  Drug_product_specification
    [1..*] has_part [0..*]
      (Drug_product_name ⊔ Active_ingredient_name)
  Drug_product_specification
    [1..*] has_part [0..*]
      Drug_identification_number

CL Drug_product_name
  Drug_product_name [1..1] has_value [0..*] String

CL Active_ingredient_name
  Active_ingredient_name [1..1] has_value [0..*] String

CL Drug_identification_number
  Drug_identification_number [1..1] has_value [0..*] String
```

L'axiome complexe suivant :

```
Drug_product_specification
  [1..*] has_part [0..*]
    (Drug_product_name ⊔ Active_ingredient_name)
```

28 https://github.com/OpenLHS/PDRO. Par souci de clarté, les IRI (les identifiants) sont remplacés par les étiquettes anglais.

doit être réduit comme suit :

```
id(Drug_product_specification) = Drug_product_specification
in(Drug_product_specification) = {}
co(Drug_product_specification) = {}

id(Drug_product_name ⊔ Active_ingredient_name) =
   Drug_product_name_OR_Active_ingredient_name
in(Drug_product_name ⊔ Active_ingredient_name) =
 in(Drug_product_name) ∪
 in(Active_ingredient_name) ∪
 { CL Drug_product_name_OR_Active_ingredient_name,
   id(Drug_product_name) ⊑ Drug_product_name_OR_Active_ingredient_name,
   id(Active_ingredient_name) ⊑ Drug_product_name_OR_Active_ingredient_name
 }

id(Drug_product_name) = Drug_product_name
in(Drug_product_name) = {}
co(Drug_product_name) = {}

id(Active_ingredient_name) = Active_ingredient_name
in(Active_ingredient_name) = {}
co(Active_ingredient_name) = {}

co(Drug_product_name ⊔ Active_ingredient_name) =
 co(Drug_product_name) ∪
 co (Active_ingredient_name) ∪
 { Drug_product_name_OR_Active_ingredient_name ⊆
    (Drug_product_name ∪ Active_ingredient_name)
 }
```

L'ontologie résultante est la suivante :

```
OP has_part  Thing Thing

DP has_value Thing String

CL Drug_product_name_OR_Active_ingredient_name

CL Drug_product_specification
  Drug_product_specification
    [1..*] has_part [0..*]
      Drug_product_name_OR_Active_ingredient_name
  Drug_product_specification
    [1..*] has_part [0..*]
      Drug_identification_number

CL Drug_product_name
  Drug_product_name [1..1] has_value [0..*] String
  Drug_product_name ⊑ Drug_product_name_OR_Active_ingredient_name

CL Active_ingredient_name
  Active_ingredient_name [1..1] has_value [0..*] String
  Active_ingredient_name ⊑ Drug_product_name_OR_Active_ingredient_name

CL Drug_identification_number
  Drug_identification_number [1..1] has_value [0..*] String
```

2.1.4 Réduction de la redondance des axiomes

La réduction de complexité peut engendrer des axiomes redondants. De plus, lorsque plusieurs ontologies sont combinées, un chevauchement sémantique [Bruijn et al. 2006] peut apparaitre. Dans notre contexte, nous nous intéressons au chevauchement sémantique entre axiomes. Informellement, pour toute paire d'axiomes, si le domaine, la portée ou la propriété sont identiques ou liés par un axiome d'héritage, cela peut impliquer un chevauchement sémantique.

Lors de la conversion d'une ontologie en un schéma relationnel, le chevauchement sémantique peut entrainer une redondance des données, des incohérences ainsi que des problèmes potentiels au moment de l'exécution de requêtes et de modifications. À mesure que la complexité du schéma de données augmente, la cohérence et la précision des données sont plus difficiles à gérer. Pour faciliter le maintien de l'intégrité des données, les axiomes sont réduits en fonction de règles spécifiques qui minimisent la redondance des axiomes et peuvent également permettre de détecter des axiomes contradictoires.

Enfin, notons que, pour le moment, la résolution du chevauchement sémantique ne concerne que les axiomes d'association de classe. Les axiomes d'association de données nécessitent une hiérarchie des types de données qui ne sont définis ni dans µOnto ni dans OWL.

Définition d'une propriété étendue

Soit les axiomes

$(a_1)\ C_{d1}\ q_{d1}\ p_1\ q_{r1}\ C_{r1}$

$(a_2)\ C_{d2}\ q_{d2}\ p_2\ q_{r2}\ C_{r2}$

Nous nous intéresserons aux propriétés étendues qui en sont implicitement (anonymement) dérivées. Une propriété étendue P notée, $C_d\ [min_d..max_d]\ [min_r..max_r]\ C_r$, est définie comme suit :

$$P \subseteq \{(d,r) \mid d \in C_d \wedge r \in C_r\}$$
tel que
$$\forall\ d \in C_d : (\#\{y \mid (d,y) \in P\} \in [min_d..max_d]) \wedge$$
$$\forall\ r \in C_r : (\#\{x \mid (x,r) \in P\} \in [min_r..max_r]))$$

On définit les fonctions suivantes :

- $dom(P) = C_d$
- $domQ(P) = [min_d..max_d]$
- $ran(P) = C_r$
- $ranQ(P) = [min_r..max_r]$

On peut définir une nouvelle propriété étendue P relativement à une autre propriété étendue S ainsi :

$$P \triangleq C_d\ [min_d..max_d]\ S\ [min_r..max_r]\ C_r$$

P est alors équivalent à :

$$(C_d \cap dom(S))\ ([min_d..max_d] \cap domQ(S))\ P\ ([min_r..max_r] \cap ranQ(S))\ (C_r \cap ran(S))$$

L'opération d'héritage entre propriétés étendues, notée « P ⊑ S », est définie ainsi :

$$dom(P) \subseteq dom(S) \wedge domQ(P) \subseteq domQ(S) \wedge P \subseteq S \wedge ranQ(P) \subseteq ranQ(S) \wedge ran(P) \subseteq ran(S)$$

Règles de réduction de redondance

Relativement à l'opérateur d'héritage, soit e et f deux ensembles quelconques (en particulier des classes, des propriétés ou des propriétés étendues), nous nous intéressons aux cas suivants :

- ◊ e et f sont identiques ($e == f$)
- ◊ e dérive non trivialement de f ($e \sqsubset^+ f$)
- ◊ e et f ne dérivent pas l'un de l'autre ($e \not\sqsubset f$)

On définit en outre : $e \sqsubset^* f \triangleq (e \sqsubset^+ f) \vee (e = f)$

Dans ce contexte, nous nous intéresserons aux propriétés étendues

$$P_1' \triangleq C_{d1}\ q_{d1}\ p_1\ q_{r1}\ C_{r1}$$

$$P_2' \triangleq C_{d2}\ q_{d2}\ p_2\ q_{r2}\ C_{r2}$$

et aux seules possibilités de réduction de redondance de paires d'axiomes pour lesquelles

$$R1.\ (C_{d1} \sqsubset^* C_{d2}) \wedge (p_1 \sqsubset^* p_2) \wedge (C_{r1} \sqsubset^* C_{r2})$$
-- alors il serait souhaitable que les participations ne contredisent pas $P_1' \sqsubset^* P_2'$
-- c'est-à-dire $qt_{d1} \subseteq qt_{d2} \wedge qt_{r1} \subseteq qt_{r2}$

Ou

$$R2.\ (C_{d2} \sqsubset^* C_{d1}) \wedge (p_2 \sqsubset^* p_1) \wedge (C_{r2} \sqsubset^* C_{r1})$$
-- alors il serait souhaitable que les participations ne contredisent pas $P_2' \sqsubset^* P_1'$
-- c'est-à-dire $qt_{d2} \subseteq qt_{d1} \wedge q_{r2} \subseteq qt_{r1}$

Ou

$$R1 \wedge R2 \Leftrightarrow (C_{d2} == C_{d1}) \wedge (p_2 == p_1) \wedge (C_{r2} == C_{r1})$$
-- il serait alors souhaitable que les participations ne contredisent pas $P_2' == P_1'$
-- c'est-à-dire $qt_{d2} = qt_{d1} \wedge qt_{r2} = qt_{r1}$

Au besoin, un traitement (en fonction de la cible) est proposé dans le tableau suivant :

Tableau 4. Inventaire des traitements de réduction de redondance

Cible	Traitement
$P_1' == P_2'$	si (a_1) et (a_2) sont engendrés remplacer (a_1) et (a_2) par (a_3) C_{d1} $q_{d1} \cap q_{d2}$ P_1 $q_{r1} \cap q_{r2}$ C_{r1} sinon action selon paramétrage
$P_1' \sqsubseteq+ P_2'$	si (a_1) est engendré remplacer (a_1) par (a3) C_{d1} $q_{d1} \cap q_{d2}$ P_1 $q_{r1} \cap q_{r2}$ C_{r1} sinon action selon paramétrage
$P_2' \sqsubseteq+ P_1'$	si (a_2) est engendrée remplacer (a_2) par (a_4) C_{d2} $q_{d1} \cap q_{d2}$ P_2 $q_{r1} \cap q_{r2}$ C_{r2} sinon action selon paramétrage
$P_1' \not\sqsubseteq P_2'$	Ne rien faire Ne rien dire

Le paramétrage possède deux options :

- ◊ Automatique : une réduction est effectuée.
- ◊ Manuel : aucune réduction n'est effectuée et un avertissement est engendré pour signaler la redondance à l'ontologiste.

Le tableau recense les cas potentiels et la cible de réduction lorsque les domaines, les propriétés et les portées correspondantes sont avérés.

Tableau 5. Inventaire des cas de réduction de redondance

Domaine	Propriété	Portée	Résultat	Cas
$C_{d1} == C_{d2}$	$p_1 == p_2$	$C_{r1} == C_{r2}$	$P_1' == P_2'$	1.
		$C_{r1} \sqsubseteq+ C_{r2}$	$P_1' \sqsubseteq+ P_2'$	2.
		$C_{r2} \sqsubseteq+ C_{r1}$	$P_2' \sqsubseteq+ P_1'$	3.
		$C_{r1} \not\sqsubseteq C_{r2}$	$P_1' \not\sqsubseteq P_2'$	4.
	$p_1 \sqsubseteq+ p_2$	$C_{r1} == C_{r2}$	$P_1' \sqsubseteq+ P_2'$	5.
		$C_{r1} \sqsubseteq+ C_{r2}$	$P_1' \sqsubseteq+ P_2'$	6.
		$C_{r2} \sqsubseteq+ C_{r1}$	$P_1' \not\sqsubseteq P_2'$	7.
		$C_{r1} \not\sqsubseteq C_{r2}$	$P_1' \not\sqsubseteq P_2'$	8.
	$p_2 \sqsubseteq+ p_1$	$C_{r1} == C_{r2}$	$P_2' \sqsubseteq+ P_1'$	9.
		$C_{r1} \sqsubseteq+ C_{r2}$	$P_1' \not\sqsubseteq P_2'$	10.
		$C_{r2} \sqsubseteq+ C_{r1}$	$P_2' \sqsubseteq+ P_1'$	11.
		$C_{r1} \not\sqsubseteq C_{r2}$	$P_1' \not\sqsubseteq P_2'$	12.
	$p_1 \not\sqsubseteq p_2$	Toute condition	$P_1' \not\sqsubseteq P_2'$	13.
$C_{d1} \sqsubseteq+ C_{d2}$	$p_1 == p_2$	$C_{r1} == C_{r2}$	$P_1' \sqsubseteq+ P_2'$	14.
		$C_{r1} \sqsubseteq+ C_{r2}$	$P_1' \sqsubseteq+ P_2'$	15.
		$C_{r2} \sqsubseteq+ C_{r1}$	$P_1' \not\sqsubseteq P_2'$	16.
		$C_{r1} \not\sqsubseteq C_{r2}$	$P_1' \not\sqsubseteq P_2'$	17.
	$p_1 \sqsubseteq+ p_2$	$C_{r1} = C_{r2}$	$P_1' \sqsubseteq+ P_2'$	18.
		$C_{r1} \sqsubseteq+ C_{r2}$	$P_1' \sqsubseteq+ P_2'$	19.
		$C_{r2} \sqsubseteq+ C_{r1}$	$P_1' \not\sqsubseteq P_2'$	20.
		$C_{r1} \not\sqsubseteq C_{r2}$	$P_1' \not\sqsubseteq P_2'$	21.
	$p_2 \sqsubseteq+ p_1$	$C_{r1} == C_{r2}$	$P_1' \not\sqsubseteq P_2'$	22.
		$C_{r1} \sqsubseteq+ C_{r2}$	$P_1' \not\sqsubseteq P_2'$	23.
		$C_{r2} \sqsubseteq+ C_{r1}$	$P_1' \not\sqsubseteq P_2'$	24.
		$C_{r1} \not\sqsubseteq C_{r2}$	$P_1' \not\sqsubseteq P_2'$	25.
	$p_1 \not\sqsubseteq p_2$	Toute condition	$P_1' \not\sqsubseteq P_2'$	26.
$C_{d2} \sqsubseteq+ C_{d1}$	Analogue à $C_{d1} \sqsubseteq+ C_{d2}$		14 à 26	27.
$C_{d1} \not\sqsubseteq C_{d2}$	Toute condition	Toute condition	$P_1' \not\sqsubseteq P_2'$	28.

Exemple 2

Voici un exemple inspiré de l'ontologie PDRO[29] (ontologie des prescriptions de médicaments). Soit les axiomes ontologiques suivants :

```
OP has_part  Thing Thing

DP has_value Thing String

CL Healthcare_prescription
  Healthcare_prescription [1..*] has_part [0..*] Author_identification

CL Drug_prescription
  Drug_prescription ⊑ Healthcare_prescription
  Drug_prescription [1..1] has_part [0..*] Prescriber_identification
```

29 https://github.com/OpenLHS/PDRO. Par souci de clarté, les IRI (les identifiants) sont remplacés par les étiquettes anglaises.

```
CL Author_identification
  Author_identification [1..1] has_value [0..*]String

CL Prescriber_identification
  Prescriber_identification ⊑ Author_identification
  Prescriber_identification [1..1] has_value [0..*] String
```

Les axiomes suivant sont redondants :

```
(a1) Drug_prescription [1..1] has_part [0..*] Prescriber_identification
(a2) Healthcare_prescription [1..*] has_part [0..*] Author_identification
```

et doivent être réduit selon le cas 15 des règles de redondance sur la base de :

(x1) $C_{d1} \sqsubseteq^{+} C_{d2}$: Drug_prescription ⊑ Healthcare_prescription
(x2) $C_{r1} \sqsubseteq^{+} C_{r2}$: Prescriber_identification ⊑ Author_identification
(x3) $p_1 == p_2$: has_part == has_part

Puisque [1..1] ⊆ [1..*] et [0..*] ⊆ [0..*], les axiomes a1 et a2 peuvent être réduits à l'axiome suivant :

```
Drug_prescription [1..1] has_part [0..*] Prescriber_identification
```

L'ontologie résultante est la suivante :

```
OP has_part Thing Thing

CL Healthcare_prescription

CL Drug_prescription
  Drug_prescription ⊑ Healthcare_prescription
  Drug_prescription [1..1] has_part Prescriber_identification
```

2.2 µRel

L'objectif principal de µRel est de construire des composants relationnels indépendamment des dialectes SQL. Le modèle relationnel utilisé par µRel est une variante simplifiée du modèle proposé dans *The Third Manifesto* [Darwen and Date 1995; Date and Darwen 2010] et son langage est une légère variante de *TutorialD* [Date 2015], un langage relationnel proposé par Date et Darwen, conforme au *Third Manifesto*. Les variantes seront identifiées dans la présentation synthétique ci-après.

2.2.1 Modèle

Une base de données relationnelle comporte un ensemble de variables de relations (relvars) définies par un ensemble de schémas. En voici les éléments constitutifs utilisés dans le cadre de la présente recherche.

Type de données

Un type de données est un ensemble fini de valeurs. Les types prédéfinis comprennent obligatoirement les booléens (`BOOLEAN`) et, usuellement, les nombres (`NUMBER`), les chaines de caractères (`CHARACTER`) et les estampilles temporelles (`INSTANT` ou `TIMEPOINT`). À noter que le mécanisme d'héritage proposé par Date [Date 2016] n'est pas pris en compte ici.

Attribut

Une définition d'attribut est une paire ordonnée nom:type. Une valeur d'attribut est une paire ordonnée nom:valeur. Une valeur d'attribut est conforme à une définition d'attribut si le nom est identique et la valeur appartient à l'ensemble des valeurs du type.

Tuple

Une définition de tuple est une paire ordonnée entete:tupval où entete est un ensemble de définitions d'attribut et tupval est un ensemble de valeur d'attributs conforme à l'entête.

Relation

Une relation (type relation) est une paire ordonnée entete:relval où relval est un ensemble de tuples, chacun ayant le même entête que celui de la relation.

Relvar

Une variable de relation (relvar) est une variable nommée de type relation. Nous distinguons deux catégories de relvars :

- ◊ La relvar de base : une variable dont la valeur est stockée et peut être modifiée.
- ◊ La relvar virtuelle : une variable dont la valeur est donnée par une expression relationnelle.

Conformément à l'usage, lorsque le contexte le permet sans ambigüité, le terme relation désigne parfois la valeur, le type ou la variable.

Algèbre relationnelle

L'algèbre relationnelle est formée des opérateurs de base suivants : renommage, projection, restriction, jointure, union et différence.

Contrainte

Une contrainte de relation est une expression booléenne définie sur un ensemble de relvars.

Description

Une description décrit un composant relationnel par un texte. Elle est représentée par un triplet <E, L, T> où E est une référence à un composant, L le code ISO 639 de la langue dans laquelle le texte est rédigé et T est le texte de l'annotation.

Affectation

L'affectation permet de remplacer la valeur d'une relvar par une relation de même entête dans la mesure où toutes les contraintes auxquelles la relvar participe demeurent avérées.

Automatisme

Un automatisme commande l'exécution d'une action sur la base d'une autre selon un mode (concurremment, avant, pendant, en lieu et place). Les actions visées comprennent obligatoirement l'affectation ; d'autres actions peuvent être définies (notamment diverses formes spécialisées d'affectation).

Schéma

Un ensemble de définitions d'entête de relvar et de contraintes auxquelles peuvent s'ajouter d'autres sortes de définitions permises par le modèle (types, opérateurs, automatismes).

Base de données

Un ensemble de schémas complété par l'ensemble des relvars qui y sont définies. Une base de données est valide si et seulement si toutes les contraintes sont avérées.

2.2.2 Langage

Le langage µRel diffère de *TutorialD* par certains ajustements syntaxiques (symboles équivalents à certains mots réservés) et par l'ajout d'une contrainte spécialisée et d'un automatisme spécialisé.

Opérateurs relationnels

Soit les expressions relationnelles `R`, R_0, R_1, les opérateurs relationnels de base sont :

- La restriction (notée `R WHERE condition`) permet de sélectionner les tuples d'une relation selon une expression booléenne (le symbole **σ** est équivalent à `WHERE`).
- La projection (notée `R` **π** `{listeAttributs}`) permet de composer une relation formée d'un sous-ensemble des attributs d'une autre (le symbole **π** est facultatif en µRel alors qu'il est omis en *TutorialD*).
- La jointure (notée R_0 `JOIN` R_1) permet de joindre deux relations selon la base de l'égalité des attributs communs de leurs tuples respectifs. Le symbole ⋈ est équivalent à `JOIN`.
- Le renommage (noté `R RENAME {listeRenommages}`) permet de renommer les attributs de l'entête de la relation. Le symbole **ρ** est équivalent à `RENAME`.

- L'union (notée `R0 UNION R1`) permet de faire l'union de deux relations. Le symbole `U` est équivalent à `UNION`.
- La différence (notée `R0 MINUS R1`) permet de faire la différence de deux relations. Le symbole – est équivalent à `MINUS`.

À ces opérateurs relationnels de base s'ajoutent de nombreux opérateurs supplémentaires définis à l'aide de cette base parmi lesquels :

- L'augmentation, ou extension, (notée `R EXTEND {listeAffectations}`) permet, pour chaque tuple de `R`, de modifier la valeur des attributs ciblés par la liste d'affectation ; lorsqu'un attribut apparaissant en partie gauche d'une affectation de la `listeAffectations` est absent de l'entête de la relation, il y est ajouté (et corolairement dans celle de chacun des tuples). Le symbole ξ est équivalent à `EXTEND`.
- L'image du tuple courant relativement à une relation `R` restreinte par un `WHERE` ou modifiée par un `EXTEND` (notée `!!R`), à savoir `R JOIN RELATION{TUPLE{*} π {x1, ..., xn}}` où
  - `TUPLE{*}` dénote soit le tuple couramment évalué dans la `condition` du `WHERE` soit le tuple modifié dans par la `listeAffectations` du `EXTEND` et
  - `{x1, ..., xn}` sont les attributs communs de `TUPLE{*}` et de `R`.

Contraintes spécialisées

TutorialD propose déjà deux contraintes spécialisées :

- Contrainte de clé candidate

  La contrainte spécialisée

```
R KEY {listeAtrributs}
```

  est équivalente à la contrainte générale :

```
#R = #(R π {listeAtrributs})
```

- Contrainte de clé référentielle

  En supposant la contrainte préalable `R1 KEY {listeAttributs}`, la contrainte spécialisée

```
FOREIGN KEY R0 {listeAttributs} REFERENCES R1
```

  est équivalente à la contrainte générale :

```
((R0π{listeAttributs}) ⊆ (R1π{listeAttributs}))
```

Au besoin l'opérateur de renommage peut être utilisé pour mettre les attributs de R_1 en accord avec ceux de R_0, par exemple

```
FOREIGN KEY R {a, b} REFERENCES S RENAME {x AS a, y AS b}
```

µRel en propose d'étendre la clé référentielle par une contrainte de participation :

◊ Contrainte de participation

Pour garantir le nombre d'associations effectives entre les valeurs d'une clé référentielle (domaine $listeAttributs_d$) et un ensemble d'attributs (portée $listeAttributs_p$) pris parmi les attributs de R_0, la contrainte de clé référentielle est étendue comme suit :

```
FOREIGN KEY R0 {listeAttributsd} REFERENCES R1
  [min..max] {listeAttributsp}
```

ce qui est équivalent à la contrainte générale :

```
(FOREIGN KEY R0 {listeAttributsd} REFERENCES R1)
  AND
  IS_EMPTY (
    EXTEND R1 π {listeAttributsd} :
      {nb := COUNT(!!(R0 π {listeAttributsd ∪ listeAttributsp}))}
    WHERE NOT (min <= nb AND nb <= max)
  )
```

Automatisme spécialisé

µRel propose d'étendre la contrainte référentielle par un automatisme spécialisé :

◊ Automatisme de propagation

La propagation d'une clé référentielle lorsque la relation cible est une relation-clé (relation dont la seule clé est formée de tous les attributs) peut être spécifiée par :

```
FOREIGN KEY R0 {listeAttributs0} REFERENCES R1 PROPAGATES
```

ce qui est équivalent à ajouter l'automatisme suivant :

```
ON INSERT i INTO R0 :
  INSERT (i π {listeAttributs0}) INTO R1
```

Note

Les deux extensions à la contrainte référentielle peuvent être combinées.

2.3 Conversion ontologique-relationnelle

Cette section décrit la méthode de conversion d'une ontologie formalisée selon µOnto en un schéma relationnel selon µRel à l'aide de règles générales de conversion et de règles spécifiques

de conversion. Les règles générales sont définies de manière à préserver la sémantique des axiomes à l'aide de la structure relationnelle et de contraintes adéquates. Les règles spécifiques sont définies pour optimiser certaines conversions dans certains cas particuliers.

2.3.1 Règles générales de conversion

Les composants ontologiques sont convertis en composants relationnels selon les règles décrites ci-dessous et illustrées à la Figure 10.

Type

Un type de données ontologique est converti en une relvar comprenant deux attributs : une clé artificielle dtid:dtid_type et la valeur value:relDatatype ; la valeur est également une clé.

Le dtid_type est configurable (typiquement `INTEGER`, `BIGINT`, `SERIAL`, `BIGSERIAL`, `UUID`). Chaque valeur de dtid identifie de manière unique une valeur. Le relDatatype est un type du modèle relationnel correspondant à un type du modèle ontologique. La liste de correspondance entre les types μOnto et les types μRel est configurable.

Classe

Une classe ontologique est convertie en une relvar comprenant un seul attribut dbid:dbid_type où dbid est un identifiant unique d'un individu ontologique au sein du schéma relationnel et dbid_type un type est configurable (typiquement `INTEGER`, `BIGINT`, `UUID`). En particulier, la classe ontologique Thing est convertie en une relvar Thing qui contient les identifiants de tous les individus ontologiques représentés dans la base de données.

Propriété de classe

Une propriété de classe est convertie en une relvar comprenant deux attributs : domain_dbid:dbid_type et range_dbid:dbid_type. La clé est composée des deux attributs. De plus, deux clés référentielles sont définies : une clé référentielle vers la relvar qui représente le domaine et une clé référentielle vers la relvar qui représente la portée.

Propriété de données

Les propriétés de données ne sont pas l'objet d'une conversion indépendante de leur utilisation dans un axiome. En effet, une telle conversion n'apporterait pas de bénéfice sans une prise en charge de la hiérarchie des types de données. Or, une telle hiérarchie n'est définie ni dans μOnto ni dans OWL.

Axiome d'héritage de classe

Un axiome d'héritage de classe est converti en une contrainte référentielle avec propagation depuis la relvar de la sous-classe vers la relvar de la superclasse.

Axiome d'héritage de propriété

Un axiome d'héritage de propriété est converti en une contrainte référentielle avec propagation depuis la relvar de la sous-propriété vers la relvar de la super-propriété.

Axiome d'association de classe

Un axiome d'association de classe est converti en une relvar comportant deux attributs : la clé candidate de la relvar qui représente le domaine de l'axiome (relvar du domaine) et la clé candidate de la relvar qui représente la portée de l'axiome (relvar de la portée). La clé candidate est composée des deux attributs.

De plus, trois clés référentielles sont définies : une clé référentielle vers la relvar du domaine, une clé référentielle vers la relvar de la portée et une clé référentielle vers la relvar qui représente la propriété de l'axiome. Ces clés référentielles sont complétées d'une contrainte de participation, le cas échéant.

Axiome d'association de données

Un axiome d'association de classe est converti en une relvar comportant deux attributs : la clé candidate de la relvar du domaine et la clé candidate de la relvar de la portée. La clé candidate est composée des deux attributs.

De plus, deux clés référentielles sont définies : une clé référentielle vers la relvar du domaine et une clé référentielle vers la relvar de la portée. Ces deux clés référentielles sont complétées d'une contrainte de participation, le cas échéant.

Annotation

Une annotation doit être utilisée pour documenter les composants du modèle de données et pour fournir plusieurs interfaces d'accès dans différentes langues à l'aide de vues.

Une annotation de description <E, description, L, T> est convertie en une description <E, L, T> dans le modèle µRel.

Une annotation d'étiquette <E, etiquette, L, T> est convertie en un nom de vue (vue d'étiquette) ou un nom d'attribut. Pour chaque annotation d'étiquette, une vue d'étiquette est définie sur la table de classe comme suit :

- le nom de la vue est l'étiquette de classe,

◊ l'entête de la vue est la projection de l'entête de la relvar de classe. La projection inclut le renommage de nom des attributs de classe relvar en fonction de leur annotation d'étiquette, c'est-à-dire que la clé dbid est renommée « <nom de la vue>_dbid ».

Contraintes

Les contraintes introduites lors de la réduction de la complexité des axiomes par la fonction co sont déjà garanties par la fermeture de modèle relationnelle en regard de toutes les classes générées (relvar) puisqu'aucun individu (tuple) de celles-ci ne peut être construit hors de la création d'un individu appartenant à une classe de l'ontologie d'origine. Ainsi, par construction « p β = Z », « (β ∩ γ) = Z » ou « Z = (β ∪ γ) » selon le cas, assurera *de facto* la contrainte correspondante « p β ⊆ Z », « (β ∩ γ) ⊆ Z » ou « Z ⊆ (β ∪ γ) ». Les contraintes peuvent donc être ignorées lors de la conversion.

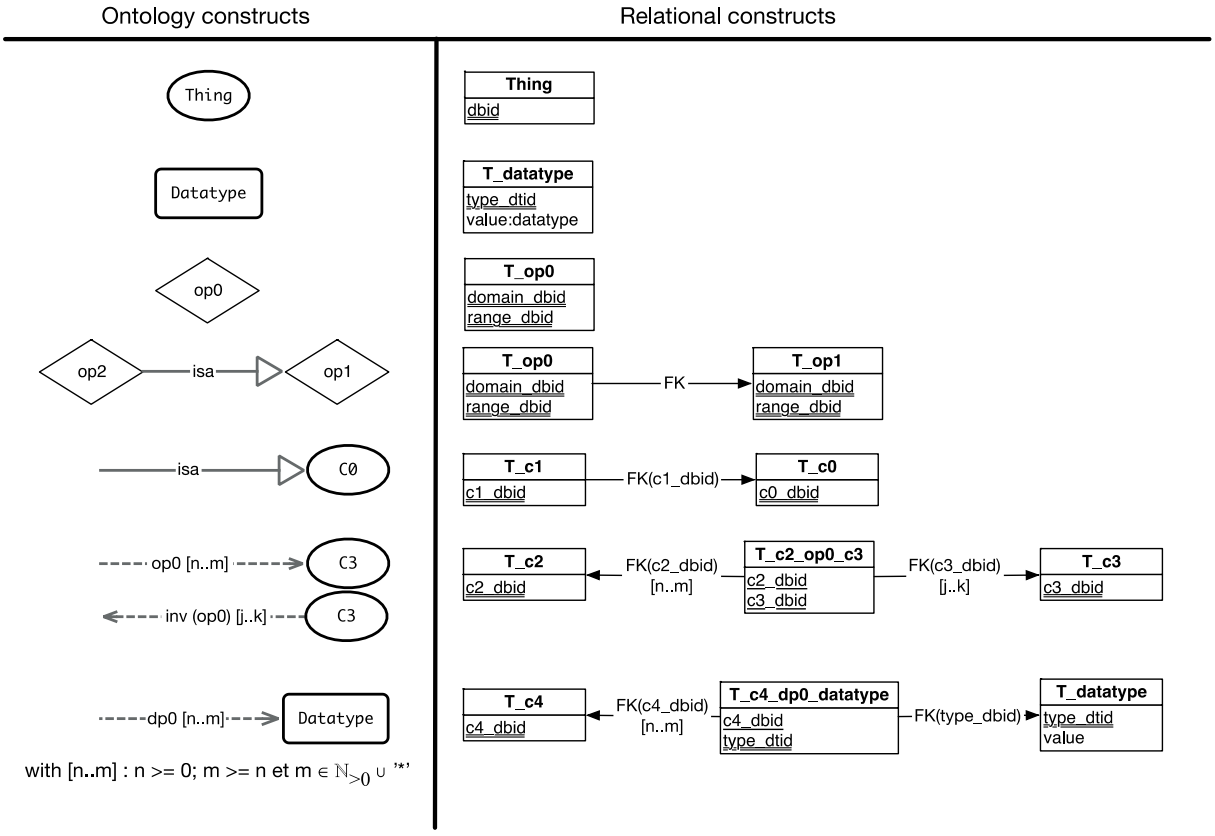


Figure 10. Illustration des règles de conversion ontologie-relationnel.

2.3.2 Règles spécifiques de conversion

La conversion peut être configurée pour effectuer des conversions plus spécifiques en fonction des caractéristiques de divers composants ontologiques.

Conversion spécifique des axiomes avec participation [1..1]

La règle spécifique de conversion concernant les axiomes avec la participation [1..1] consiste à créer un attribut supplémentaire au lieu de créer une variable associée (Figure 11). Plus spécifiquement, pour chaque axiome d'association : (1) la clé primaire de la relvar de la portée doit être ajoutée à la relvar du domaine et (2) une contrainte référentielle doit être ajoutée depuis la relvar du domaine vers la relvar de la portée.

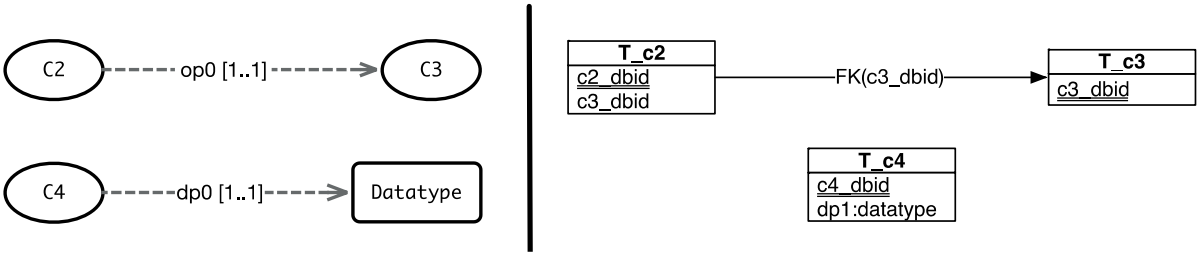


Figure 11. Illustration des règles spécifiques des axiomes avec une participation [1..1]

Conversion spécifique des types

Une conversion spécifique est possible pour les types de données si le stockage maximal requis pour une valeur de type de données est faible ou stable. Il peut être intéressant de représenter l'attribut de valeur directement dans la relvar d'association de données. Plus précisément, pour chaque axiome d'association de données, il s'agit d'ajouter à la relvar du domaine un attribut value:relDatatype. Le type de données µOnto est converti en un type de données µRel à l'aide d'une liste de correspondance.

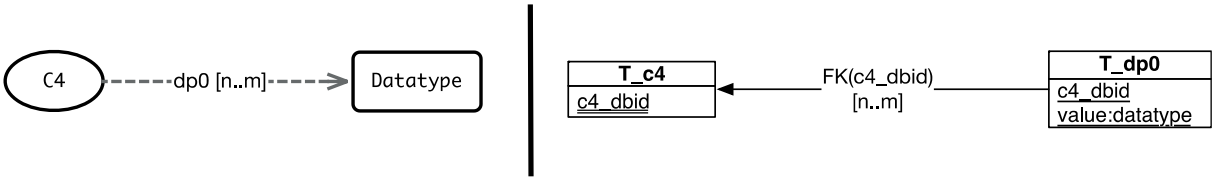


Figure 12. Illustration de la règle spécifique de conversion d'un type

2.3.3 Exemple

Voici un exemple tiré de l'ontologie PDRO. Un petit sous-ensemble de classes, de propriétés et d'axiomes est décrit ci-dessous à l'aide de µOnto, et une partie de la base de données générée est illustrée par un diagramme relationnel à la Figure 13.

```
PREFIXE : "http://purl.obolibrary.org/obo/"

ONTOLOGY :PDRO
  LABEL @en 'Drug prescription ontology'
```

```
PROPERTY :has_part
  LABEL @en 'has part'

PROPERTY :has_value
  LABEL @en 'has value'

CLASS :PDRO_0000001
  LABEL @en 'Healthcare prescription'
  :has_part [1..*] :IAO_0000302
  :has_part [1..*] :PDRO_0000003
  :has_part [1..*] :PDRO_0000005

CLASS :IAO_0000302
  LABEL @en 'Author identification'
  :has_value [1..1] :String

CLASS :PDRO_0000003
  LABEL @en 'Patient identification'
  :has_value [1..1] :String

CLASS :PDRO_0000005
  LABEL @en 'Document creation time identification'
  :has_value [1..1] Date

CLASS :PDRO_0000024
  LABEL @en 'Drug prescription'
  isa :PDRO_0000001
  :has_part [1..1] :PDRO_0000003
  :has_part [1..1] :PDRO_0000005
  :has_part [1..*] :PDRO_0000007

CLASS :PDRO_0000007
  LABEL @en 'drug administration and dispensing specification'
  :has_part [1..*] :PDRO_0010022

CLASS :PDRO_0010022
  LABEL @en 'drug administration specification'
  :has_part [1..*] :PDRO_0000060
  :has_part [1..*] :PDRO_0000103

CLASS :PDRO_0000060
  LABEL @en 'drug product specification'
  :has_part [1..*] (:PDRO_0000044 OR :PDRO_0040002)
  :has_part [1..*] :PDRO_0000097

CLASS :PDRO_0000044
  LABEL @en 'drug product name'
  :has_value [1..1] :String

CLASS :PDRO_0040002
  LABEL @en 'active ingredient name'
  :has_value [1..1] :String

CLASS :PDRO_0000097
  LABEL @en 'drug identification number'
  :has_value [1..1] :String

 CLASS :PDRO_0000103
  LABEL @en 'prescribed dosing specification'
  :has_part [1..*] :PDRO_0000190

 CLASS :PDRO_0000190
```

```
  LABEL @en 'prescribed dosing specification'
  :has_value [1..1] :String
```

L'axiome complexe défini dans la classe `:PDRO_0000060` doit être réduit avant la conversion relationnelle. Il en résulte les axiomes suivants traduits dans le schéma relationnel de la Figure 13 B :

```
[…]
CLASS :PDRO_0000060
  LABEL @en 'drug product specification'
  :has_part [1..*] :PDRO_0000044_U_PDRO_0040002
  :has_part [1..*] :PDRO_0000097

CLASS :PDRO_0000044_U_PDRO_0040002
  LABEL @en 'drug product name U active ingredient name'
  DESCRIPTION @en 'New union class of PDRO_0000044 and PDRO_0040002'

CLASS :PDRO_0000044
  LABEL @en 'drug product name'
  :has_value [1..1] :String
  isa :PDRO_0000044_U_PDRO_0040002

CLASS :PDRO_0040002
  LABEL @en 'active ingredient name'
  :has_value [1..1] :Strinh
  isa :PDRO_0000044_U_PDRO_0040002
[…]
```

De plus, certains axiomes de la classe `:PDRO_0000001` et `:PDRO_0000024` peuvent causer de la redondance au niveau relationnel (tuples en double) selon le cas 14 des règles de redondance (voir Tableau 5). Le résultat de la règle de réduction est le suivant (voir aussi Figure 13 A) :

```
[…]

CLASS :PDRO_0000001
  LABEL @en 'Healthcare prescription'
  :has_part [1..*] :IAO_0000302
[…]

CLASS :PDRO_0000024
  LABEL @en 'Drug prescription'
  isa :PDRO_0000001
  :has_part [1..1] :PDRO_0000003
  :has_part [1..*] :PDRO_0000003
  :has_part [1..*] :PDRO_0000005
 […]
```

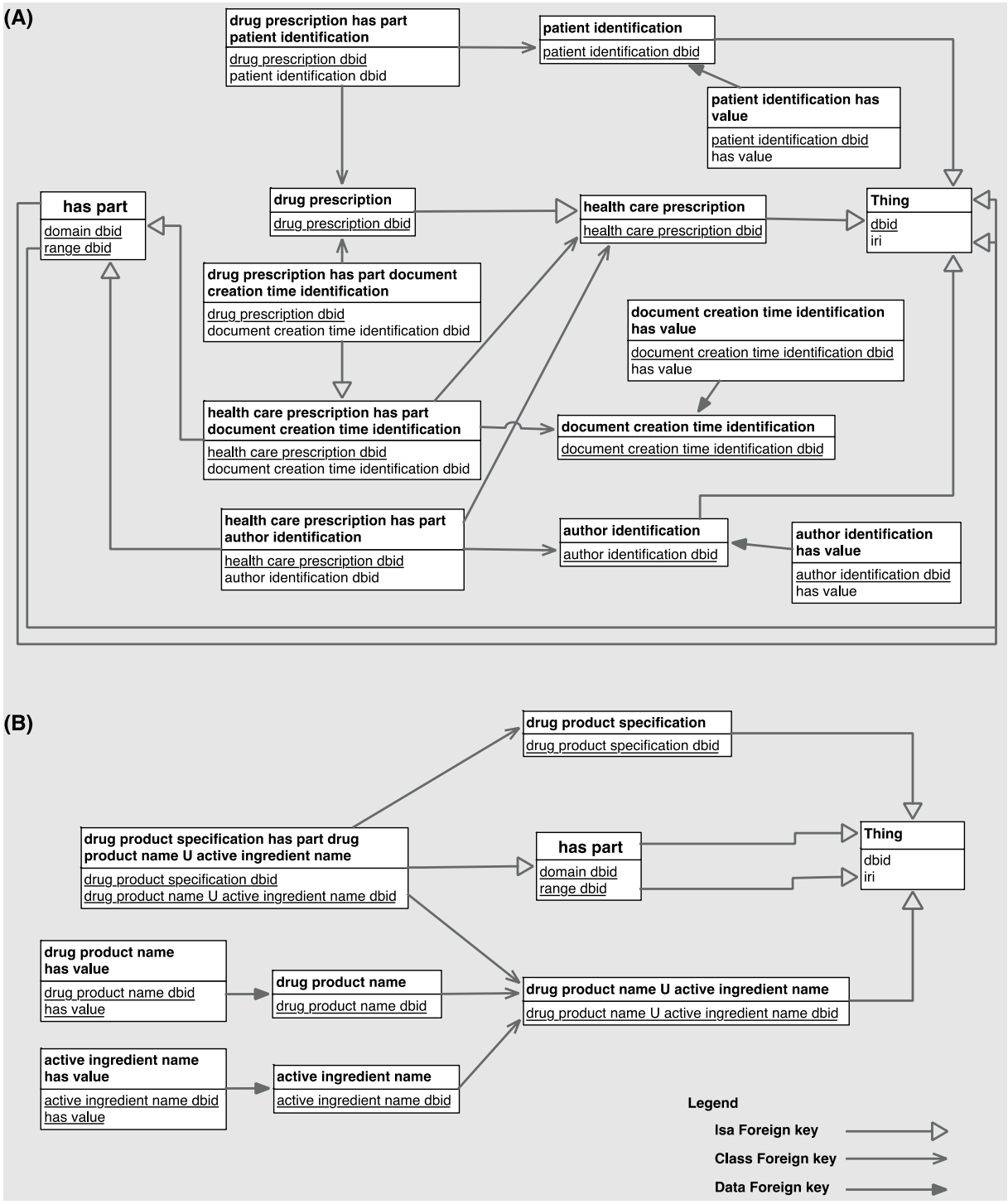


Figure 13. Diagramme relationnel d'une partie de la base de données générée

2.4 Historicisation

Le schéma initial (non historicisé) doit respecter certaines exigences afin de permettre une automatisation de l'historicisation. Le schéma initial doit contenir la définition des clés candidates et des clés référentielles ce qui permet la création automatique des contraintes temporelles. De plus, le schéma initial doit être en 5FN pour garantir que les doublons

engendrés par les données non historicisées soient éliminés. Le schéma final doit être en 6FN pour permettre le traitement des données absentes et l'évolution temporelle indépendante de chacun des attributs. Finalement, chaque relation contient un et un seul prédicat et donc une et une seule sémantique.

De manière informelle :

- ◊ Une relation est en 5FN si et seulement si chaque dépendance fonctionnelle applicable est induite par les clés candidates.
- ◊ Une relation de degré n est en 6FN si elle est en 5FN et si elle ne comprend aucune clé de degré inférieur à n-1.

Les règles générales de conversion de l'ontologie garantissent un schéma en 6FN et si les règles spécifiques de conversion sont appliquées, le schéma peut être en 5FN[30].

La méthode d'historicisation est esquissée dans le mémoire de maitrise [Khnaisser 2016]. La présente thèse y ajoute le passé indéterminé [Khnaisser et al. 2017b], incluant la structure et les contraintes qui en découlent ainsi que le traitement systématique de l'absence de données dans tous les cas de figure.

Dans la mesure où nous nous intéresserons spécifiquement à l'historicisation de schémas relationnels dérivés d'une ontologie selon la méthode décrite précédemment, il en découle certaines simplifications par rapport à la méthode générale d'historicisation. Ces simplifications seront mises à contribution tant pour optimiser le schéma temporalisé résultant que pour alléger la présentation. Elles seront explicitement signalées tout au long de ladite présentation.

2.4.1 Temporalisation

Dans [Khnaisser 2016], nous avons proposé un formalisme pour décrire un modèle relationnel temporalisé inspiré des travaux de [Snodgrass 2000; Date et al. 2014] ainsi qu'une méthode automatisée d'historicisation bitemporelle d'un modèle relationnel non temporalisé (*Unified Bitemporal Historicization Framework* – UBHF). Le formalisme unifiait la représentation syntaxique et sémantique des concepts temporels afin d'automatiser les étapes d'historicisation.

Nous complétons ici cette proposition avec le traitement de l'indétermination [Khnaisser et al. 2017b] en ajoutant deux catégorisations temporelles supplémentaires :

- ◊ Un attribut temporel de validité indéterminé dans le passé [@Vxe] (nommé *until* dans UBHF) où seule la fin de la période de validité de la proposition est connue.

[30] Les règles spécifiques produisent des attributs pour les axiomes avec participation [1..1] au lieu de produire une relvar d'association.

◊ Un attribut temporel de validité indéterminé [@Vxx] où ni le début et ni la fin de la période de validité de la proposition sont connus. Cet attribut représente le fait que la proposition est (a été, sera) avérée et que la période de validité en est inconnue.

Le Tableau 6 ci-dessous résume les catégories d'attributs temporels du modèle d'historicisation.

Tableau 6. Nouvelle catégorisation des attributs temporels

Notation	Définition	Type
@V	Attribut temporel de validité	PERIOD
@Vbx	Attribut temporel de validité indéterminé dans le futur	INSTANT
@Vbe	Attribut temporel de validité déterminé	PERIOD
@Vxe	Attribut temporel de validité indéterminé dans le passé	INSTANT
@Vxx	Attribut temporel de validité indéterminé	sans objet
@T	Attribut temporel de transaction	PERIOD
@Tbx	Attribut temporel de transaction indéterminé dans le futur	INSTANT
@Tbe	Attribut temporel de transaction déterminé	PERIOD

La Figure 14 ci-dessous montre toutes les catégories possibles d'une relation temporelle.

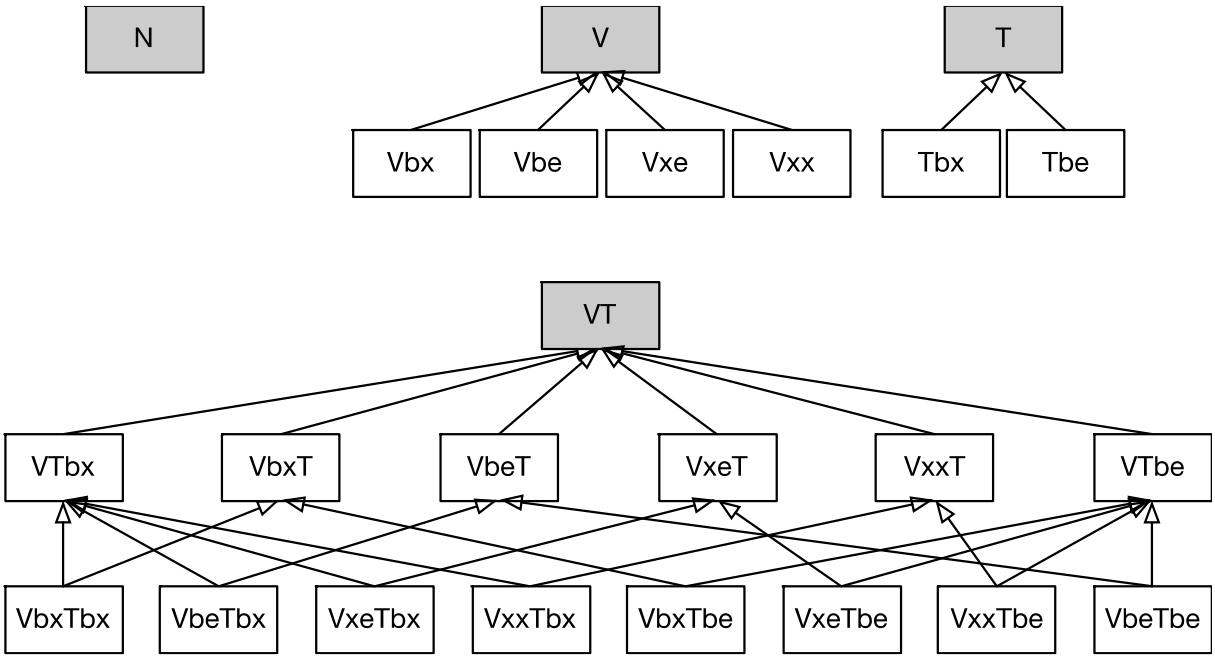


Figure 14. Nouvelle catégorisation des relations dans un modèle relationnel temporel

Les catégories temporelles finales comprennent donc en plus des catégories présentées précédemment (voir Figure 6) les catégories suivantes : Vxe, Vxx, VxeTbx, VxeTbe, VxxTbx et VxxTbe.

2.4.2 Structure

Afin d'assurer l'uniformité d'interprétation des requêtes au sein d'un schéma temporel, les relations qu'il contient doivent toutes être historicisées selon la même catégorie primaire (@V, @T ou @VT). Ainsi, nous distinguerons donc trois catégories de schémas temporels de haut niveau dans le cadre d'UBHF :

- ◊ Un schéma unitemporel de validité .
- ◊ Un schéma unitemporel de transaction.
- ◊ Un schéma bitemporel.

L'historicisation s'applique seulement pour les relvars de classes et des relvars d'association.

2.4.2.1 Décomposition

La décomposition consiste à construire un ensemble de relparts afin de représenter l'historicisation (selon une catégorie temporelle primaire S) d'une relvar R(K) du schéma non temporalisé. Les règles générales de décomposition présentées dans [Khnaisser 2016; Khnaisser et al. 2017a] sont exprimées en termes de R@S(K, B, C, D_V, D_T) où B et C sont des sous-catégories des attributs non-clés.

Or, les relvars du schéma non temporalisé, induites par les règles de conversion générales, sont toutes des relvars-clés. Nous en tirerons parti ici en omettant le traitement des sous-catégories d'attributs non clés B et C puisqu'il n'y a pas d'attributs non-clé. Si les règles spécifiques de conversion devaient être utilisées, il suffirait d'utiliser les règles générales de décomposition. Il faut toutefois noter que les gains attendus des règles spécifiques de conversion s'en trouveront significativement diminués, voire annihilés, car les relvars économisées initialement seront pour la plupart réintroduites.

La construction est basée sur les transformations de restriction-union et de projection-jointure qui permettent de garantir la composition (union et jointure) et la décomposition (restriction, projection) des relations sans perte de données [Date et al. 2014]. Le processus de décomposition est illustré à la Figure 15 ci-dessous.

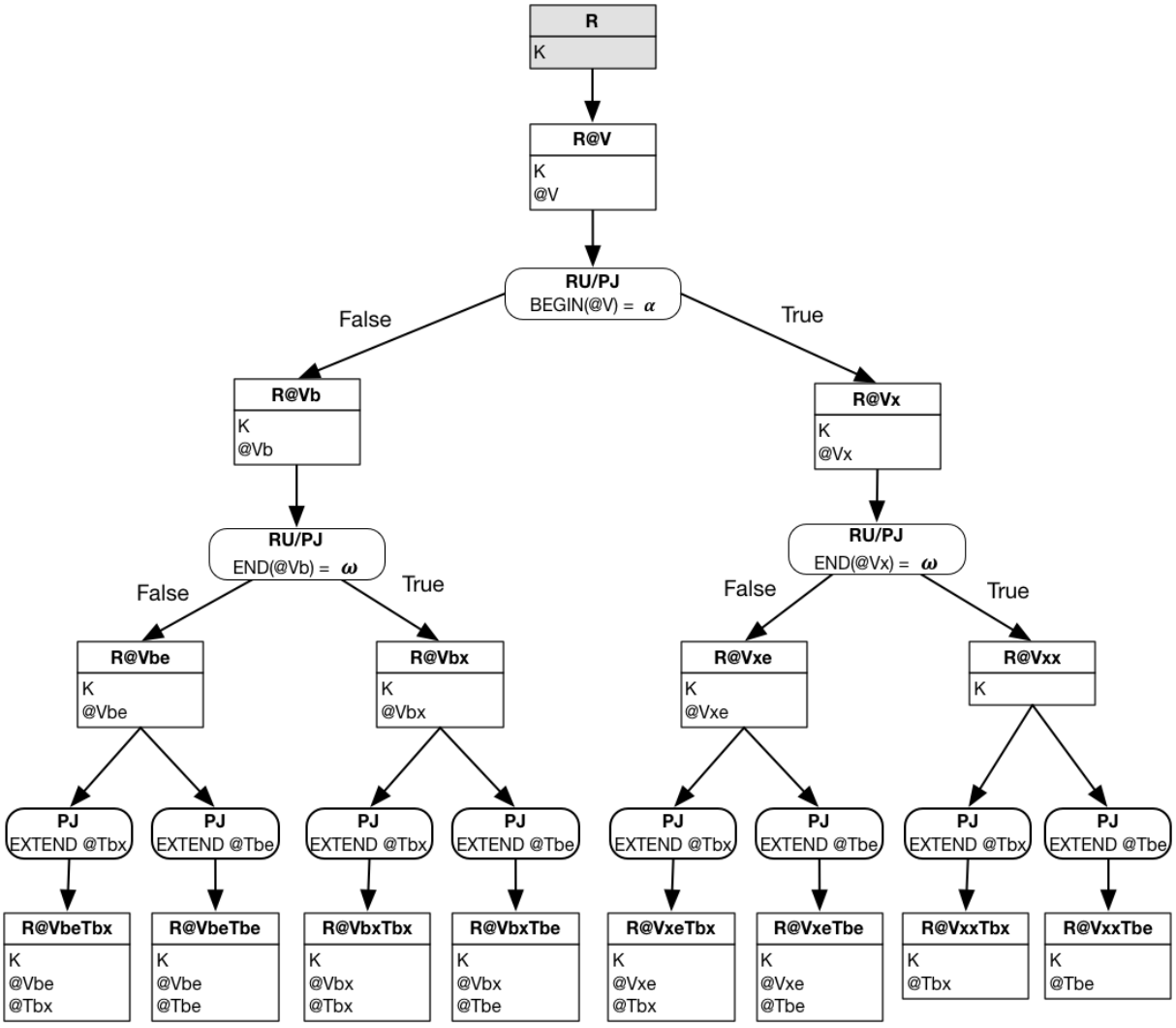


Figure 15. Illustration des étapes de décomposition

Les Figure 16, Figure 18 et Figure 18 illustrent respectivement le résultat final de l'historicisation unitemporelle de validité , unitemporelle de transaction et bitemporelle d'une relvar.

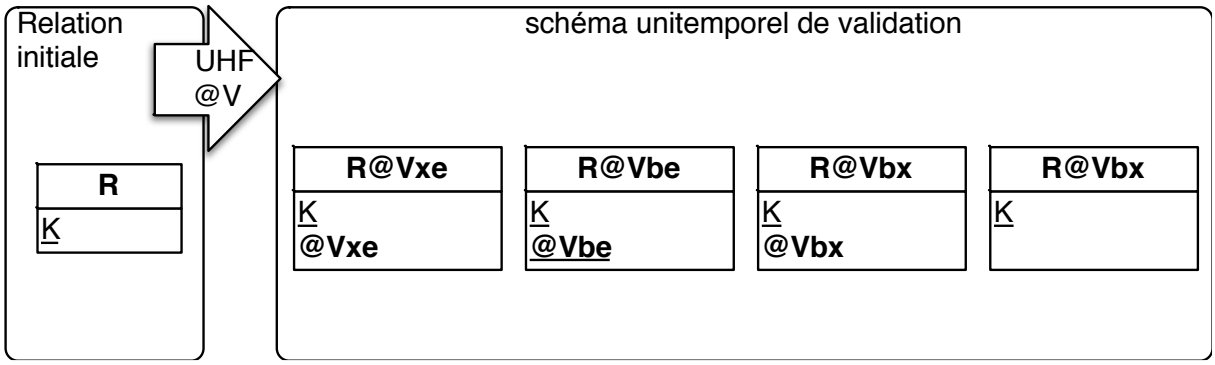


Figure 16. Illustration du résultat du processus d'historicisation unitemporelle de validité

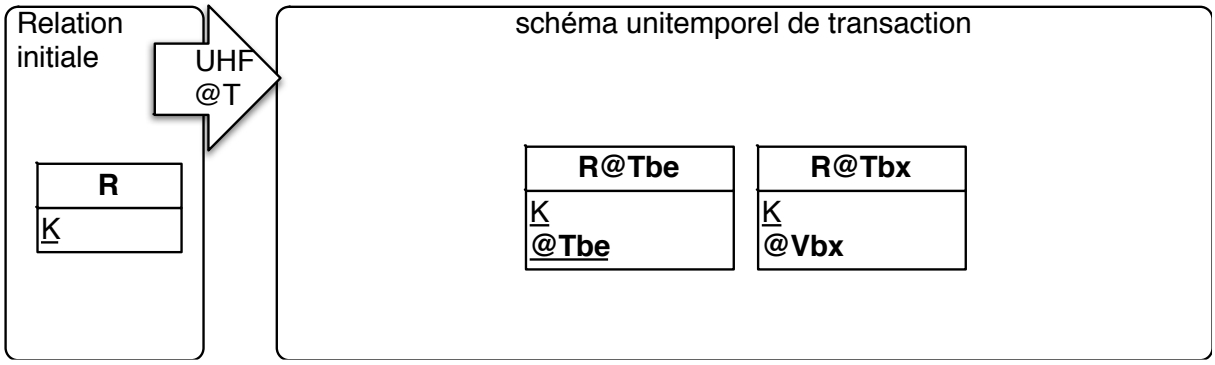


Figure 17. Illustration du résultat du processus d'historicisation unitemporelle de transaction

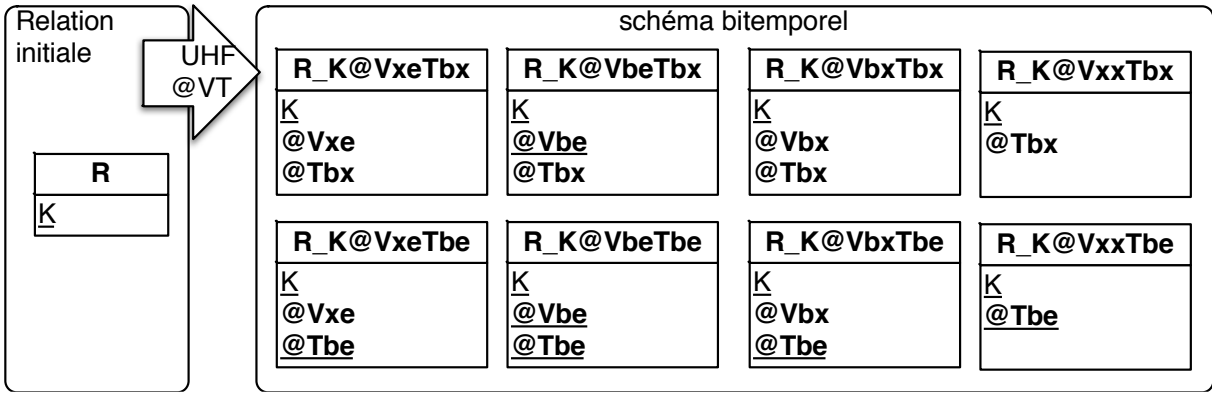


Figure 18. Illustration du résultat du processus d'historicisation bitemporelle

2.4.2.2 Opérateur *gHistory*

Dans la mesure où les relvars temporalisées ne comportent qu'un seul regroupement (celui de la clé), l'opérateur *gSpace* présenté dans l'article [Khnaisser et al. 2017b] peut être simplifié (et remplacé) par l'opérateur *gHistory* défini ci-après. L'opérateur *gHistory* permet de construire l'histoire temporelle d'une relvar en calculant les périodes temporelles à partir des périodes de chacune des relvars temporelles dérivées d'une même relvar non temporalisée. Le traitement des relvars dérivées qui ne comportent pas de périodes est le suivant :

- ◊ Pour une relvar @Vxe, la période correspondante est définie par @V := [*saw*:@Vxe].
- ◊ Pour une relvar @Vbx, la période correspondante est définie par @V := [@Vbx:*ufn*].
- ◊ Pour une relvar @Vxx, la période correspondante est définie par @V := [*saw*:*ufn*].

Variables ufn et saw

Une période indéterminée dans le passé @Vxe est notée [*saw:*e] où e est l'instant de fin connu et *saw* (pour « *since a while* ») est une variable qui représente l'instant de début inconnu. La

valeur effective de *saw* ne doit jamais être utilisée dans un autre contexte qu'un début de période indéterminée dans le passé.

Une période indéterminée dans le futur (@Vbx ou @Tbx) est notée [b:*ufn*] où b est l'instant du début connu et *ufn* (pour « *until further notice* ») est une variable qui représente l'instant de fin inconnu. La valeur effective de *ufn* ne doit jamais être utilisée dans un autre contexte qu'un début de période indéterminée dans le futur. Remarquons que *ufn* ne peut être antérieur à *now* dans le cas d'un @Tbx.

Hors d'un contexte temporel spécifique, la valeur de *saw* (resp. *ufn*) est remplacée par α [la valeur minimale du type `INSTANT` utilisée par le schéma relationnel] (resp. ω [la valeur maximale du type `INSTANT` utilisé par le schéma relationnel]). Par exemple, dans le contexte d'un patient, la date de naissance peut être assignée à *saw* et, s'il est décédé, la date de son décès à *ufn*. Le mécanisme d'établissement de contexte n'est pas prescrit par *TutorialD*. Chaque mise en œuvre doit spécifier le sien.

Définition de l'opérateur

Pour une relvar R, l'histoire unitemporelle de validité est construite en unissant les trois relvars temporelles R@Vxe, R@Vbx et R@Vbe et la relvar R@Vxx comme suit :

- ◊ Pour la relvar @Vxe, l'attribut temporel @Vxe est remplacé par un intervalle @V := [*saw*:@Vxe].
- ◊ Pour la relvar @Vbx, l'attribut temporel @Vbx est remplacé par un intervalle @V := [@Vbx: *ufn*].
- ◊ Pour la relvar @Vxx, un attribut temporel est ajouté @V := [*saw*: *ufn*].
- ◊ Pour la relvar @Vbe, l'attribut @Vbe est renommé @V.
- ◊ Les relvars sont réunies : R@Vxe ∪ R@Vbx ∪ R@Vxx ∪ R@Vbe.

```
OPERATOR gHistory@V (R RELATION) RETURNS RELATION(K, @V)
  WITH (
    r_xe := (EXTEND R@Vxe : {@V:=[saw:@Vxe]}) {K, @V},
    r_bx := (EXTEND R@Vbx : {@V:=[@Vbx:ufn]}) {K, @V},
    r_xx := (EXTEND R@Vxx : {@V:=[saw:ufn]}) {K, @V},
    r_be := R@Vbe RENAME {@Vbe AS @V}
    ): USING (@V): r_xe UNION r_bx UNION r_xx UNION r_be
  END IF
  END OPERATOR
```

Pour construire l'histoire unitemporelle de transaction d'une relvar R, il faut unir les deux relvars R@Tbx et R@Tbe comme suit :

- ◊ Pour la relvar R@Tbx, l'attribut temporel @Tbx est remplacé par un intervalle @T := [@Tbx:*now*] où *now* est l'instant où une transaction (la requête) est réputée avoir lieu.

◊ Pour la relvar R@Tbe, l'attribut @Tbe est renommé @T.

◊ Les relvars sont réunies R@Tbx ∪ R@Tbe.

```
OPERATOR gHistory@T (R RELATION) RETURNS RELATION(K, @T)
  WITH (
    r_bx := (EXTEND R@Vbx : {@V:=[@Tbx:now]}) {K, @T},
    r_be := (R@Vbe RENAME {@Tbe AS @T}) {K, @T}
    ): USING (@T): r_xe UNION r_bx UNION r_be
  END IF
  END OPERATOR
```

Pour construire l'histoire bitemporelle d'une relvar R, il faut unir les huit relvars R@VxeTbx, R@VxeTbe, R@VbxTbx, R@VbeTbx, R@VbeTbe, R@VxxTbx et R@VxxTbe comme suit :

```
OPERATOR gHistory@VT (R RELATION) RETURNS RELATION(K, @V, @T)
  WITH (
    r_vxetbx :=
      (EXTEND R@VxeTbx : {@V:=[saw:@Vxe], @T=[@Tbx, now]})
      {K, @V, @T},
    r_vxetbe :=
      (EXTEND R@VxeTbe : {@V:=[saw:@Vxe]}) RENAME {@Tbe AS @T}
      {K, @V, @T},
    r_vbxtbx :=
      (EXTEND R@VbxTbx : {@V:=[@Vbx:ufn], @T=[@Tbx, now]})
      {K, @V, @T},
    r_vbxtbe :=
      (EXTEND R@VbxTbe : {@V:=[@Vbx:ufn]}) RENAME {@Tbe AS @T}
      {K, @V, @T},
    r_vxxtbx :=
      (EXTEND R@VxxTbx : {@V:=[saw:ufn], @T=[@Tbx, now]})
      {K, @V, @T}
    r_vxxtbe :=
      (EXTEND R@VxxTbe : {@V:=[saw:ufn]}) RENAME {@Tbe AS @T}
      {K, @V, @T}
    r_vbetbx :=
      (EXTEND R@VbeTbx : {@T=[@Tbx, now]}) RENAME {@Vbe AS @V}
      {K, @V, @T}
    r_vbetbe :=
      R@VbeTbe RENAME {@Vbe AS @V, @Tbe AS @T}
      {K, @V, @T}
    ): USING (@V, @T):
        r_vxetbx UNION r_vxetbe
        UNION
        r_vbxtbx UNION r_vbxtbe
        UNION
        r_vxxtbx UNION r_vxxtbe
        UNION
        r_vbetbx UNION r_vbetbe
  END IF
  END OPERATOR
```

2.4.3 Contraintes temporelles

L'introduction d'attributs de type intervalle (ce que sont les attributs temporels) dans une relvar nécessite l'adaptation des contraintes de clés candidates et référentielles. En outre, elle peut

induire des problèmes de redondance, de circonlocution, de contradiction et de non-compacité [Date et al. 2014] qui nécessitent l'introduction de trois nouvelles contraintes :

- ◊ Cohérence : est garantie par la non-redondance et la non-contradiction :
  - o Non-redondance : une redondance se produit lorsque les périodes de deux tuples de valeur équivalente se chevauchent.
  - o Non-contradiction : une contradiction se produit lorsque les périodes de deux tuples ayant des valeurs de clé identiques et des valeurs d'attributs non-clés différentes se chevauchent.
- ◊ Non-circonlocution : une circonlocution se produit lorsque les périodes de deux tuples de valeur équivalente se jouxtent (l'un suit immédiatement l'autre).
- ◊ Compacité : une non-compacité se produit lorsque la période d'un attribut dépendant d'une clé n'est pas un sous-ensemble de la période de la clé. La contrainte de non-compacité est même plus stricte dans le cas d'une relation obtenue par décomposition de projection-jointure (comme cela est fait lors du passage de la 5FN à la 6FN) : les deux périodes doivent en fait être égales.

Finalement, il faut également prendre en compte les contraintes liées à l'héritage qui sont introduites dans la transposition des ontologies (héritage conjoint et héritage disjoint).

La mise en œuvre du temps de transaction peut être entièrement prise en charge par le SGBD à même son mécanisme de journalisation [Date et al. 2014]. Il est donc suffisant de définir les contraintes temporelles pour les seules relvars de catégorie {N, Vxe, Vbx, Vbe, Vxx}. C'est ce que traitent les rubriques suivantes.

Clé candidate

La clé candidate d'une relvar de catégorie S parmi {N, Vxe, Vbx, Vxx} ne comporte pas d'attribut temporel :

```
CONSTRAINT R@S_key R@S KEY {K};
```

La clé candidate d'une relvar de catégorie S parmi {Vbe} comporte l'attribut temporel de validité :

```
CONSTRAINT R@S_key R@S USING(@Vbe): KEY {K, @Vbe};
```

Clé référentielle

Soit une clé référentielle non temporelle `FOREIGN KEY Ro {K} REFERENCES R`, la contrainte référentielle temporalisée correspondante est donnée par :

```
CONSTRAINT Ro-R_fk
  USING (@V) : gHistory@V(Ro) ⊆ gHistory@V(R);
```

ce qui est équivalent à

```
CONSTRAINT Ro-R_fk
  USING (@V) :
    FOREIGN KEY gHistory@V(Ro) {K, @V} REGERENCES gHistory@V(R);
```

et qui a l'avantage, grâce à la clause `USING (@V)`, de permettre l'ajout d'une contrainte de participation `[min..max] {listeAttributsp}`, sans adaptation supplémentaire.

Contrainte de cohérence

La contrainte de cohérence temporelle permet de garantir la non-redondance et la non-contradiction de la clé dans le temps.

```
CONSTRAINT R_temporal-uniqueness
  IS_EMPTY (R@Vbx {K, @Vbx} JOIN R@Vbe {K, @Vbe}
    WHERE PRE(@Vbx) < POST(@Vbe)) AND
  IS_EMPTY (R@Vbe {K, @Vbe} JOIN R@Vxe {K, @Vxe}
    WHERE POST(@Vxe) > PRE(@Vbe)) AND
  IS_EMPTY (R@Vxe {K, @Vxe} JOIN R@Vbx {K, @Vbx}
    WHERE POST(@Vxe) > PRE(@Vbx));
```

De plus, par définition, R@Vxx doit être disjointe de R@Vxe, R@Vbx et R@Vxx.

```
CONSTRAINT R_gHistory-uniqueness
  IS_EMPTY DISJOINT(R@Vxx{K}, (R@Vxe{K} UNION R@Vbx{K} UNION R@Vbe));
```

Contrainte de non-circonlocution

La contrainte de non-circonlocution temporelle permet de garantir la non-circonlocution de la clé dans le temps.

```
CONSTRAINT R_circumlocution
    IS_EMPTY (R@Vbx JOIN R@Vbe WHERE PRE(@Vbx) = POST(@Vbe)) AND
    IS_EMPTY (R@Vbe JOIN R@Vxe WHERE POST(@Vxe) = PRE(@Vbe)) AND
    IS_EMPTY (R@Vxe JOIN R@Vbx WHERE POST(@Vxe) = PRE(@Vbx));
```

Contrainte de compacité

La contrainte de compacité n'est utilisée que dans les cas où la règle spécifique de conversion concernant les axiomes avec la participation [1..1] a été appliquée (dans les autres cas, ce sont des relations-clés qui n'ont pas à être décomposées par projection-jointure).

Soit la décomposition de projection-jointure de R en sa relation-clé R_K et ses parties {R_a_1, ..., R_a_n}, une contrainte entre chaque relvar R_a_i et sa relation-clé R_K doit être ajoutée :

```
CONSTRAINT R-K-ai_denseness
  USING (@V) : gHistory@V(R_K) = gHistory@V(R_ai);
```

Contrainte d'héritage général

Dans le cas d'héritage, la contrainte suivante doit être définie entre chacune des relvars des sous-classes R_i et la relvar de la superclasse R :

```
CONSTRAINT R-Ri_inheritance
  USING (@V) :
    FOREIGN KEY gHistory@V(Ri) {K, @V} REGERENCES gHistory@V(R)
```

***Contrainte d'héritage disjoint* (*disjoint*)**

Dans le cas d'un héritage disjoint, une contrainte supplémentaire doit être ajoutée à la contrainte d'héritage général afin de garantir l'exclusion mutuelle temporelle :

```
CONSTRAINT R-Ri_inheritance-d
  USING (@V) :
    D_UNION{gHistory_Z(R1), ... gHistory_Z(Rn)} ⊆ gHistory_Z(R)
```

Automatismes

L'automatisme d'insertion n'est applicable qu'aux cas d'héritage. Il est fortement recommandé puisqu'il simplifie la gestion des clés et ne peut introduire d'erreur. La contrainte d'héritage général pourrait donc être

```
CONSTRAINT R-Ri_inheritance
  USING (@V) :
    FOREIGN KEY gHistory@V(Ri) {K, @V} REFERENCES gHistory@V(R)
      PROPAGATES ON INSERT
```

au sein de laquelle l'interprétation de l'automatisme est la suivante :

```
USING (@V) :
    ON INSERT i INTO Ri :
      INSERT (i π {K, @V}) INTO R
```

L'automatisme de retrait est applicable tant aux clés référentielles associées aux propriétés qu'à l'héritage. Cet automatisme est discutable puisqu'il entraine une perte d'information. On remarquera que l'information ne peut cependant plus être pertinente, puisque la clé (et conséquemment l'individu qu'elle représente) est réputée ne plus exister. Cette possibilité devrait au moins être offerte sous le contrôle d'un paramètre. Dans un contexte temporel, l'extension

```
CONSTRAINT X
  USING (@V) :
    FOREIGN KEY gHistory@V(Ri) {K, @V} REFERENCES gHistory@V(R)
      PROPAGATES ON DELETE
```

doit être interprétée comme suit :

```
USING (@V) :
    ON DELETE d FROM R :
      DELETE d ⋈ Ri FROM Ri
```

L’automatisme de retrait peut s’ajouter à celui d’insertion dans le cas de l’héritage.

3
Prototype

Cette section présente le prototype développé pour construire automatiquement un schéma relationnel historicisé à partir d'une ontologie. Le processus de construction est paramétrable par l'entremise de fichiers de configuration. La conception, l'implémentation et une évaluation du prototype sont également présentées.

3.1 Présentation

Afin de montrer la faisabilité et l'applicabilité de notre méthode, un prototype, OntoRelα, a été développé. OntoRelα génère à partir d'une ontologie OWL et de fichiers de configuration : (1) des scripts pour une base de données relationnelle, (2) des listes d'avertissements, (3) un dictionnaire d'arrimage (OntoRelDic), et (4) une ontologie normalisée formalisée selon μOnto. La Figure 19 illustre les entrées et les sorties d'OntoRelα.

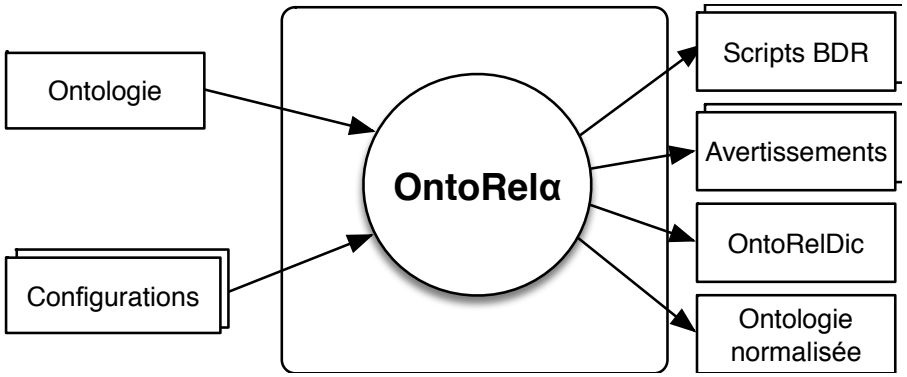


Figure 19. Diagramme de contexte d'OntoRelα

3.2 Conception

OntoRelα est constitué de cinq processus qui sont illustrés à la Figure 20 et décrits ci-dessous. L'ontologie est analysée et vérifiée, puis chaque composant ontologique est converti en un composant relationnel et, finalement, le schéma est historicisé selon la méthode d'historicisation. À l'issue de ces processus, des scripts et des fichiers sont générés pour permettre d'analyser l'exécution des processus, de mettre en œuvre une base de données relationnelle dérivée de l'ontologie et de décrire l'arrimage entre les composants de l'ontologie et les composants relationnels.

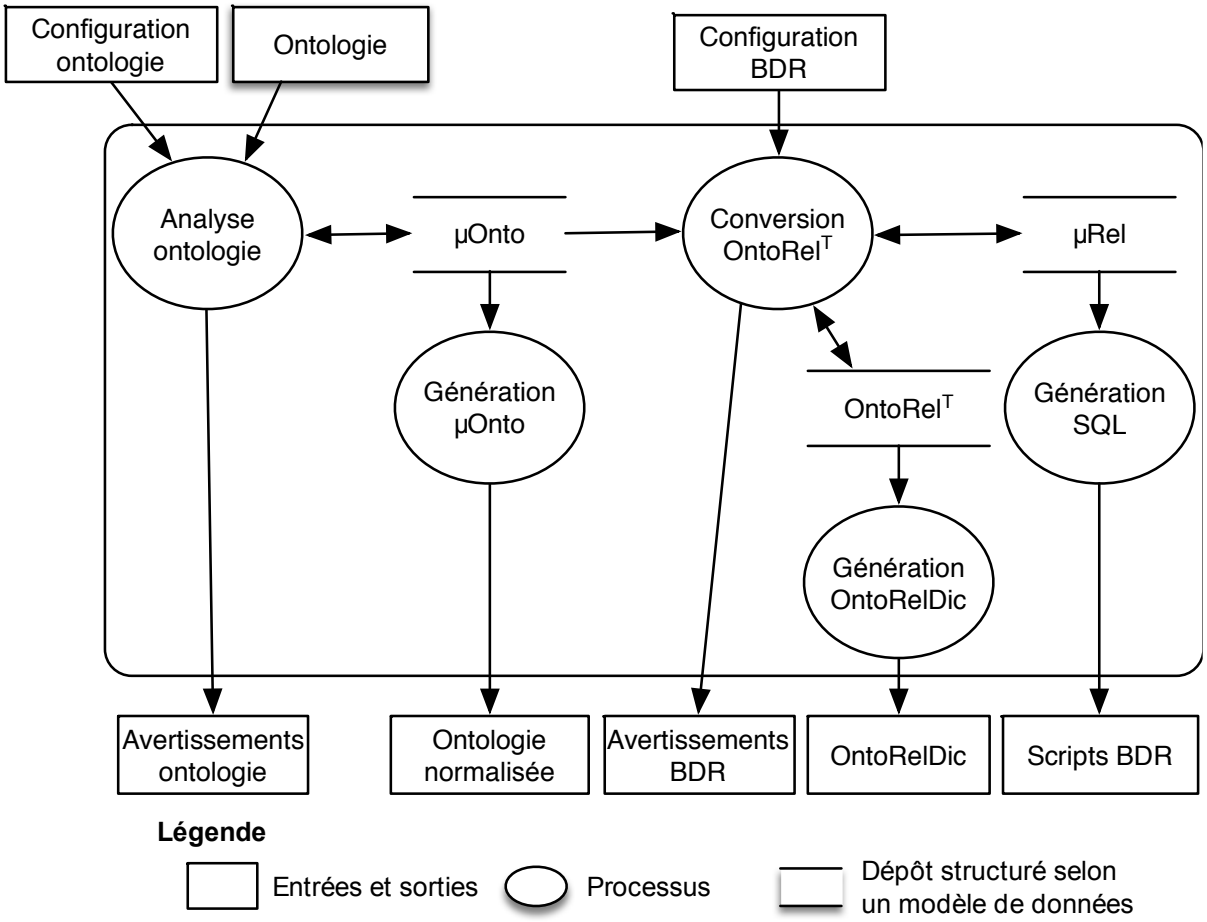


Figure 20. Diagramme de flux de données d'OntoRela

3.2.1 Création de µOnto

Analyse de l'ontologie

Le processus d'analyse de l'ontologie se déroule en trois étapes :

1. Traduction de l'ontologie selon une représentation conforme au modèle µOnto tel qu'illustré par un diagramme EAE à la Figure 21 (où quelques attributs parmi les plus importants sont présentés). La traduction produit des avertissements suivants :
   - ◊ une liste de toutes les classes et propriétés sans étiquettes ni annotations de définition,
   - ◊ une liste de toutes les classes sans axiome d'association de données,
   - ◊ une liste des messages émis par l'étape de réduction.
2. Filtrage de la représentation µOnto. L'étape est pilotée grâce aux paramètres suivants du fichier de configuration ONTO :
   - ◊ la liste de classes et de propriétés d'intérêt (facultative),
   - ◊ la liste des annotations d'intérêt spécifiées par type et par langue (facultative).

3. Création du graphe ontologique orienté (*OntoGraph*) avec les classes (*OntoClass*) et les types (*OntoDatatype*) comme sommets et les axiomes comme arêtes.
4. Vérification de l'instance µOnto. Le graphe est utilisé pour effectuer plusieurs diagnostics et calculs :
   - ◊ vérification de la connectivité du graphe et l'identification des classes « orphelines » (si graphe non connexe) ;
   - ◊ identification des classes n'ayant pas d'axiome d'association de données ;
   - ◊ identification des classes qui n'ont pas de sous-classes ;
   - ◊ identification des classes avec héritage multiple.

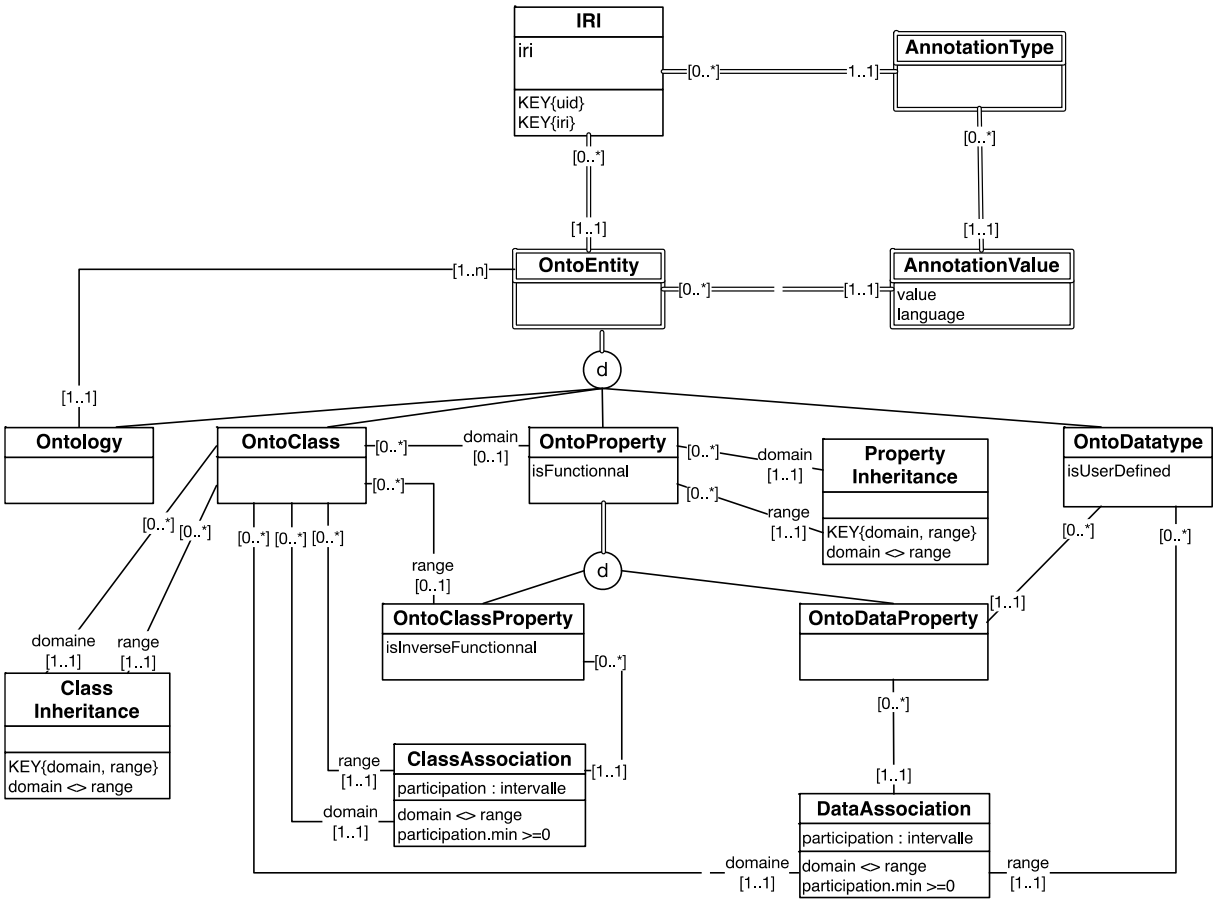


Figure 21. Diagramme EAE d'une ontologie selon µOnto.

Génération de µOnto

Le processus de génération µOnto produit une ontologie normalisée selon la grammaire µOnto qui est décrite à l'annexe 1.

3.2.2 Création de µRelT et de OntoRelT

Conversion OntoRelT

Le processus de conversion OntRelT se déroule en quatre étapes :

1. Traduction de l'ontologie µOnto en une représentation conforme au modèle µRelT illustré par un diagramme EAE à la Figure 22 (où quelques attributs parmi les plus importants sont présentés) et la création de l'OntoRelT, une structure simple d'arrimage entre les composants de µOnto et leur contrepartie µRelT historicisée illustrée par un diagramme EAE à la Figure 23.

   L'étape est pilotée grâce aux paramètres suivants du fichier de configuration BDR :

   ◊ l'identifiant du schéma de base,

   ◊ la liste des langues pour la production des schémas d'interface,

   ◊ le nom du SGBDR cible,

   ◊ l'identifiant de la clé primaire et son type,

   ◊ le mode de normalisation des types,

   ◊ la liste de correspondance entre les types µOnto et les types µRel.

2. Historicisation de µRel historicise les schémas relationnels selon la catégorisation temporelle du schéma (@V, @T, @VT).

3. Vérification des composants de la représentation µRel. Les avertissements suivants sont produits :

   ◊ une liste des types de données sans arrimage,

   ◊ une liste des identifiants dont la longueur dépasse le nombre maximal de caractères autorisés par le SGBD cible.

4. Création du graphe relationnel orienté (*RelGraph*) avec les relvars comme sommets et les contraintes référentielles comme arêtes.

5. Création du graphe ontologique-relationnel (*OntoRelGraph*) avec les *ClassRelvar* et les *DatatypeRelvar* comme sommets et les *AxiomRelvar* comme arêtes.

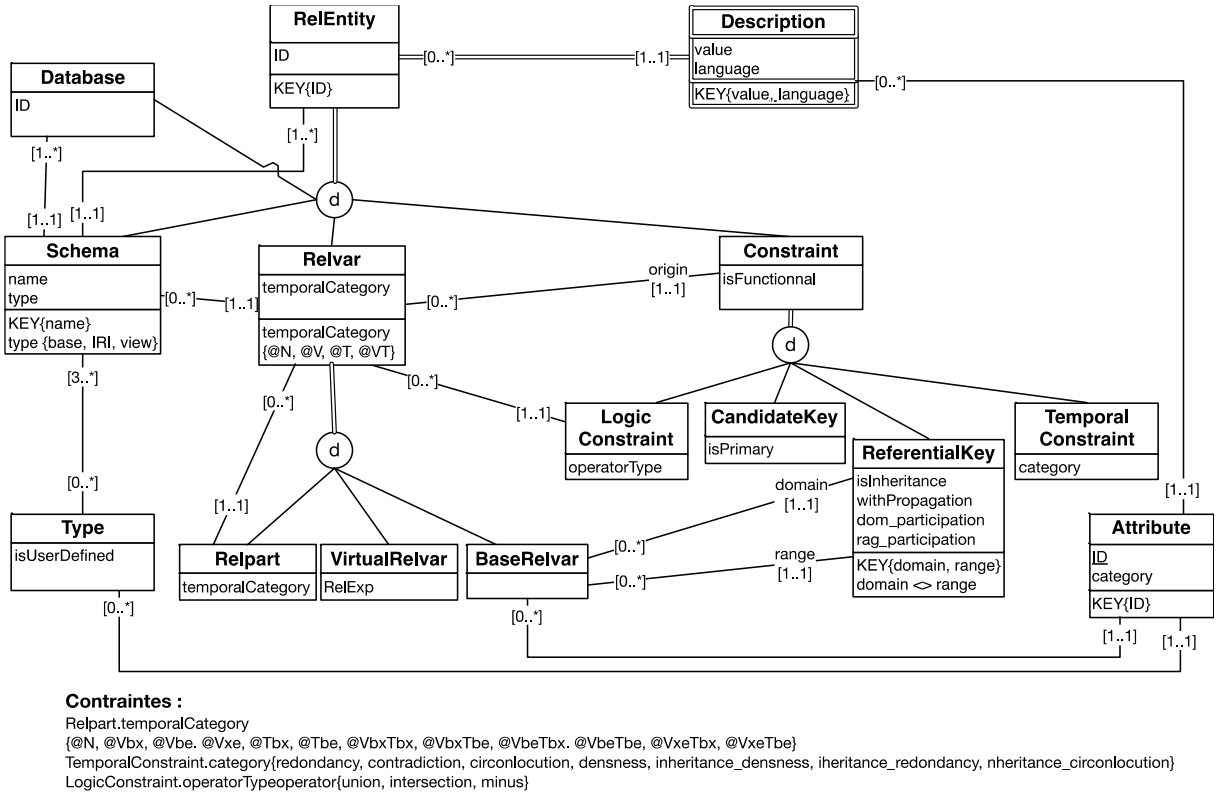


Figure 22. Diagramme EAE d'un modèle relationnel selon µRel$_T$.

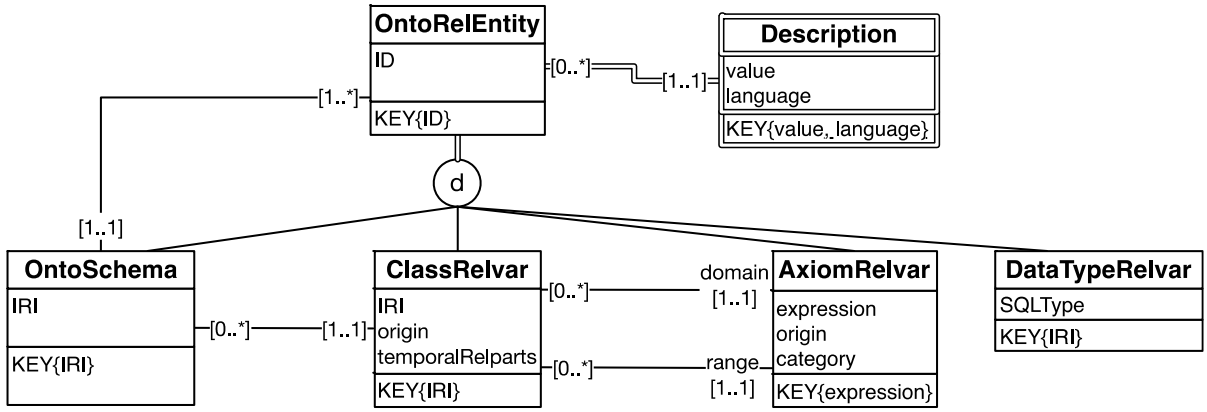


Figure 23. Diagramme EAE d'un OntoRel.

Génération OntoRelDic

Le processus de génération OntoRelDic produit une version réutilisable à l'extérieur d'OntoRelα de l'arrimage entre les composants de µOnto et µRel. OntoRelDic peut être utilisé pour plusieurs activités : arrimage des sources, génération de requêtes, etc.

Génération SQL

Le processus de génération SQL produit l'ensemble de scripts SQL suivants :

- ◊ Un script contenant la définition du schéma de base et les instructions de création de tables de base avec les contraintes de clé candidate et les contraintes de clé référentielle.
- ◊ Un script contenant les instructions de créations des fonctions qui encapsulent les contraintes générales et les contraintes temporelles.
- ◊ Un script contenant la définition du schéma IRI (identifiant unique d'un composant ontologique) et les instructions de création des vues IRI sur les tables de base. Les vues sont définies par renommage en utilisant le IRI de la classe représentée par la table.
- ◊ Un script contenant la définition du schéma par langue naturelle (LN) et les instructions de création des vues LN sur les tables de base. Les vues sont définies par renommage en utilisant les annotations de la langue spécifique.

Il est à noter que les schémas de vues sont essentiellement créés pour offrir une « interface » d'accès pour les utilisateurs et les applications externes.

3.3 Mise en œuvre

OntoRelα est mis en œuvre en Java 8 avec les bibliothèques externes suivantes : OWLAPI 5.1[31] pour charger et analyser l'ontologie au format OWL 2, JGraphT[32] pour créer le graphe de l'ontologie et le graphe relationnel, Snakeyaml[33] pour définir et charger les fichiers de configuration, Jackson[34] pour générer du JSON, StringTemplate[35] pour générer les scripts SQL, et Logback[36] pour gérer les traces du programme.

Les fichiers de configuration sont définis en YAML. L'ontologie normalisée est générée en µOnto, l'OntoRelDic en JSON, les scripts de la base de données sont générés en SQL/PLSQL, les graphes sont produits au format DOT et les avertissements sont produits au format texte.

[31] https://github.com/owlcs/owlapi
[32] https://jgrapht.org
[33] https://mvnrepository.com/artifact/org.yaml/snakeyaml
[34] https://github.com/FasterXML/jackson
[35] https://www.stringtemplate.org
[36] https://logback.qos.ch

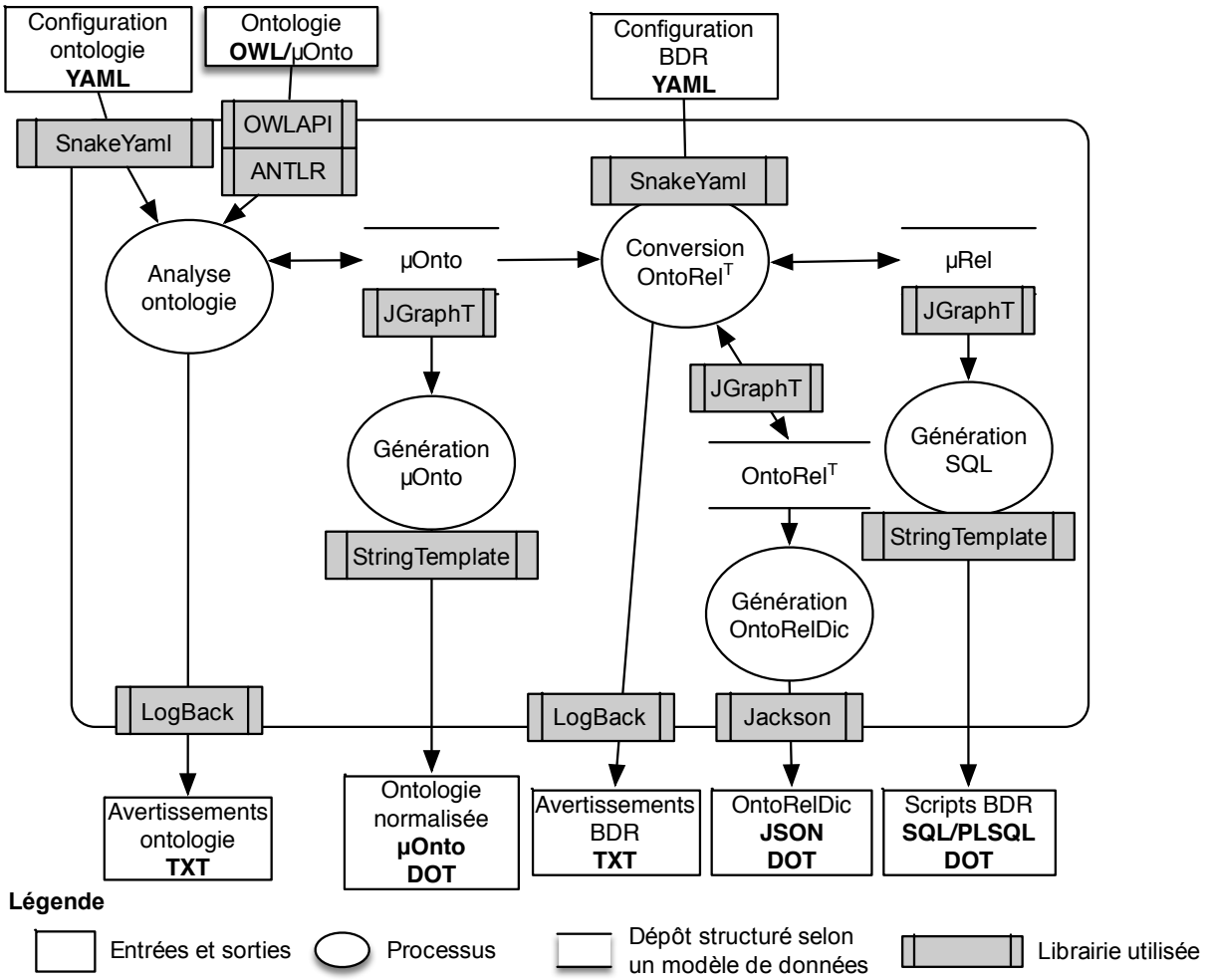


Figure 24. Diagramme de flux de données de mise en œuvre d'OntoRela

3.3.1 Traduction OWL en µOnto

Le Tableau 7 ci-dessous présente, pour chaque composant OWL, la traduction en composant µOnto. Pour plus de détails relativement au langage OWL, il est possible de se référer à [Motik et al. 2012b].

Tableau 7. Traduction des composants OWL en µOnto

µOnto	**OWL**
Class	*Class*
Individual	*Individual* *ClassAssertion* *PropertyAssertion* *DataPropertyAssertion*
Datatype	*Datatype*
Object property	*ObjectProperty*
Data property	*DataProperty*
Annotation	*AnnotationAssertion*
Class inheritance axiom	*SubClassOf* *DisjointUnion*
Property inheritance axiom	*SubObjectPropertyOf* *SubDataPropertyOf*
Class association axiom	*SubClassOf* *ObjectPropertyDomain* *ObjectPropertyRange*
Data association axiom	*SubClassOf* *DataPropertyDomain* *DataPropertyRange*
Inverse property axiom	*InverseObjectProperty*
Participation	*FunctionalObjectProperty* *FunctionalDataProperty* *ObjectSomeValuesFrom* *ObjectMinCardinality* *ObjectMaxCardinality* *ObjectExactCardinality* *DataSomeValuesFrom* *DataMinCardinality* *DataMaxCardinality* *DataExactCardinality*

La version actuelle d'OntoRelα accepte en entrée une ontologie formalisée selon OWL 2 [Motik et al. 2012b], mais ne prend pas en charge la traduction des constructions OWL suivantes :

- ◊ individus nommés ;
- ◊ axiomes d'équivalence (*EquivalentTo*), axiome de clé (*HasKey*) ; axiomes de propriété d'annotation, axiomes disjoints ;
- ◊ caractéristiques des propriétés : réflexives, anti-réflexives, symétriques, asymétriques et transitives ;
- ◊ négation et complément des opérateurs ;
- ◊ contraintes de types de données (facettes XML).

Dans OWL, les axiomes sont classés en plusieurs catégories qui peuvent être traduites en µOnto en axiomes d'héritage ou en axiomes d'association. Dans la suite, une traduction est décrite par l'axiome OWL à traduire, exprimée dans la syntaxe fonctionnelle, le symbole « => » qui

désigne une traduction et les axiomes résultants µOnto. Il est également à noter que la traduction a lieu après le processus de réduction des axiomes.

Création d'axiomes d'héritage de classe

Pour chaque axiome *SubClassOf* qui ne comporte pas une propriété, un axiome d'héritage est créé :

```
SubClassOf(C0, C1) => C0 ⊑ C1
```

Pour chaque axiome *DisjointUnion* qui ne comporte pas une propriété, un axiome d'héritage par élément de l'union et une contrainte sont créés :

```
DisjointUnion(C0, Ci, …, Cn) =>
  [A 1≤ i ≤ n : [A 1≤ j ≤ n : (i≠j) => Ci ∩ Cj = ∅] Ci ⊑ C0 ; … ; Cn ⊑
  C0
```

Création d'axiomes d'héritage de propriété

Pour chaque axiome SubObjectPropertyOf ou SubDataPropertyOf, un axiome d'héritage de propriété est créé :

```
SubObjectPropertyOf(p0, p1) => p0 ⊑ p1
SubDataPropertyOf(p0, p1) => p0 ⊑ p1
```

Création d'axiomes d'association de classe

Pour chaque axiome *SubClassOf* qui ne comporte pas de propriété de classe, un axiome d'association de classe est créé :

```
SubClassOf(C0 owlRestriction (p C1)) => C0 p qt C1
```

où *owlRestriction* est une restriction définie par un quantificateur (some) ou une cardinalité (exact, min, max). Une restriction OWL est traduite en participation (qt) selon le Tableau 8 ci-dessous.

Tableau 8. Traduction des restrictions OWL en µOnto

µOnto	**OWL**
[1..*] (some)	*ObjectSomeValuesFrom* *DataSomeValuesFrom*
[n..n] (exact n)	*ObjectExactCardinality* *DataExactCardinality*
[n..*] (min n)	*ObjectMinCardinality* *DataMinCardinality*
[0..n] (max n)	*ObjectMaxCardinality* *DataMaxCardinality*
[0..1] (functional)	*FunctionalObjectProperty* *FunctionalDataProperty*

Pour chaque paire d'axiomes *ObjectPropertyDomain* et *ObjectPropertyRange* ayant la même propriété, un axiome d'association de classe selon la caractéristique de la propriété est créé :

```
ObjectPropertyDomain(p C0); ObjectPropertyRange(p C1) =>
  Si c'est une propriété fonctionnelle : C0 p [0..1] C1
  Sinon : C0 p [0..*] C1
```

Création d'axiomes d'association de données

Pour chaque axiome *SubClassOf* qui ne comporte pas de propriété de données, un axiome d'association de données est créé :

```
SubClassOf(C0 owlRestriction(p D1)) => C0 p q D1
```

Pour chaque paire d'axiomes *DataPropertyDomain* and *DataPropertyRange* ayant la même propriété, un axiome d'association de données selon la caractéristique de la propriété est créé :

```
DataPropertyDomain(p C0) ;DataPropertyRange(p D1) =>
  Si c'est une propriété fonctionnelle : C0 p [0..1] D1
  Sinon : C0 p [0..*] D1
```

Création d'annotations

Pour chaque *AnnotationAssertion*, une annotation µOnto <E, N, L, T> est créée. La catégorisation des types d'annotations est configurable selon le IRI. Par défaut, les IRI suivants sont classés comme suit :

- Les annotations d'étiquettes :
  - « http://www.w3.org/2000/01/rdf-schema#label »
  - « http://purl.obolibrary.org/obo/BFO_0000179 »
- Les annotations de définition :
  - « http://www.w3.org/2000/01/rdf-schema#comment »
  - « http://purl.org/dc/elements/1.1/description »
  - « http://purl.obolibrary.org/obo/BFO_0000180 »
  - « http://purl.obolibrary.org/obo/IAO_0000115 »

3.3.2 Traduction µRel en SQL

Le générateur SQL traduit une instance du modèle µRel en plusieurs scripts SQL exécutables relativement à une base de données relationnelle spécifique. Le Tableau 9 présente les instructions SQL créées pour chaque composant de µRel. La version actuelle de OntoRelα ne génère pas les procédures stockées ni les automatismes (triggers).

Tableau 9. Traduction des restrictions µRel en PostgreSQL

µRel	SQL
Schema	CREATE SCHEMA
Description	COMMENT ON
Base relvar (relvar)	CREATE TABLE
Virtual relvar (view)	CREATE VIEW
Type	CREATE DOMAIN
General Constraint	CREATE FUNCTION + CREATE TRIGGER
Candidate key	PRIMARY KEY
Referential key	FOREIGN KEY
Participation Constraint	CREATE FUNCTION + CREATE TRIGGER
Inheritance referential key	FOREIGN KEY + CREATE PROCEDURE + CREATE TRIGGER

3.3.3 Traduction OntoRelDic en JSON

Le dictionnaire d'arrimage (OntoRelDic) décrit l'arrimage entre un composant ontologie et un ou plusieurs composants relationnels. L'OntoRelDic est généré à partir du modèle OntoRelT dans un format JSON.

3.4 Évaluation

Le prototype est testé avec plusieurs ontologies de taille et de complexité variées : *Prescription of Drugs Ontology* (PDRO)[37], PDRO-P3 extension de PDRO pour le système PARS3, *The Drug Ontology* (DRON)[38], *Environment Ontology* (ENVO)[39], *Univ-Bench*[40], *Mondial*[41], *Pizza*42. Les scripts des bases de données sont exécutables sur PostgreSQL v9.0+.

Le code et les résultats sont disponibles sur GitHub : https://github.com/OpenLHS/OntoRela.

De plus, l'outil a été évalué avec un cas d'étude pour la modélisation des organisations hospitalières en France (voir l'article soumis en annexe 5).

3.4.1 Résultats expérimentaux

Le Tableau 10 présente [A] une synthèse du décompte des composants ontologiques et le décompte des composants relationnels [B] le détail du décompte des composants ontologiques et [C] une mesure simple de la complexité de la structure de la base de données résultante pour

37 https://github.com/OpenLHS/PDRO

38 https://bioportal.bioontology.org/ontologies/DRON

39 https://bioportal.bioontology.org/ontologies/ENVO

40 http://swat.cse.lehigh.edu/projects/lubm/

41 https://www.dbis.informatik.uni-goettingen.de/Mondial/#RDF

42 https://protege.stanford.edu/ontologies/pizza/pizza.owl

permettre d'avoir une idée qualitative de la complexité de la gestion de l'évolution de la structure et des données. Cette mesure est définie en termes de ratio de la somme des composants relationnels par rapport aux axiomes originaux de l'ontologie `(Tables + FK + Func)/Ax-O total`. Également, dans tous les tableaux la durée d'exécution est donnée.

Voici deux observations :

- ◊ La durée d'exécution dépend fortement du nombre d'axiomes d'origines de l'ontologie. Cela est attendu, car le noyau des algorithmes OntoRelα se concentre sur l'analyse des axiomes. À noter également, plusieurs ontologies du domaine biomédical contiennent un grand nombre d'axiomes complexes (le nombre d'Ax-R est élevé), ce qui explique le nombre plus élevé d'Ax-G (Tableau 10.A) et la nécessité de la réduction de la complexité des axiomes.
- ◊ Le nombre total d'axiomes d'origine (Ax-O somme des axiomes isa, class et data) est un bon prédicteur du temps d'exécution pour les ontologies dont le nombre total d'axiomes dépasse un certain seuil. Davantage de tests sont nécessaires pour évaluer ce seuil, qui semble se situer entre 1000 et 2000.

Enfin, il faut mettre en place des expériences et des mesures supplémentaires pour pouvoir estimer l'effort requis pour l'utilisation du schéma obtenu en termes d'interrogation et de maintenance. En outre, des jeux de données et des requêtes doivent être définis pour alimenter et tester la base de données résultante.

Tableau 10. Résultats expérimentaux d'OntoRelα

[A]

Ontology	**Classes**	**DP**	**OP**	**Ax-O total**	**Ax-G total**	**Ax-R total**	**Tables**	**FK**	**Func.**	**Exec. (sec)**
VSO	33	16	38	99	99	0	105	162	71	3
MONDIAL	52	28	59	124	114	0	134	192	82	1
UNIBENCH	43	7	25	54	54	0	64	83	20	1
PIZZA	100	0	8	262	234	0	250	400	150	2
GENO	393	9	123	468	483	8	507	617	108	4
ECSO	561	2	23	893	794	61	872	1 172	288	14
CHMO	3 078	0	25	3 307	3 340	0	3 360	3 736	281	38
IFAR	4 530	0	0	4 977	4 977	0	4 531	4 984	0	56
PDRO	3 952	12	131	6 265	7 766	543	7 510	10 833	2 964	150
PDRO-P3	3 963	15	139	6 256	7 970	554	7 684	11 133	3 127	154
GAMUTS	18 001	0	0	6 249	6 249	0	18 002	18 720	0	347
OAE	5 700	3	120	10 882	14 404	764	11 629	20 328	5 889	503
ONTOLURGENCES	10 031	0	60	11 999	12 236	6	11 491	13 765	1 455	529
ENVO	8 511	0	63	13 394	14 262	335	11 026	16 499	2 211	537

[B]

Ontology	Ax-O isa	Ax-O class	Ax-O data	Ax-O total	Ax-G isa	Ax-G class	Ax-G data	Ax-G total	Exec. (sec)
VSO	28	58	13	99	28	58	13	99	3
MONDIAL	32	66	26	124	32	56	26	114	1
UNIBENCH	34	20	0	54	34	20	0	54	1
PIZZA	84	178	0	262	84	150	0	234	2
GENO	366	96	6	468	380	101	2	483	4
ECSO	470	423	0	893	518	276	0	794	14
CHMO	3 056	251	0	3 307	3 056	284	0	3 340	38
IFAR	4 977	0	0	4 977	4 977	0	0	4 977	56
PDRO	4 153	2 087	25	6 265	4 879	2 862	25	7 766	150
PDRO-P3	3 963	2 200	93	6 256	4 933	2 944	93	7 970	154
GAMUTS	6 249	0	0	6 249	6 249	0	0	6 249	347
OAE	6 294	4 588	0	10 882	9 268	5 135	1	14 404	503
ONTOLURGENCES	10 781	1 218	0	11 999	10 799	1 436	1	12 236	529
ENVO	11 569	1 825	0	13 394	12 169	2 093	0	14 262	537

[C]

Ontology	Ax-O total	Tables	FK	Func.	CPLX	Exec. (sec)
VSO	99	105	162	71	3,41	3
MONDIAL	124	134	192	82	3,29	1
UNIBENCH	54	64	83	20	3,09	1
PIZZA	262	250	400	150	3,05	2
GENO	468	507	617	108	2,63	4
ECSO	893	872	1 172	288	2,61	14
CHMO	3 307	3 360	3 736	281	2,23	38
IFAR	4 977	4 531	4 984	0	1,91	56
PDRO	6 265	7 510	10 833	2 964	3,40	150
PDRO-P3	6 256	7 684	11 133	3 127	3,51	154
GAMUTS	6 249	18 002	18 720	0	5,88	347
OAE	10 882	11 629	20 328	5 889	3,48	503
ONTOLURGENCES	11 999	11 491	13 765	1 455	2,23	529
ENVO	13 394	11 026	16 499	2 211	2,22	537

Légende :

Classes le nombre de classes dans l'ontologie d'origine
DP le nombre de propriétés de données dans l'ontologie d'origine
OP le nombre de propriétés de classes dans l'ontologie d'origine
Ax-O le nombre d'axiomes dans l'ontologie d'origine
- isa axiome d'héritage
- class axiome d'association de classe
- data axiome d'association de données
- total nombre d'axiomes total (isa+class+data)

Ax-G le nombre d'axiomes générés par OntoRelα
- isa axiome d'héritage
- class axiome d'association de classe
- data axiome d'association de données
- total nombre d'axiomes total (isa+class+data)

Ax-R le nombre d'axiomes complexes réduits (Ax-I class)
Tables le nombre de relvars
FK le nombre de contraintes référentielles
Func. le nombre de fonctions
Exec. le temps d'exécution en seconde

CPLX la mesure de complexité `(Tables + FK + Func)/Ax-O total.`

3.4.2 Modélisation des structures hospitalières

La méthode a été appliquée dans le contexte des structures hospitalières (voir l'article tel que soumis en annexe 5). Deux principales raisons justifient le choix de ce cas d'étude.

Premièrement, le manque de représentation formelle des données administratives liées aux structures organisationnelles des hôpitaux limite le partage des données au niveau international [Burgun et al. 2017]. Par exemple, les informations relatives aux services hospitaliers sont décrites au niveau des données du patient dans le registre national danois des patients [Schmidt et al. 2015], alors que ces informations sont absentes du système national français de données de santé [Tuppin et al. 2017]. Ces études évaluent notamment l'efficacité des procédures cliniques, des processus de santé et de la gestion des établissements de santé [National Academy of Medicine 2018]. Toutes ces études nécessitent des informations organisationnelles détaillées plutôt qu'une analyse au niveau de l'hôpital [Harper et al. 2001; Olsen and Street 2008] pour pouvoir comparer les résultats, les couts en pratique et les variations des structures dans les processus de soins et à l'organisation [Hearld et al. 2008; Kulkarni et al. 2009]. Une modélisation insuffisante des données structurelles organisationnelles des hôpitaux pourrait générer des biais et fournir des informations inexactes aux décideurs. Ainsi, la définition d'une représentation formelle de la structure organisationnelle permet de mieux comprendre leurs relations et assure un partage sémantiquement uniforme.

Deuxièmement, l'accès, le partage et la réutilisation des données de santé à des fins de recherche sont soumis à des règles strictes. Les stratégies d'accès aux données sont basées sur les organisations, les patients ou une combinaison des deux. De nombreux hôpitaux en Europe ont adopté des politiques d'accès aux données standard basées sur les structures organisationnelles. Par défaut, les médecins sont autorisés à accéder aux données de tous les patients admis dans leur service et à les réutiliser. Au niveau du patient, l'accès aux données par projet est adopté par un entrepôt de données situé aux États-Unis [Danciu et al. 2014] et en Europe [Jannot et al. 2017]. Au niveau de l'organisation, l'accès aux données est accordé par des structures certifiées ou par un comité d'éthique [Tuppin et al. 2017]. L'évolution des structures organisationnelles des hôpitaux ne figure pas toujours dans les bases de données régionales et nationales ni dans les entrepôts de données où cette information est en grande partie intégrée avec des méthodes *ad hoc*, parfois même manuellement. Ainsi, cette information ne peut être ni interrogée par les chercheurs ni intégrée à d'autres données. En conséquence, pour une étude rétrospective, de telles règles ne peuvent être mises en œuvre sans une compréhension approfondie des structures organisationnelles des hôpitaux et de leurs relations

dans le temps. Une base de données temporelle pourrait être utile pour mettre en évidence les modifications des structures organisationnelles des hôpitaux au fil du temps (la création, la destruction, la fusion ou la séparation d'unités).

Plus spécifiquement nous avons :

- ◊ défini une ontologie des structures hospitalières en France HORG-FR ;
- ◊ engendré automatiquement un schéma relationnel temporalisé ;
- ◊ alimenté la base de données depuis un ensemble de fichiers extraits de la base de données AP-HP Sirius [TIcSante.com 2009] ;
- ◊ interrogé la base de données pour valider les contraintes temporelles entre les structures ;
- ◊ montré l'apport de la temporalité pour les deux cas d'utilisation : (1) les variations dans les processus de soins et (2) l'application de règles d'accès aux données.

La méthode permet de traiter une classe de problèmes importants de façon beaucoup plus satisfaisante. Il faut maintenant passer de la preuve de concept à la mise à l'échelle en développant les ontologies générales appropriées, en améliorant l'automatisation de certaines étapes et en développant des outils généraux.

L'ontologie de la structure organisationnelle d'un hôpital en France

La structure organisationnelle d'un hôpital comprend deux structures élémentaires : l'unité d'hospitalisation (UH) et l'unité fonctionnelle (UF). L'UH a un champ de responsabilité administrative et l'UF a un champ de responsabilité médicale. Les autres niveaux sont des agrégats d'éléments d'unité UH ou UF selon différentes perspectives (administrative, médicale ou mixte). Trois propriétés principales ont été utilisées pour relier les unités définies : « unitOf » et « hasUnit » (inverse de « unitOf ») représentent la hiérarchie entre les unités et « linkedTo » représente un lien entre deux unités.

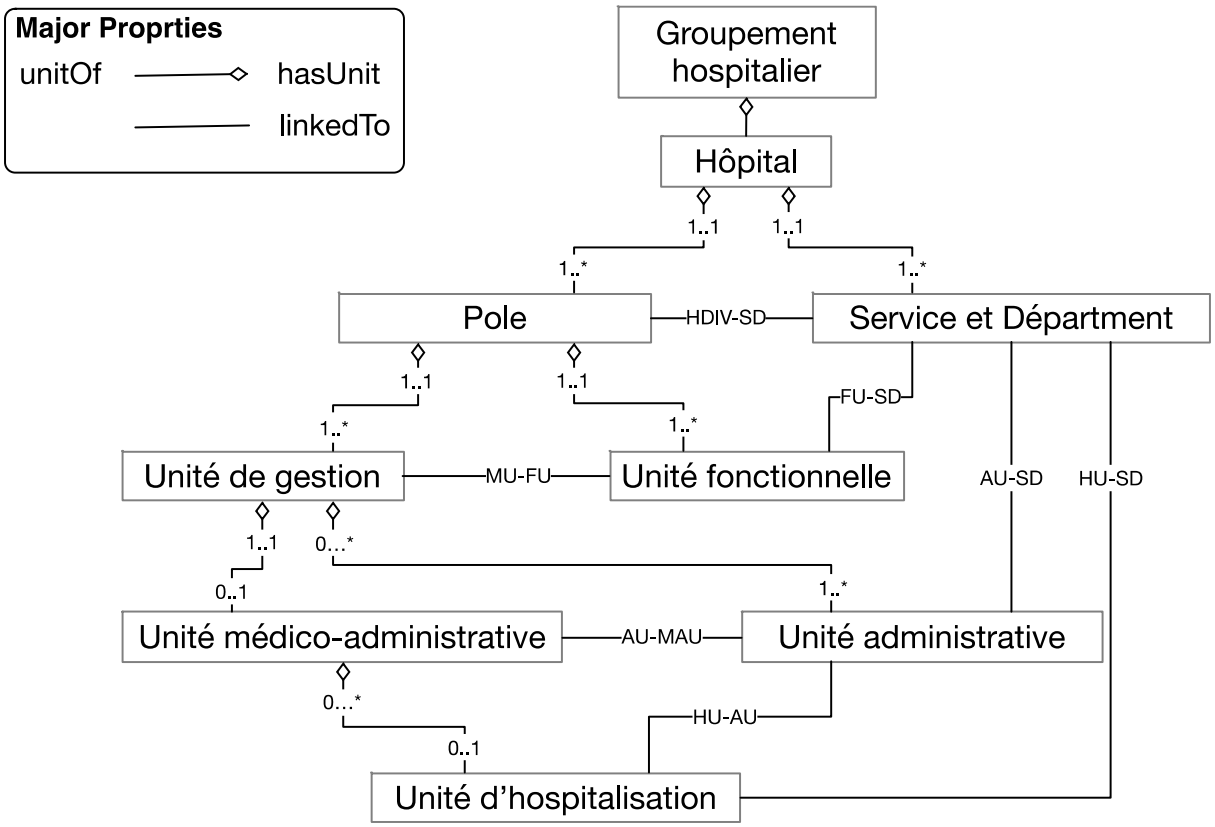


Figure 25. Illustration de la structure organisationnelle d'un hôpital

L'ontologie HORG-FR est une extension de ORG, une ontologie recommandée par le W3C [Reynolds 2014] qui représente des structures organisationnelles génériques. HORG-FR a été conçue à l'aide de Protégé et contient 288 définitions (Tableau 11). De plus, le contrôle de cohérence a été effectué avec succès avec les raisonneurs HermiT et Pellet.

Tableau 11. Nombre de composants ontologique de ORG et HORG-FR

Nombre	ORG	HORGO-FR	HORGO-FR total
Classes	16	30	46
Propriétés de classe	36	3	39
Propriétés de données	2	4	6
Axiomes	110	87	197

La base de données historicisée

La base de données historicisée contient 656 composants relationnels (Tableau 12) et 2985 unités avec 3529 relations différentes de deux hôpitaux (Hôpital européen Georges-Pompidou et Hôpital Ambroise-Paré).

Tableau 12. Nombre de composants relationnels de ORG et HORG-FR

Nombre de constructions relationnelles	
Tables	153
Contraintes	343
Vues	70
Procédures (ETL)	80
Fonctions	10
Nombre de tuples	
Unités HEGP	1887
Liens entre unités HEGP	2254
Unités *Ambroise-Paré*	1098
Liens entre unités *Ambroise-Paré*	1275

La base de données offre une structure axiomatique cohérente pour les données temporelles alignées sur les entités et les axiomes issus de l'ontologie et des règles de validité temporelle utilisant des contraintes temporelles.

Les données ont été validées à l'aide d'un ensemble de requêtes de validité définies en fonction des axiomes d'ontologies ayant les propriétés suivantes : « hasUnit », « unitOf » et « linkedTo ». Chaque requête de validation teste (1) la restriction de propriété dans le temps et (2) l'inclusion temporelle entre les unités liées. Le test d'inclusion temporelle vérifie si une période valide d'une sous-unité est incluse dans la période valide de la super-unité. À noter que ce test est réalisé à l'aide des opérateurs Allen. Les résultats montrent que la majorité des restrictions portant sur « hasUnit » et « unitOf » sont respectées. Pour la propriété « linkedTo », seules 3 requêtes de validation sur 7 valident la restriction de propriété. De plus, un nombre élevé de non-inclusions temporelles entre les unités est détecté. Cela peut s'expliquer par (1) le fait que les sources de données ne sont pas totalement temporalisées, et (2) par le manque de contraintes d'intégrité permettant de suivre la réorganisation des unités.

De plus, la base de données permet de définir des requêtes temporelles pour reconstruire des structures organisationnelles pour des périodes spécifiques. Par exemple, la Figure 26, illustre la requête qui permet de récupérer une structure des unités ayant de la spécialité « NUTRITION » durant la période [2013-01-01, 2019-01-01]. Le résultat montre la hiérarchie des structures des unités des deux hôpitaux.

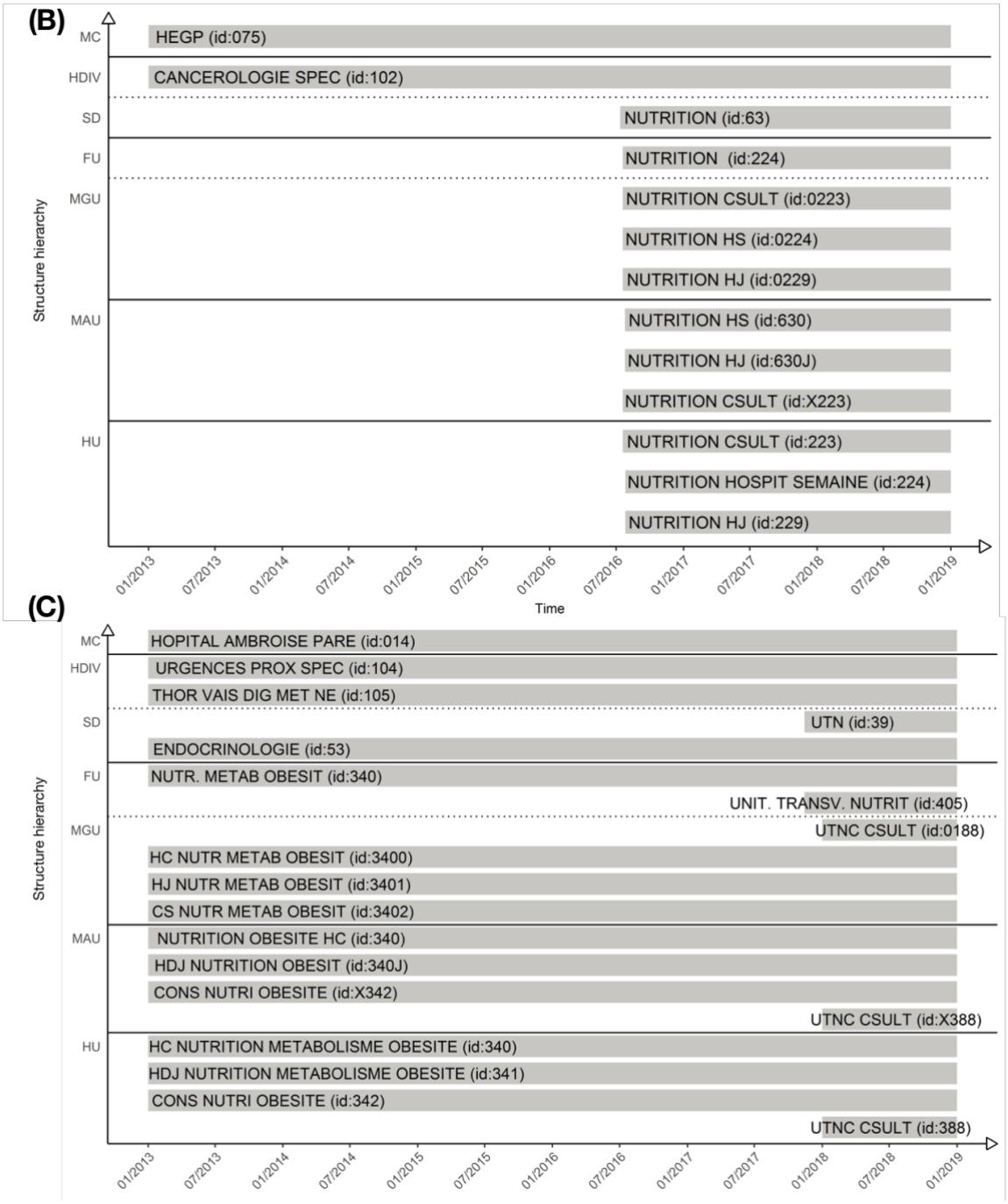


Figure 26. Illustration de requêtes de deux structures organisationnelles

4

Discussion

Cette section présente les avantages, les limites et les perspectives de la construction et de l'utilisation de l'ontologie et du modèle ontologique relationnel temporalisé.

4.1 Ontologies

Il existe un grand nombre de publications dans le domaine de l'ingénierie d'une ontologie (*ontology engineering*) qui incluent le processus de développement, le cycle de vie et des techniques de construction [Cardoso 2007]. Les principes de base de la construction d'une ontologie sont [Guarino 1997] : la réutilisation, le consensus et le partage. Cela implique l'élaboration d'une représentation des entités de la réalité avec un accord « commun » entre les experts du domaine et cela consiste à rendre accessibles l'ontologie et les mises à jour. La mise en place d'une méthodologie de construction systématique et la disponibilité d'outils adéquats sont cruciaux afin de respecter ces principes et contrôler le processus de développement [De Nicola and Missikoff 2016].

Un des avantages des ontologies au point de vue de la modélisation est la possibilité de représenter un domaine à plusieurs niveaux d'abstraction. À cet effet, selon [Guarino 1998], il existe 3 niveaux ontologiques (Figure 27) :

- ◊ Ontologie de haut niveau (*top-level ontology*) : représente des entités *générales* de la réalité indépendamment d'un domaine particulier (p. ex. BFO, DOLCE, GFO, UFO, SUMO).
- ◊ Ontologie de domaine (*domain ontology*) : représente des entités relatives à un domaine particulier (p. ex. *Ontology for general medical science*[43]*, the ontology of genes and genomes*[44], *The prescription of drugs ontology*[45], *Genetic test ontology*[46]).
- ◊ Ontologie d'application : une spécialisation d'une ontologie du domaine selon une application particulière (p. ex. *The prescription of drugs ontology* for PARS3).

[43] https://bioportal.bioontology.org/ontologies/OGMS
[44] https://bioportal.bioontology.org/ontologies/OGG
[45]https://github.com/OpenLHS/PDRO
[46] https://bioportal.bioontology.org/ontologies/GTO

Pour notre part, nous proposons d'ajouter un niveau intermédiaire entre le niveau de domaine et le niveau d'application. Ce niveau a pour but de particulariser une ontologie de domaine ou de tâche selon les pratiques d'une juridiction (pays, état, province, canton, territoire, etc.) ; p. ex. la prescription des médicaments au Québec. Ce niveau d'abstraction permet d'offrir une classification plus spécifique sans compromettre les définitions des entités d'une ontologie de domaine pour éviter la perte d'informations. Il s'agit souvent d'ajouter des contraintes qui ne s'appliquent pas dans toutes les juridictions ; p. ex. (a) inclure l'obligation d'inscrire une date de début de prise de médicament explicite versus (b) définir qu'en l'absence de date de prise de médicament, l'interprétation soit que la date de la prescription en tienne lieu.

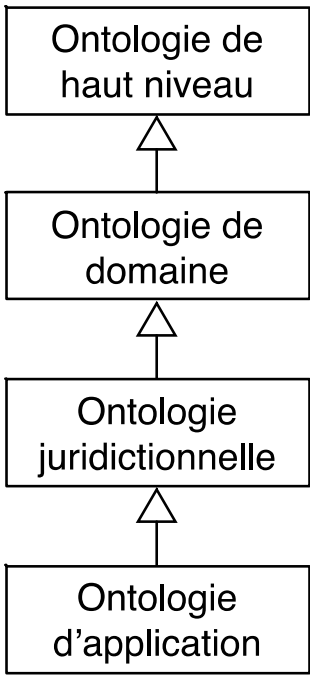


Figure 27. Niveaux ontologiques et leurs dépendances.

Nous considérons qu'une ontologie d'application est nécessaire pour avoir un modèle de données qui soit utilisable en pratique, cela attendu le nombre et l'importance des détails devant y être précisés. En outre, le maintien de la spécification de ces détails à même l'ontologie permet d'utiliser les outils ontologiques pour diverses validations (notamment les différents mécanismes d'inférence disponibles).

4.2 Modèle OntoRelT

La thèse a présenté une méthode de construction d'un modèle relationnel à partir d'une ontologie et une méthode d'historicisation de schéma relationnel. Les deux méthodes répondent à des règles bien définies s'appuyant sur le modèle OntoRelT, qui est lui-même dérivé des

modèles ontologique et relationnel. De plus, les contraintes temporelles engendrées sont uniformes et indépendantes du domaine d'application – ce qui est très avantageux au moment d'écrire les requêtes. En fin de compte, un schéma relationnel historicisé est automatiquement produit et mis en œuvre dans un SGBDR. Il reste cependant des défis qui sont présentés ci-après.

4.2.1 Conversion ontologique-relationnelle

L'utilisation des ontologies en conjonction avec un modèle de données relationnel reste un défi en raison de l'hypothèse du monde ouvert (*Open Word Assumption*) utilisée dans les ontologies par rapport à l'hypothèse du monde fermé (*Closed Word Assumption*) utilisées dans le modèle relationnel. Une formulation de cette différence est que les propositions (ou les axiomes) qui pourraient être représentées dans un modèle relationnel — mais qui ne le sont pas — sont présumées être fausses, alors que les propositions (ou les axiomes) qui ne sont pas énoncées dans une ontologie ne sont présumées ni fausses ni vraies. Bien que beaucoup ait été écrit sur la manière de générer un modèle de données à partir d'une ontologie, il n'y a pas de consensus sur les règles de conversion à adopter pour gérer ce problème. Les processus d'analyse et de conversion d'OntoRelα permettent de réduire l'écart entre le monde ouvert et le monde fermé en offrant une modélisation flexible qui permet de représenter un grand nombre de composants ontologiques. Ainsi, plusieurs règles de conversion sont proposées pour permettre la réduction de l'écart entre le monde ouvert et le monde fermé.

Règles de réduction de la complexité

En pratique, nombre d'ontologies sont exprimées à l'aide d'expressions complexes et de classes anonymes. Sans réduction de la complexité, elles ne pourraient être traitées. Par ailleurs, la réduction à des axiomes simples est primordiale pour garantir une conversion uniforme et directe à travers toute l'ontologie. Les règles de réduction de complexité permettent de définir les axiomes avec des formats canoniques qui correspondent à une règle spécifique de conversion. Cette uniformisation permet d'automatiser la conversion et l'inclusion de contraintes de validation. L'inclusion de ces contraintes accroit la pertinence du schéma relationnel produit et diminue considérablement le nombre d'erreurs de traitement qui pourraient autrement passer inaperçues. Il s'agit d'un apport important de notre méthode.

Il convient de noter que la version actuelle d'OntoRelα effectue automatiquement la réduction de la complexité.

Règles de réduction de redondance

La réduction de la redondance des axiomes est indispensable pour assurer l'intégrité des données lors de la manipulation (modification et interrogation). Une redondance d'axiomes induit la création de deux ou plusieurs structures relationnelles qui peuvent contenir des données en double. Les tuples de ces structures relationnelles seraient dès lors susceptibles d'être modifiés indépendamment et de façon incohérente, entrainant ainsi des résultats incorrects. La réduction de la redondance des axiomes est un apport important de notre méthode.

La version actuelle d'OntoRelα effectue automatiquement la réduction de redondance des axiomes issus directement de la réduction de complexité. Elle peut en outre signaler les redondances potentielles issues des axiomes originaux de l'ontologie source.

Définition d'axiomes d'association de classes

Dans les ontologies, il est usuel de ne pas formaliser les axiomes d'association facultatifs (dont la borne minimale de la participation est nulle. Dans un monde ouvert, des associations peuvent exister sans problème entre deux individus même si elles ne sont pas explicitées au niveau de l'ontologie. À noter que leur introduction au moment de l'utilisation de l'ontologie demeure toutefois possible. Par contre, dans le modèle relationnel, ces liens doivent être anticipés, prévus dans l'entête des relvars (grâce à un attribut) et le plus souvent représentés par une clé référentielle (avec ou sans une relvar d'association), sans quoi ils ne pourront être conservés au moment de l'exploitation du schéma relationnel. Pour cette raison, au moment du développement d'une ontologie applicative devant servir à la construction d'un schéma relationnel grâce à OntoRelα, des axiomes d'association avec un quantificateur « min 0 » peuvent être ajoutés dans l'ontologie. Il est aussi possible de créer des classes équivalentes qui regrouperaient les individus présentant de telles associations. En procédant ainsi, la spécification du modèle de données représenté par le schéma relationnel demeure intégralement comprise dans l'ontologie applicative. Toutefois, la création de classes équivalentes à cette seule fin conduit à un schéma relationnel plus complexe et plus encombrant. Pour obtenir la même efficience que celle permis par le quantificateur « min 0 », il faut être capable, au moment de l'analyse de l'ontologie, de déterminer quelles sont les classes équivalentes introduites pour cette seule raison – ce qui nécessite l'analyse sémantique des annotations.

La version actuelle d'OntoRelα autorise donc les participations minimales nulles qui ne sont d'ailleurs pas interdites par les principaux langages ontologiques.

Définition d'axiomes d'association de données

Un autre défi apparait avec les classes qui peuvent être associées à des instances de types (des valeurs selon un type) provenant des données des sources, mais pour lesquelles aucun axiome d'association de données n'a été déclaré. Dans un tel cas, les données associées ne peuvent pas être stockées et extraites, car aucun composant approprié n'existe dans le modèle de données. De tels axiomes sont rarement déclarés dans les ontologies de domaine puisqu'ils restreignent le type des valeurs permises et limitent l'exploration des individus potentiels qui ne sont pas prévus par l'ontologie. Bien que l'association entre les individus soit essentielle pour comprendre le domaine, la définition d'une association entre un individu et une valeur est également essentielle lorsque les ontologies sont utilisées pour prendre en charge la collecte, le stockage et la mise à jour de données. Ce faisant, la déclaration d'axiomes d'association de données est souhaitable pour guider l'extraction des données pertinentes, normaliser les valeurs, garantir l'intégrité des données et permettre la formulation de la plupart des requêtes en pratique.

Suite à ce défi, un problème apparait avec la granularité[47] des classes lors d'un arrimage avec des sources de données qui ont des granularités différentes. Les sources de données stockent des données à des niveaux de granularité variées. Dans certaines situations, il est difficile de déduire automatiquement le niveau d'appartenance d'une donnée. Il est important d'offrir la structure adéquate pour la gestion de la granularité afin d'éviter la perte de données.

La version actuelle d'OntoRelα détecte le manque d'axiomes d'association de données et avise l'utilisateur qu'une correction pourrait être bénéfique.

Dispersion des axiomes

Dans les ontologies OWL, les axiomes concernant une entité sont dispersés dans l'ontologie et peuvent être définis à plusieurs endroits. Premièrement, la restriction des associations entre individus peut être définie d'une façon générale directement au niveau de la propriété (p. ex. OWL : *ObjectPropertyDomain, ObjectPropertyRange*) ou d'une façon spécifique dans les axiomes (p. ex. *SubClassOf, EquivalentTo*).

Deuxièmement, les participations des entités pour un même axiome peuvent être définies dans les deux directions grâce à la définition des « inverses » (p. ex. OWL : *ObjectInverseOf, inverseFunctionalProperty*). Afin de récupérer les participations dans les deux directions, il est nécessaire de parcourir tous les axiomes et vérifier si la propriété possède un inverse.

[47] Degré de fragmentation d'une entité donnée en plus petites unités.

La version actuelle d'OntoRelα permet de regrouper (fusionner) diverses définitions d'axiomes en une seule définition, cela grâce à la définition plus concise de μOnto, en particulier grâce aux propriétés étendues.

Traitement des axiomes d'équivalences

La conversion des axiomes d'équivalences ne peut être identique à la conversion des axiomes d'héritage ou d'association. Les axiomes d'équivalences sont généralement définis afin de permettre l'inférence et le raisonnement.

La version actuelle d'OntoRelα ne traite pas les axiomes d'équivalence. Une étude est en cours et la mise en œuvre sera entreprise sous peu.

Modularisation des ontologies

Les ontologies de grande taille [48] introduisent des problèmes (et des défis) en termes de capacité de réutilisation, de validation, d'évolutivité et de maintenance [Stuckenschmidt et al. 2009]. Ces ontologies ont suscité un intérêt croissant pour la modularisation des ontologies (*ontology modularization*) [d'Aquin et al. 2009] ; mentionnons que la modularisation des ontologies consiste à définir des techniques de décomposition et d'extraction de modules à partir d'une ontologie [Grau et al. 2007]. Les modules d'une ontologie sont des sous-ensembles cohérents d'une ontologie pouvant être manipulés indépendamment les uns des autres et fournir le même service que l'ensemble de l'ontologie initiale. Une revue de littérature [Oh and Yeom 2012] recense et évalue les différentes méthodes de modularisation. À partir d'une certaine taille (nombre total de composants), il est indispensable de modulariser le modèle de données pour permettre d'extraire des modules « gérables » et de garantir le lien entre les modèles de données générées par ces modules en définissant les composants de « liaison » adéquats.

La version actuelle d'OntoRelα ne propose pas de mécanisme de modularisation. Une intégration de techniques de modularisation adéquate pour la génération en modèle de données est une exigence pour un déploiement à grande échelle.

Normalisation du schéma en 6FN

La normalisation du schéma en 6FN favorise l'automatisation de la construction et simplifie l'évolution ainsi que la gestion de l'intégrité des données [Rönnbäck et al. 2010]. Le schéma obtenu est très simple, chacune des relvars est composée d'un attribut-clé (relvar de classe) ou de deux attributs clés (relvar d'association) avec des contraintes référentielles ou d'un attribut-clé et d'un attribut pour une valeur avec une contrainte référentielle (relvar de données). Une

[48] *Gene Ontology* une des ontologies les plus utilisées contient en date du 25 juillet 2019 : 49 933 classes avec 107 839 axiomes.

relvar en 6FN représente un et un seul prédicat, ce qui renforce la sémantique et diminue considérablement les ambigüités.

Avec cette représentation :

- ◊ les modifications (ajout, retrait et mise à jour) sont bien contrôlées et l'impact sur l'ensemble du modèle[49] est minimisé [Rönnbäck et al. 2010] ;
- ◊ dans le contexte d'une implémentation physique, la performance peut être améliorée grâce à l'élimination des tables et à la création d'index appropriés et au stockage dans des SGBDR en colonnes [Abadi et al. 2009; Lamb et al. 2012] ;
- ◊ les requêtes peuvent être plus ciblées avec une sémantique et une syntaxe vérifiables ;
- ◊ le contrôle d'accès par table peut être mis en place par SGBDR très facilement.

Finalement, le 6 FN est indispensable pour l'historicisation [Date et al. 2014] pour permettre de suivre et contrôler l'évolution de chacun des attributs d'une relvar indépendamment. La 6FN est aussi préalable au traitement uniforme et systématique des données absentes.

La version actuelle d'OntoRelα propose une conversion selon les règles générales de conversion de l'ontologie garantissant un schéma en 6FN ou une conversion selon les règles spécifiques de conversion garantissant un schéma en 5FN.

Élaboration de règles de création de vues ontologiques

En raison de la normalisation du schéma en 6FN, le nombre de tables générées est élevé. Toutefois, des vues relationnelles peuvent être générées (p. ex. en joignant toutes les relvars associées à une classe dans une seule vue 5FN) pour fournir une structure de niveau supérieur, des vues dimensionnelles pour les requêtes analytiques, des vues en triplets pour les raisonneurs d'ontologies, etc. De plus, à travers ces vues, des procédures de modification peuvent être générées pour fournir une interface afin d'uniformiser la manipulation des données à travers des applications.

La version actuelle d'OntoRelα génère uniquement des vues de renommage regroupées en schémas linguistiques (à raison d'un schéma par langue retenue parmi les langues pour lesquelles les étiquettes sont définies dans l'ontologie). Dans chaque schéma linguistique, une vue est définie pour chaque relvar de l'OntoRel. Des travaux sont d'ailleurs en cours afin d'explorer les autres besoins.

[49] L'ajout ou le retrait d'entités ontologiques impliquera l'ajout de composants relationnels (relvar et contraintes), mais n'impliquera pas la modification des composants existants. Par contre, la mise à jour d'une entité ontologique peut impliquer une modification des composants relationnels correspondante, mais cela va impacter un ensemble de composants restreints.

Élaboration de règles de modifications

Les modifications consistent à des opérations d'insertion, de suppression et de mise à jour des tuples. Pour respecter la sémantique, deux « modes » de règles de modification peuvent être définies :

- ◊ Mode ouvert : modifications en cascades qui consistent à propager les modifications entre les entités.
- ◊ Mode fermé : modifications strictement ordonnées qui consistent à définir les tuples dans un ordre prescrit.

La version actuelle d'OntoRelα ne définit pas ces règles, mais c'est une des extensions prioritaires pour faciliter l'utilisation dans un contexte d'entrepôt de données. De plus, il convient de noter que ces règles se complexifient si le modèle est historicisé.

4.2.2 Historicisation du modèle

La méthode décrit des règles de construction d'un modèle relationnel temporalisé à partir d'un modèle relationnel non temporalisé. Le modèle représente trois types de temps de validité pour gérer l'indétermination future, la détermination complète et l'indétermination passée. De plus, des contraintes d'intégrité temporelles sont définies uniformément pour toute relation temporelle afin de garantir une évolution cohérente des données. En ce sens, les défis de construction et d'utilisation du modèle sont présentés ci-dessous.

Historicisation au niveau relationnel

L'historicisation doit être fondée sur un modèle temporel définissant des structures et des contraintes uniformes. Dans les ontologies, il n'existe actuellement aucune normalisation de l'historicisation des classes et des propriétés ni un modèle temporel consensuel comme dans la communauté de bases de données [Dyreson et al. 1994] et particulièrement pour les bases de données relationnelles. De nombreuses techniques ont été proposées par la communauté OWL (réification, *4D fluents* et *versioning*, par exemple), mais leur définition est complexe, elles offrent des capacités de raisonnement limitées et leur mise en œuvre et leur maintenance peuvent être couteuses (au moins une classe supplémentaire doit être ajoutée à chaque relation temporelle) [Batsakis et al. 2017]. Par conséquent, l'utilisation d'un modèle relationnel temporel pour stocker l'évolution des données au fil du temps est en réalité un système plus adéquat, car susceptible d'être mis en oeuvre dès à présent à échelle réaliste. Le modèle temporel utilisé est fondé sur la théorie relationnelle ce qui permet de profiter de la maturité des implémentations existantes pour une utilisation directe en pratique. La flexibilité du modèle

relationnel peut être facilement étendue pour ajouter un autre référentiel, tel que le temps de transaction, temps d'assertion, cela sans impacter la structure actuelle [Khnaisser et al. 2017b].

Historicisation par agent

La documentation et le maintien de la provenance des données et de leur évolution sont indispensables à l'élaboration de politiques de gouvernance et de contrôle de qualité. Dans plusieurs systèmes d'information, la connaissance des membres de l'organisation est ramené à un seul point de vue, celui de l'organisation. Les membres sont alors occultés au profit d'un agent unique, implicite.

Dans certains domaines comme la médecine, il est important de reconnaitre les différences d'opinion entre les agents — par exemple, un médecin peut poser un diagnostic différent d'un autre médecin. Il est également important de reconnaitre que la base de données n'exprime pas toujours les connaissances actuelles de tous les agents : certains agents peuvent avoir certaines connaissances qu'ils n'entreront que plus tard dans la base de données. Ceci peut parfois expliquer la différence de diagnostic entre deux médecins par exemple. Un modèle de données doit donc permettre la traçabilité des données par agent. Pour assouplir cette hypothèse, nous pouvons considérer que plusieurs agents peuvent avoir plusieurs avis sur une proposition donnés (p. ex. le diagnostic d'un patient). L'ajout de l'agent au temps de validité permet d'indiquer le moment où un agent donné considère qu'une proposition donnée est vraie.

Appliquons notre exemple à un scénario sur les structures organisationnelles. Par exemple, le Tableau 13 présente une partie de la structure hiérarchique du service de chirurgie cardiovasculaire et du service d'anesthésie et de réanimation qui est valide pendant la période [2013-01-01.. 2019-01-01]. Le tableau [A] présente une modélisation sans agent et le tableau [B] présente une modélisation avec agent (SIRIUS la source des données et HORG la base de données historicisée produite à l'HEGP). Concrètement, à l'HEGP l'unité HU 339 a été remplacée par l'unité HU 681 le 2018-08-30, mais cette modification ne se trouve pas dans les données sources de la base de données SIRIUS. Avec une modélisation avec agent, cette information peut être représentée adéquatement en indiquant le « point de vue » de chaque agent.

Tableau 13. Exemple d'historicisation avec et sans agent

[A]

level	unitType	id	supId	label	validTime
0	MC	75	-	HEGP	[2001-01-01,)
1	HDIV	103	75	CVRM	[2012-01-01,)
1	HDIV	105	75	ANEST REA TRAUMATO	[2012-01-01,)
2	MGU	613	105	REANIMATION CHIR	[2007-11-28,)
2	MGU	332	103	Ne plus util. REACCV	[2007-11-28,)
3	MAU	556	613	REA CHIRURGICALE	[2007-01-01,)
3	MAU	386	332	REA CHIR CARDIOVASC	[2005-01-01,)
4	HU	339	386	REA CHIR CARDIO VASCULAIRE	[2005-01-01,)
4	HU	681	556	REA CHIR POLY	[2018-09-01,)

[B]

level	unitType	id	supId	label	validTime	agent
0	MC	75	-	HEGP	[2001-01-01,)	SIRIUS
1	HDIV	103	75	CVRM	[2012-01-01,)	SIRIUS
1	HDIV	105	75	ANEST REA TRAUMATO	[2012-01-01,)	SIRIUS
2	MGU	613	105	REANIMATION CHIR	[2007-11-28,)	SIRIUS
2	MGU	332	103	Ne plus util. REACCV	[2007-11-28,)	SIRIUS
3	MAU	556	613	REA CHIRURGICALE	[2007-01-01,)	SIRIUS
3	MAU	386	332	REA CHIR CARDIOVASC	[2005-01-01,)	SIRIUS
4	HU	339	386	REA CHIR CARDIO VASCULAIRE	[2005-01-01,)	SIRIUS
4	HU	339	386	REA CHIR CARDIO VASCULAIRE	[2005-01-01,2018-08-30)	HORG
4	HU	681	556	REA CHIR POLY	[2018-09-01,)	SIRIUS

Pour une traçabilité plus complète des changements des données, une historicisation en fonction de l'agent et du temps doit être intégrée à UBHF. Une première étude sur la définition sémantique de la temporalité avec agent a été présentée dans [Barton et al. 2017].

Mise en œuvre d'un entrepôt de données historicisé

Le modèle OntoRelT n'est pas destiné à une utilisation spécifique. Le modèle peut être utilisé pour la création d'entrepôts de données physiques (par alimentation) ou un entrepôt de données virtuelles (par médiation), ou tout simplement pour une application en particulier.

Dans un contexte d'entrepôt virtuel, l'OntoRelT peut être utilisé comme modèle commun pour les arrimages (*mapping*) avec les sources de données, l'interrogation ou la construction d'applications basées sur le même modèle. Il est préférable que les sources de données relationnelles soient temporalisées pour permettre d'extraire une histoire complète des données, même si le journal du SGBD, lorsqu'il a été conservé sur une longue période avec suffisamment

de détails, peut fournir une base en regard du temps de transaction. Plusieurs travaux, notamment ceux de l'équipe d'ODBA [Borgwardt et al. 2015; Calvanese et al. 2017; Xiao et al. 2018], présentent des solutions théoriques intéressantes.

Actuellement, le modèle généré par OntoRelα est utilisé comme modèle commun dans PARS3.

Dans un contexte d'entrepôt physique, le modèle généré peut utiliser des fonctionnalités temporelles existantes sans développement supplémentaire. Depuis l'ajout de l'extension temporelle dans SQL:2011 [Kulkarni and Michels 2012] plusieurs SGBDR ont introduit des concepts temporels [Böhlen et al. 2018] : Teradata v15 (2014), DB2 v10 (≈2012), Oracle v12c (≈2013), MSSQL v2016, PostgreSQL v9.2 (2012), MariaDB v10.3.4 (2018).

Il est à noter que la version actuelle d'OntoRelα génère un schéma temporalisé qui est compatible avec PostgreSQL.

Finalement, d'autres modèles peuvent en être dérivés (semi-automatiquement) pour faciliter certaines tâches analytiques comme le modèle dimensionnel, le modèle en cube et le modèle en graphe.

4.3 Synthèse

Construction d'une ontologie

La construction d'une ontologie est une tâche complexe et nécessite une expertise multidisciplinaire [Blaisure and Ceusters 2018]. Le manque de méthodologie et d'outils pour contrôler le déroulement du processus de développement diminue considérablement la qualité des ontologies [De Nicola and Missikoff 2016] pour les raisons suivantes :

- ◊ la réutilisation introduit un grand nombre de dépendances entre les ontologies et ces dépendances sont difficiles à maintenir ;
- ◊ l'évolution est complexe à cause de l'absence d'outils appropriés de gestion de versions ;
- ◊ l'estimation du temps pour la construction ou la mise à jour est difficile à définir, faute de modèle et de métriques adéquates.

Cependant, la construction des ontologies multiniveaux favorise un alignement « naturel » entre les ontologies. Cela implique une meilleure interopérabilité sémantique entre les systèmes utilisant une ou des ontologies de la même hiérarchie.

Selon notre expérience, nous supposons que la méthodologie de construction d'une ontologie peut avoir des implications sur la qualité du schéma relationnel généré par OntoRelα. L'adoption d'une méthodologie calquée sur celle des processus de développement logiciel

permettra de profiter de la maturité de ces processus [John et al. 2018]. En plus d'un processus de développement, il est nécessaire de mettre en place des conventions de nommage, de documentation et de collaboration. Un tel travail a été entamé par la communauté des ontologies biomédicales *The Open Biomedical Ontologies Foundry (OBO)*[50]. Cependant, ces conventions demeurent inachevées et l'absence d'outils de vérification ne permet pas de garantir le respect des règles à grande échelle. Des projets sont en cours à cet effet.

Construction du modèle OntoRelT

La construction du modèle OntoRelT est semi-automatisée (configurable) et indépendante de l'ontologie de base utilisée. L'OntoRelT est un modèle de données dans lequel la sémantique est explicitement définie (et retrouvable). Autrement dit, un modèle de données dérivé uniquement des connaissances offre une axiomatisation explicite de la structure et des contraintes de la base de données (ou entrepôt de données). L'utilisation de l'ontologie permet une première validation structurelle. Grâce au mécanisme de raisonnement, la cohérence de de l'ontologie peut être validée préalablement, du moins partiellement. De plus, le processus d'analyse des axiomes d'OntoRelα permet de réduire et de détecter des redondances potentielles dans les axiomes qui ne sont pas vérifiables par les outils actuels. L'utilisation d'un modèle relationnel permet la création d'une structure uniforme et des contraintes d'intégrité. Les contraintes de clés (clés candidates et clés référentielles) sont générées explicitement et les contraintes générales (contraintes de participations, d'unions et d'intersection) et les contraintes temporelles générales (contraintes de non-redondance, de non-contradiction, de non-circonlocution et de compacité) sont générées sous forme de fonction. Ces contraintes peuvent être utilisées, selon les exigences de l'environnement, pour valider la qualité des données ou être imposées en passant par des déclencheurs (*triggers*). Par conséquent, la formulation et le résultat des requêtes sont plus systématiques et fiables.

L'historicisation est fondée sur la théorie relationnelle et l'algèbre des intervalles d'Allen. Le cadre UBHF qui est défini dans nos travaux antérieurs est utilisé pour générer automatiquement un schéma historicisé avec une sémantique temporelle unifiée qui favorise l'intégrité des données et l'expressivité des requêtes. Plus spécifiquement, avec UBHF (1) les relvars et les attributs sont catégorisés selon plusieurs référentiels temporels au sein d'une sémantique unique, (2) la décomposition est définie à l'aide des opérateurs de l'algèbre relationnelle, et (3) la structure temporelle et les contraintes générales indépendantes du domaine sont générées automatiquement. De plus, le cadre proposé ne nécessite aucune extension à la théorie

[50] http://www.obofoundry.org/principles/fp-000-summary.html

relationnelle, ce qui permet une implémentation dans de nombreux SGBDR qui sont actuellement disponibles. Par contre, malgré la maturité des SGBDR, de nombreux défis restent ouverts pour la modélisation de plusieurs référentiels temporels et l'implémentation optimale de différents opérateurs temporels [Böhlen et al. 2018]. Ces défis sont principalement dus à l'absence du type (primitif) intervalle et à l'absence de la gestion de transactions et requêtes temporelles.

4.4 Travaux futurs

Concernant la conversion ontologique-relationnelle, des extensions sont en cours de développement ou sont prévues, cela dans l'optique de pousser plus loin la sémantique du modèle OntoRel :

- ◊ prise en compte des opérateurs ONLY et NOT dans la normalisation des axiomes et la conversion ;
- ◊ utilisation des règles de modularisation de l'ontologie pour produire des schémas relationnels plus faciles à manipuler ;
- ◊ élaboration des règles de gestion de la granularité ;
- ◊ définition des règles de génération de vues ontologiques ;
- ◊ définition des règles de génération de procédures de modifications selon plusieurs modes ;
- ◊ définition du processus de réversibilité des données (tuple vers RDF) pour bénéficier des raisonneurs existants.

De plus, vu le grand nombre d'ontologies biomédicales définies selon les principes OBO, l'élaboration de règles spécifiques de conversion à BFO serait intéressante pour optimiser la structure et les contraintes d'intégrité des modèles de données en santé.

Concernant l'historicisation, plusieurs extensions sont en cours de développement ou sont prévues, dans l'optique de pousser plus loin l'expressivité temporelle et l'implémentation du modèle :

- ◊ la représentation du temps d'assertion d'un agent [Barton et al. 2017], incluant la structuration et les contraintes ;
- ◊ la représentation de l'incertitude [O'Connor et al. 2000; Anselma et al. 2015] ;
- ◊ la définition des règles d'historicisation selon la sémantique BFO, distinction entre la modélisation des continuants et les occurrents ;

◊ la définition des règles de génération des procédures de modifications temporelles en plusieurs modes ;

◊ la génération d'un modèle temporel compatible avec plusieurs SGBDR.

Finalement, la gestion du grand nombre de composants d'un OntoRelT nécessitera probablement un investissement dans la création d'algorithmes de modularisation, des vues ontologiques et des outils avancés de manipulation et d'interrogation. La définition d'un langage ontologique comprenant le concept de module, convertible en SQL, est envisagée. De plus, quelques défis restent à résoudre pour la gestion de l'évolution des composants d'un OntoRelT par rapport à l'évolution de l'ontologie.

4.5 Contributions

Dans le cadre cette thèse, nous avons élaboré une méthode de construction d'un modèle de données historicisé combinant un modèle ontologique et un modèle relationnel temporel. Il en découle quatre contributions principales.

Premièrement, la méthode inclut la définition des modèles, des règles de conversion d'axiomes et des contraintes d'intégrité. La méthode de conversion se distingue des autres méthodes proposées dans la littérature (voir Tableau 14 en annexe 2) par plusieurs points :

◊ l'analyse avancée des axiomes maximise le nombre de composants ontologiques convertit en composant relationnel tout en préservant la sémantique ;

◊ la normalisation du schéma relationnel en 6FN permet l'automatisation de la génération, le contrôle de l'intégrité des données lors des modifications ou de l'interrogation ;

◊ la génération de contraintes générales avancées, telles que les contraintes de participation, d'intersection et d'union, assure une meilleur intégrité des données ;

◊ la génération d'un dictionnaire de données permet la documentation du schéma, la réversibilité entre la structure ontologique et la structure relationnelle, la génération de requêtes sémantiques, etc. ;

◊ l'historicisation du schéma ontologique relationnel selon UBHF permet la traçabilité de l'évolution des données indispensables pour diverses études ;

◊ la présentation d'un prototype open source OntoRelα prêt à être utilisé avec diverses ontologies OWL et une base de données PostgreSQL.

Deuxièmement, la méthode d'historicisation, dont la définition a été amorcée par des travaux préalables a été étendue afin d'inclure des contraintes temporelles plus spécifiques dérivant des ontologies et modélisant la représentation du passé indéterminé, incluant la structure et des contraintes temporelles. L'historicisation proposée se distingue des méthodes existantes par plusieurs points :

- ◊ l'ajout du passé indéterminé et l'indétermination totale incluant la structure @Vxe et @Vxx compatible avec la structure existante (@Vbx, @Vbe) pour uniformiser la représentation et la sémantique temporelle ;
- ◊ l'ajout de l'opérateur *gHistory* pour la construction de l'histoire d'une relvar selon sa catégorie temporelle ;
- ◊ le traitement systématique des contraintes d'intégrité temporelle indépendamment du domaine : non-redondance, non-contradiction, non-circonlocution et la compacité avec une version spécifique adaptée aux cas d'un héritage ;
- ◊ l'automatisation du processus à partir d'un schéma généré par l'ontologie avec des règles de conversion simples et systématiques.

Troisièmement, le modèle de données résultant de l'application des méthodes proposées présente des qualités propices à son utilisation dans des contextes variés (alimentation, médiation, extraction, requêtes relationnelles, reconversion ontologique, etc.) :

- ◊ **uniforme** au niveau sémantique, structurel et technologique, ce qui facilite le maintien d'une documentation axiomatique, un accès simple et efficient aux données ainsi qu'un potentiel d'évolution supérieur aux autres méthodes ;
- ◊ **temporalisé** d'une façon uniforme et systématique, ce qui facilite le maintien de l'intégrité temporelle des données ;
- ◊ **dépendant** uniquement du modèle relationnel, ce qui en permet la mise en oeuvre à l'aide de technologies diverses et accessibles.

Quatrièmement, un prototype, OntoRelα, a été mis à disposition en sources libres :

- ◊ il permet de mettre à l'épreuve les principales propositions depuis une vaste gamme d'ontologies décrites grâce au langage OWL vers des schémas PostgreSQL.

4.6 Perspectives

Dans le cadre de cette thèse, les ontologies jouent un rôle principal dans la définition de la sémantique d'un modèle de données et la théorie relationnelle offre une base pour une historicisation uniforme et intègre. En plus de diverses extensions propres à la méthode et au prototype, plusieurs autres projets en découlent qui permettront de faciliter la construction de plusieurs OntoRel, d'étendre l'expressivité des langages d'interrogation des données et de simplifier la construction d'applications d'aide à la décision.

Mise en oeuvre d'un atelier de construction interactive de bases de données historicisées à partir d'ontologies

Le développement d'un atelier de construction interactive de bases de données historicisées à partir d'ontologies pourrait être un projet intéressant. Dans un premier temps, l'atelier permettra de choisir des ontologies d'intérêt et générer divers modèles de données historicisés (ou non) selon des SGBD cibles. Une fois le modèle généré, l'atelier permettra de choisir les sources de données à intégrer. La création de modèles de données historicisés à partir d'une ontologie permettra une meilleure annotation sémantique des données et favorisera l'automatisation des méthodes d'évaluation de la qualité [Looten et al. 2018]. L'atelier simplifiera les différentes étapes de construction : la sélection des ontologies et les entités d'intérêt, l'analyse des composants ontologiques, la génération du modèle de données selon un type de SGBD cible, l'historicisation et la génération de code pour SGBDR cible, la sélection des sources, la mise en correspondance des données et l'extraction des données.

Construction de requêtes sémantiques

La construction de requêtes à partir d'une ontologie favorise l'extraction sémantique. Les données et leurs sémantiques peuvent être retournées à l'utilisateur (à l'application). De plus, à l'aide des mécanismes de raisonnement, de nouvelles connaissances peuvent être inférées.

Également, l'intégration de la temporalisation dans l'ontologie (structure et données) en utilisant des règles sémantiques SWRL [Anagnostopoulos et al. 2013] pourrait être intéressante pour le raisonnement temporel unidimensionnel pour des extractions spécifiques. De plus, des règles de transformations des données relationnelles RDF, ainsi qu'un langage de requête relationnel-ontologique incluant un traducteur automatisé vers SQL pourraient être développés en ce sens.

Construction des bases de données de graphe

Les bases de données relationnelles et les bases de données de graphes ont des forces et des faiblesses selon le type de données à stocker et selon le type de requêtes à effectuer [Vyawahare et al. 2018]. Notamment pour les données *omics*, une récente étude démontre une meilleure performance avec des bases de données de graphe [Mattioli and Gubitoso 2018].

La création d'un modèle de données hybride pourrait être une solution intéressante. La structure de l'OntoRel normalisé en 6FN permettra une transformation automatisée d'un sous-ensemble de composants ontologiques en utilisant le langage de définition de schéma de graphe proposé par [Bonifati et al. 2019]. Avoir un schéma et des données dans une base de données de graphe permettrait de mettre en oeuvre plus simplement plusieurs des algorithmes avancés d'analyse.

Intégration du traitement du langage naturel

Divers travaux dans le domaine du traitement du langage naturel (NLP) utilisent les ontologies pour améliorer la qualité de l'annotation de données non structurées [Nebot and Llavori 2014] ou de l'extraction sémantique [Martinez-Rodriguez et al. 2018; Wimalasuriya and Dou 2010].

La structure de l'OntoRel pourrait servir de base à la structuration des données non structurées annotées à l'aide de l'ontologie utilisée pour construire l'OntoRel. Dans ce cas, l'intégration des processus NLP dans les processus de récupération des données (alimentation, médiation ou extraction) devient beaucoup plus facile et fiable.

Construction et comparaison de trajectoires de soins

Les trajectoires de santé d'un patient englobent un grand nombre de concepts [Pinaire et al. 2017] et la temporalité en est un des concepts centraux. Le modèle ontologique semble être plus utile que les seules exigences initiales pour déchiffrer la structure de la source et isoler les éléments de données qui sont intéressants à extraire. De plus, la construction de trajectoire transversale nécessite un partage de données temporalisées entre divers établissements du réseau de santé afin d'avoir une vue de « l'histoire » de santé du patient. Le modèle d'historicisation utilisé par OntoRelT pourrait vraisemblablement soutenir un modèle commun pour les trajectoires de soins telles que définies dans [Defossez et al. 2014; Vanasse et al. 2018; Ledieu et al. 2018].

Conclusion

Un grand nombre de solutions indépendantes émergent fréquemment pour répondre aux problématiques de modélisation de données temporelles en santé. Un modèle de données est interopérable s'il permet d'exprimer les modèles susceptibles d'être utilisés par les sources d'une façon à garantir l'intégrité, à préserver la sémantique et à faciliter l'accessibilité aux données selon une structure et une sémantique uniformes et selon les règles de gouvernance mises en place. La méthode de construction du modèle ontologique relationnel temporalisé présentée dans notre thèse répond à un grand nombre des exigences formulées : (1) avoir une sémantique unique, une structure uniforme et une traçabilité temporelle intrinsèque ; (2) contrôler l'intégrité des données ; (3) automatiser le processus de construction et (4) utiliser des technologies accessibles à une majorité d'établissements. Soulignons que les deux caractéristiques les plus importantes dans notre contexte sont : (1) l'adéquation du modèle de données à l'ontologie dont il est issu est garantie par sa structure et ses contraintes et non par l'utilisation qu'en font les applications externes, et (2) le maintien de l'intégrité des données en regard de l'ontologie qui est garantie par des contraintes dérivées de l'ontologie elle-même, aussi précises que possible et appliquées automatiquement et systématiquement. Le fait que les caractéristiques (1) et (2) soient maintenues au fil des transactions découle de l'architecture même des SGBDR. De plus, le fait que la structure du modèle de données soit normalisée en 6FN permet d'assurer un contrôle précis.

Ainsi, il apparait que l'historicisation est une tâche complexe qui requiert beaucoup d'efforts. Les modèles temporels contemporains manquent de généralité et requièrent un travail considérable pour les rendre extensibles et réutilisables indépendamment du domaine d'application. Par conséquent, les règles d'historicisation sont le plus souvent basées sur des règles de pratique variables, floues, incomplètes et non validées. Pour assurer une meilleure intégrité, une expressivité suffisante et une grande interopérabilité entre différentes sources de données, l'historicisation doit être basée sur un modèle temporel s'appuyant sur un cadre théorique rigoureux, des règles de structuration et de contraintes uniformes qui sont utilisables indépendamment du domaine d'application.

La méthode que nous proposons permet d'automatiser certaines décisions fondées sur des critères fondamentaux (théorie relationnelle), des critères de conception (normalisation, bitemporalité, etc.), de critères technologiques (choix du SGBDR d'hébergement, des primitives de temporalisation, du dialecte SQL, etc.) ; la mise en correspondance traçable entre

le modèle relationnel et le modèle ontologique. L'approche semi-automatisée rendue possible par le prototype est innovante dans le sens où, jusqu'à présent, les modèles de données du domaine de la santé ont été développés manuellement selon des règles ad hoc. Elle l'est également par la prise en compte d'un modèle temporel unifié et par l'intégration des contraintes du domaine provenant du modèle de connaissances.

Finalement, le maintien de la compatibilité optimale d'un OntoRelT au gré de l'évolution des sources, des connaissances et des besoins n'a pas été traité dans la thèse. En pratique, ce problème doit être résolu. Heureusement, des solutions sont d'ores et déjà en cours d'élaboration et seront rendues disponibles sous peu.

Annexes

Annexe 1
Grammaires µOnto

Les grammaires de µOnto sont décrites à l'aide du formalisme défini par ANTLR v4.

Grammaire lexicale

```
/*
-- ================================================================== A
Produit : OntoRelA
Segment : MicroOnto
Composant : MOnto_LEX.g4
Encodage : UTF8 sans BOM, fin de ligne Unix (LF)
Responsable : Christina.Khnaisser@USherbrooke.ca
Description : Mots réservés et conventions lexicales spécifiques
  MicroOnto (v1).
Statut : V1

*Copyright* 2019 GRIIS (http://griis.ca/)
GRIIS (Groupe de recherche interdisciplinaire en informatique de la
  santé)
Faculté des sciences et Faculté de médecine et sciences de la santé
Université de Sherbrooke
Sherbrooke(Québec)  J1K 2R1  CANADA
[CC-BY-NC-3.0 (http://creativecommons.org/licenses/by-nc/3.0)]

*Références*
  S.O.

@version 1.0.0
@author [CK] Christina.Khnaisser@USherbrooke.ca
@author [LL] Luc.Lavoie@USherbrooke.ca
-- ================================================================== A
*/

lexer grammar MOnto_LEX;
import LEX;

ONTOLOGY :            O N T O L O G I E     | O N T O L O G Y ;
CLASS    : 'κ' | 'κ' | C L A S S E           | C L A S S ;
EQUIV    : 'ι' | 'ι' | E Q U I V A L E N C E;
PROPERTY : 'ρ' | 'ρ' | P R O P R I E T E     | P R O P E R T Y ;
INDIV    : 'α' | 'α' | I N D V I D U         | I N D V I D U A L ;
TYPE     : 'τ' | 'τ' | T Y P E;
PREFIX   : 'π' | 'π' | P R E F I X E         | P R E F I X ;

/*====================== Les annotations ====================== */
LABEL    : '#'       | E T I Q U E T T E     | L A B E L ;
DESC     : '!'       | D E S C R I P T I O N ;

/*====================== Les opérateurs logiques ============= */
AND      : '∧' | '∩' |  E T   |  A N D |  I N T E R S E C T I O N ;
OR       : '∨' | '∪' |  O U   |  O R   |  U N I O N ;
NOT      : '¬'       |  N O N |  N O T ;

/*====================== La taxonomie ============ */
ISA      : '⊑' | '⊆' |  E S T  |  I S A ;
```

```
/*====================== Les modificateurs d'axiome ============ */
INV : '-1' | I N V | I N V E R S E ;
FCT : 'ƒ'  | F C T | F O N C T I O N E L L E | F U N C T I O N A L ;

/*======== À compléter grâce à la référence ISO ============= */
LNG :
  'de' | 'en' | 'es' | 'fr' | 'it' ;


/*
-- ========================================================================= Z
-- fin
-- ========================================================================= Z
*/
```

Grammaire syntaxique

```
/*
-- ========================================================================A
Produit : OntoRelA
Segment : MicroOnto
Composant : MOnto_LEX.g4
Encodage : UTF8 sans BOM, fin de ligne Unix (LF)
Responsable : Christina.Khnaisser@USherbrooke.ca
Description : Mots réservés et conventions lexicales spécifiques
  MicroOnto (v1).
Statut : V1

*Copyright* 2019 GRIIS (http://griis.ca/)
GRIIS (Groupe de recherche interdisciplinaire en informatique de la
  santé)
Faculté des sciences et Faculté de médecine et sciences de la santé
Université de Sherbrooke
Sherbrooke(Québec)  J1K 2R1  CANADA
[CC-BY-NC-3.0 (http://creativecommons.org/licenses/by-nc/3.0)]

*Références*
  S.O.

*Tâches projetées et questions*
TODO 2019-01-19 [CK] Permettre les participations dans les deux sens.
TODO 2019-01-19 [LL] Ajouter le concept d'Alias.
TODO 2017-08-14 [LL] Traiter les équivalences.
TODO 2017-08-14 [LL] Traiter les individus.
TODO 2017-08-14 [LL] Ajouter un langage de définition de types.

*Références*
  S.O.

@version 1.0.0
@author [CK] Christina.Khnaisser@USherbrooke.ca
@author [LL] Luc.Lavoie@USherbrooke.ca
-- ========================================================================A
*/

grammar MicroOnto;
import MOnto_LEX, LEX;

@header {
package ca.griis.monto.microOnto;
```

```
/**
 * Syntaxe du langage MicroOnto.
 *
 * @version 0.1.0 2018-04-08
 * @author [CK] Christina.Khnaisser@USherbrooke.ca
 * @author [LL] Luc.Lavoie@USherbrooke.ca
 */
}

@parser::members
{
  // public AnalyseurDLUS mru = new AnalyseurDLUS();
  // public Descripteur trace = mru.getTrace();
}

//
  ======================================================================
  ======
// Catégorie initiale : ontologie_def
// ...
// note : ...
// ----

ontology_def :
  (prefix_def)*
  ONTOLOGY ontology_dec
  (entity_def)*
  ;
prefix_def :
  PREFIX alias? ':' IDENT
  ;
entity_def :
  class_def | property_def | type_def // | equivalence_def |
  individal_def
  ;
class_def :
  CLASS class_dec inheritance* axiom*
  ;
property_def :
  PROPERTY property_dec (dataProperty_def | objectProperty_def)
  ;
dataProperty_def :
  inheritance*
  // Domaine Portee                                -- il faut au moins
  ajouter le domaine et la portée
  // Domaine participation Portee participation -- ou mieux, permettre
  une (ou deux) participation(s)
  // INV? FCT?                                     -- désormais inutile?
  ;
objectProperty_def :
  inheritance*
  inverse_def*
  // Domaine Portee                                -- il faut au moins
  ajouter le domaine et la portée
  // Domaine participation Portee participation -- ou mieux, permettre
  une (ou deux) participation(s)
  ;
type_def :
  TYPE type_dec
  // la définition du type est pour le moment externe dans tous les cas
  ;
inheritance :
  ISA reference
```

```
  ;
inverse_def :
  INV property
  ;
/*
 À venir

equiv_def :
  EQUIV class_dec association_exp
  ;
*/

/*====================== Les déclarations ====================== */
ontology_dec :
  declaration
  ;
class_dec :
  declaration
  ;
property_dec :
  declaration
  ;
type_dec :
  declaration
  ;
declaration :
  iri (LABEL label+)? (DESC description+)?
  ;
label :
  annotation
  ;
description :
  annotation
  ;
annotation :
  '@' LNG STRING
  ;
/*====================== Les expressions ====================== */
axiom :
  property participation association_exp
  ;

association_exp :
  association_exp AND association_exp
  | association_exp OR association_exp
  | property participation association_exp
  | ontoClass
  | '(' association_exp ')'

  ;
/*====================== Les références ====================== */
ontoClass :
  reference
  ;
type :
  reference
  ;
property :
  reference
  ;
reference :
  iri
  ;
```

```
iri :
  alias? ':' iri_local
  ;
alias :
  IDENT
  ;
iri_local :
  IDENT
  ;
participation :
  '[' min '..' max ']'
  ;
min :
  INTEGER
  ;
max :
  INTEGER | '*'
  ;

/*
-- ======================================================================B
Pour normaliser, il suffit de limiter les catégories suivantes en
  préservant
l'équivalence :

classe_exp :
  classe
  ;
relation_exp :
  relation
  ;
-- ======================================================================B
*/

/*
-- ======================================================================Z
-- fin MicroOnto.g4
-- ======================================================================Z
*/
```

Annexe 2
Extension de la comparaison des méthodes de conversion ontologique-relationnelle

Suite à la revue de la littérature publiée [Khnaisser et al. 2018] (voir annexe 3) une colonne pour OntoRelα et une ligne pour la temporalité ont été ajoutées pour voir les contributions de la thèse.

Tableau 14. Comparaison entre les méthodes A1-A11 et OntoRelα

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	OntoRelα
Ontology												
Ontology language	OWL ?	OWL ?	OWL ?	OWL ?	OWL DL	OWL DL	OWL QL, RL, EL	OWL ?	OWL DL	OWL DL	OWL ?	OWL QL, RL, EL
Schema												
Structure	? S	BCNF S	3NF S	BCNF S	BCNF S	3NF S	BCNF S	? S	? S	? G	? S	**6NF S**
Temporal	No	No	No	No	No	No	No	No	No	No	No	**Yes**
Domains	No	No	No	No	No	No	No	No	No	No	No	**Yes**
Primary keys	G	C	G	G	C	G	C	G	G	G	?	G
Secondary keys	No	?	No	Yes	?	?	Yes	?	?	Yes	?	**Yes**
Foreign keys	?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Participation constraints	?	?	No	No	No	No	Yes	No	No	Yes	?	**Yes**
General constraints	?	?	Yes	Yes	?	Yes*	Yes	Yes*	No	Yes	?	**Yes**
Modification procedure	No	No	No	No	No	No	No	No	No	Yes	?	No*
Target DBMS	?	OntoDB	PostgreSQL	?	?	many	?	MySQL	MS-SQL	?	?	PostgreSQL*
Process												
Axiom reduction	?	?	No	?	?	No	No	?	?	?	No	**Yes**
Intermediate structure	FOL	MOF	No	OWL	RDF	No	?	Jena	Jena	?	?	µOnto
Type conversion	?	?	No	?	?	?	Yes	Yes	Yes	?	?	**Yes**
Restriction conversion	?	I	No	I	No	No	E*	I	I	E	?	**E**
Individual conversion	Yes	No	No	No	Yes	No	Yes	Yes	No	Yes	Yes	No*
Annotation conversion	No	No	No	Yes	Yes	No	No	No	No	No	?	**Yes**
Structural reversibility Onto <-> REL	Yes Yes	Yes ?	No No	Yes ?	No No	? ?	No No	Yes No	Yes ?	? ?	No No	Yes No*
Tuples reversibility Tuple <-> RDF	Yes Yes	Yes ?	No No	Yes ?	Yes Yes	? ?	Yes Yes	? ?	? ?	? ?	No No	No* No*
Tool												
Existence	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	?	Yes	Yes
Availability	No	?	?	No*	Yes*	?	?	?	?	?	?	**Yes**

Références des articles

A1. Dou et al. [Dou et al. 2010]
A2. Bellatreche et al. [Bellatreche et al. 2010]
A3. Saccol et al. [Saccol et al. 2011]
A4. Vyšniauskas et al. [Vyšniauskas et al. 2012]
A5. Hornung and May [Hornung and May 2013]

A6. Podsiadły-Marczykowska et al [Podsiadły-Marczykowska et al. 2014]
A7. Jiménez-Ruiz et al. [Jiménez-Ruiz et al. 2015]
A8. Ho et al. [Ho et al. 2015]
A9. Afzal et al. [Afzal et al. 2016]
A10. Achpal et al. [Achpal et al. 2016]
A11. Mahmudi et al. [Mahmudi et al. 2018]

Notes

- For A6 and A8, the general constraints are generated only for the OWL `DataRange oneOf` axiom using check constraints.
- For A4 a tool called OWL2RDB[51] for Protégé 4.1 exists but it seems that it is no longer supported on version 5+.
- For OntoRelα the addition of modification procedures, support of other RDBMS, individual conversion, structural and tuple reversibility are under development.
- A5. Tool availability: the tool is a web application and works with Mondial ontology.
- A7. Restriction conversion: CHECK constraints are generated according to individual values.

Criteria

- Ontology language – the ontology language supported by the method: OWL-DL, OWL-QL, OWL-RL, OWL- EL, RDF(S), DAML, etc.
- Structure normalization – the relational schema normal form: 3NF, BCNF, 5NF or 6NF. This criterion can be deduced from the conversion rules.
- Structure scope – the form of the predicate represented by the relvars of the generated relational: schema generic [G] (RDF generic style <predicate, subject, object>) or specific [S] (a specific predicate per class or association).
- Temporal feature: does the method produce a temporal schema from the ontology?
- Domains – does the method convert the ontology data types and their constraints into domains (e.g. CREATE DOMAIN in PostgreSQL)?
- Primary keys – does the method generates [G] or calculates [C] the primary keys? A generated key is an artificial key defined independently of the set of axioms. A calculated key is a key deduced from the set of axioms.
- Secondary keys – does the method generate the secondary key from the set of axioms?
- Foreign keys – does the method convert the appropriate axioms into foreign keys?

[51] https://protegewiki.stanford.edu/wiki/OWL2ToRDB

- ◊ General constraints – does the method convert cardinalities into general constraints?
- ◊ Modification procedures – does the method define the procedures for modifying the data (insert, delete, and update triggers)?
- ◊ Target RDBMS – such as PostgreSQL, MySQL, Oracle, MSSQL, etc.
- ◊ Axiom reduction – does the conversion process deals with composite axiom?
- ◊ Intermediate structure – the intermediate data structure used for the conversion of OWL into a relational schema: MOF (Meta-Object Facility) FOL (First order logic), RDF, Jena model, etc.
- ◊ Type conversion – does the conversion process specifies or configures the conversion rules between ontology types and SQL types?
- ◊ Restriction conversion – does the conversion process converts the restrictions to general constraints? If yes: is explicit conversion [E] or metadata (implicit conversion [I]).
- ◊ Annotation conversion – does the conversion process converts the annotations to document the relational schema?
- ◊ Structural reversibility – does the conversion process makes it possible to refer to the ontology construct? Structural reversibility algorithm – does the method describes the algorithm and proposes an implementation of structural reversibility?
- ◊ Tuples reversibility – does the conversion process makes it possible to import tuples stored in the DB in their full ontological expression? Tuples reversibility algorithm – does the method describes the algorithm and proposes an implementation of tuples reversibility?
- ◊ Existence – has the method been implemented?
- ◊ Availability – is the tool publicly available?

Annexe 3
Generating Relational Database using Ontology Review : Issues, Challenges and Trends

KHNAISSER, C., LAVOIE, L., BURGUN, A., AND ETHIER, J.-F. 2018. Generating Relational Database using Ontology Review : Issues, Challenges and Trends. International Journal of Advanced Computer Science and Applications (IJACSA) 9, 6.

Generating relational database using ontology review

Issues, challenges and trends

Christina Khnaisser
Département d'informatique
Université de Sherbrooke
Sherbrooke, Canada
Christina.Khnaisser@usherbrooke.ca

Anita Burgun
INSERM UMR 1138 team 22, CRC
Université Paris Descartes
Paris, France

Luc Lavoie
Département d'informatique
Université de Sherbrooke
Sherbrooke, Canada
Luc.Lavoie@usherbrooke.ca

Jean-Francois Ethier
Département de médecine / informatique
Université de Sherbrooke, Sherbrooke, Canada
INSERM UMR 1138 team 22, CRC
Paris, France
jf.ethier@usherbrooke.ca

***Abstract*— A huge amount of data is being generated every day from different sources. Access to these data can be very valuable for decision-making. Nevertheless, the extraction of information of interest remains a major challenge given a large number of heterogeneous databases. Building shareable and (re)usable data access mechanisms including automated verification and inference mechanisms for knowledge discovery needs to use a common knowledge model with a secure, coherent, and efficient database. For this purpose, an ontology provides an interesting knowledge model and a relational database provides an interesting storage solution. Many papers propose methods for converting ontology to a relational database. This paper describes issues, challenges, and trends derived from the evaluation of 10 methods using 23 criteria. Following this study, this paper shows that none of the methods are complete as well as the conversion process does not use the full expressivity of ontology to derive a complete relational schema including advanced constraints and modification procedures. Thus, more work must be done to decreases the gap between ontologies a relation database.**

Keywords — ontology; relational database; database modeling; knowledge model; ontology to relational database;

I. INTRODUCTION

Information systems are now at the center of decisions in many areas (healthcare, economic, industrial, manufacturing and so on). A huge amount of data is being generated every day from different sources. Organizations desire to reuse this data for many kinds of analyses to enhance decision-making. The correctness of decision-making depends on the quantity but also on the quality of data collected. Nevertheless, data is stored in different sources that are structured (structural heterogeneity) and encoded (terminological heterogeneity) in different ways. On the one hand, to use data efficiently and correctly from these various heterogeneous sources, experts would ideally be able to express their queries according to a unified knowledge model that represents their domain without the need to know the structure of each database nor to manually extract data from many sources each time. The resulting data should also be available in a unified format reflecting the knowledge model used to define the query. On the other hand, data managers must be able to create, manage, and maintain data with the least possible resources while ensuring its fidelity, integrity, and traceability of its evolution. For this purpose, data integration is the mechanism used for combining data from different sources in a unique unified model offering a single access point. In the database field, the two main and widely used techniques to represent a data model (let's call them conventional techniques) are the entity-relationship model [1] and the object-oriented model [2] which are otherwise mutually convertible [3]. However, these conventional techniques do no longer provide expressivity that is sufficiently complete to semantically interpreted and widely reused data outside a restricted field of application [4]. The current trend in data integration is, therefore, the use of knowledge to enhance the process [4], [5].

In many heterogeneous environments, a knowledge model seems very useful to decipher source structure, and isolate interesting data elements to extract and combine [4]. In the early 80s, the computer science community adopted the ontologies as a knowledge model with reasoning abilities [6] to provide a shared conceptualization of some domain of interest [7]. Since then, ontologies (such as those expressed using the OWL language [8]) have been used in different ways: database modeling, data integration, data mapping, data exchange, data annotation, information retrieval, knowledge discovery, and so on [9]. In particular, ontologies are becoming an important tool for data integration because they handle semantic interoperability by describing a common understanding of data without preoccupation with the underlying layer [7], [4]. In addition, sound integration cannot be done without handling data integrity when a large amount of data sources is highly fragmented and heterogeneous. In light of the recent technology, Relational Database Management System (RDBMS) with vertical representation [10] and in-memory databases [11] are performing significantly better than "triple stores" sometimes used in conjunction with ontologies [12]. Thus, a RDBMS is needed to allow multiple users to store, modify, and interrogate

a large volume of data in a concurrent, reliable, and secure manner [13]–[15].

Yet, since ontologies offer excellent data integration support for disparate systems, a relational database derived directly from an ontology can be hypothesized to be the best way to ensure data integrity and a unique data access point for a large volume of data. First, modeling a relational database using ontologies (rather than entity-relationship) allows the reuse of ontology in many tasks facilitating queries expression as well as ensuring semantic and structural uniformity [16], [17]. Second, with the axiomatic model underlying the ontology, data storage and verification can be automated, increasing data integrity. Reasoning mechanisms can also be leveraged to allow for knowledge discovery. Finally, both models (ontological and relational) must be used together to benefit from the expressiveness of ontologies and the maturity of databases [16].

Several methods have been proposed in the literature to convert an ontology into a relational database [5]. Nevertheless, methods differ in terms of the various ontological constructs covered to generate an enriched relational model. Therefore, 10 methods using 23 specific criteria were described and analyzed.

The rest of the paper is organized as follows: Section II describes the study methodology including conversion issues and challenges as well as criteria definition and the selection process. Section III presents the analysis of the evaluated methods. Section IV elaborates on the advantages of using ontologies and gives an overview of current trends. Section V concludes the paper. Finally, in the appendix, the results of each criterion for the 10 methods are presented in a table.

II. STUDY METHODOLOGY

In this section, first, some ontology and relation database constructs are briefly presented. Then, some noteworthy conversion issues and, challenges are described. Twenty-three evaluation criteria were then derived to evaluate different methods, analyze the challenges, and underline the trends. Using the criteria, the evaluation of 10 methods is presented in the form of a table in the appendix.

A. Issues and challenges

An ontology and a relational database (relational theory) represent facts using different modeling construct [18]. An ontology is defined using classes, individuals, axioms, properties (object properties and data properties), datatypes, and annotations. For more detail please refer to [8], [19]. A relational database is defined by a set of relational variables (a.k.a *relvar* in relational theory or table in SQL), each relational variable is defined by a set of attributes (pairs of a unique name and a datatype) and constraints. For more detail please refer to the references [20], [21]. Both models also share common foundations: the set theory and the first order logic. Thus, a conversion from one to the other is possible, at least in part, but the main challenge is to maintain the richness of ontological definitions in the resulting relational database by converting uniformly and consistently ontology constructs into a relational construct. Some related challenges are now presented.

Preserve property cardinalities

An axiom is an expression that links classes or individuals using properties and cardinalities. A cardinality represents the number of individuals of the related entities that can participate regarding a property. A cardinality is represented as a participation with a range having a minimum value and a maximum value such as [0..1], [1..1], [0..n], [1..n], [0..*], [1..*] and [n..n] where n is a positive integer representing the exact participation value, and a "*" (star) is an undefined participation. In a relational database, an axiom can be represented as candidate keys (primary keys, unique keys), referential keys (foreign keys), and general constraints (functions and triggers). A conversion process that handles cardinality constraints increases data integrity and automatic inconsistency detection.

Handle missing information

In an ontology, properties of some class can be optional (as attributes in a relational database). That is, the value can be unknown (in some cases) or inapplicable (in some other cases). This can be implemented by the use of null values, but this implementation is not semantically complete. Moreover, null values can introduce data inconsistency in query results. Thus, it is recommended to avoid them by using other modeling techniques as described in [22], [23].

Preserve axioms expressivity

Axioms can be defined using different forms: simple or composite. A simple axiom is defined by atomic entities including a property, one or two classes, and a datatype. Several simple axioms can be combined by set operators (intersection and union) and logical operators (conjunction and disjunction) thus forming composite axioms. Let be the following composite axiom ***A OP qt (B ∩ C)*** where ***A***, ***B***, and ***C*** are classes, ***OP*** is an object property, and ***qt*** is a quantifier: some, only, min, max, exactly; it is possible to express it in 3 simple axioms: ***A OP qt Z ; Z isa B ∧ Z isa C*** where **Z** is a new class subclass of Thing. In a relational database, such operators are defined in the conceptual model and may have different transformations [3] generating different structures and constraints. This may be an issue when using popular medical and biomedical ontologies (see ontologies in OBO foundry repositories [24]) because composite axioms are frequently used. There is a need for a conversion process that uniformly converts such axioms by preserving the complete semantic.

Maintain Ontology-SQL type compatibility

A type (datatype) is a set of values. On the one hand, in computing, these sets are necessarily finite but this restriction is not automatically applicable on a data property. On the other hand, for some ontology-type (e.g. owl:rational, xsd:positiveInteger) there is no direct mapping to an SQL-type. Furthermore, SQL-types compatibility and exact handling depend on the target RDBMS. This must also be considered when converting types.

Enable structural and tuple reversibility

One of the objectives of using ontologies is the reuse of the model (described entities and axioms) in several tasks. Once data is integrated, a sound mechanism for linking tuples to entities can be very valuable for knowledge discovery and ontology-based data access [7]. In this context, the reversibility

between the components of an ontology and those of a relation schema is an essential point to take into account when defining the conversion process. Two reversibility features can be defined: the structure reversibility, the ability to reconstruct or identify an ontological construct from a relational construct; the tuples reversibility, the ability to reconstruct individuals (i.e. as RDF triples) from tuples.

Handle knowledge and schema evolution

Knowledge is in constant evolution. This implies changes to ontologies with ensuing repercussions on the related relational schema. The schema must, therefore, cope with it while maintaining earlier knowledge interpretations and preserving coherent data [17]. Moreover, with the opportunity to easily access data, new needs will emerge, and existing needs may change. The impact when integrating knowledge change that implies schema evolution can be very large. Consequently, the conversion process needs to be defined in a way to facilitate structure modification and extensibility.

Maintain schema documentation

Relational schema documentation is often ignored despite the added value in many tasks such as data querying, mapping definition, application development and so on [25]. Without a clear definition of the data or the database structure, several tasks cannot be verified or even done [4]. Part of the semantics of the class is carried by the axioms but complementary information can be found in annotations. An annotation in the ontology represents a text in a specific language that defines various aspects of an entity. Using annotations can be beneficial for schema generation and documentation. Examples of interesting types of annotations are: labels (e.g. rdf:label) and comments (e.g. rdf:comment), and definitions (e.g. definition from IAO ontology [1]) which propose a common language description of classes, individuals, and properties.

B. Criteria definition

Based on the issues and challenges described above, criteria were defined and grouped into four categories: the ontology criteria, the relational schema criteria, the conversion process criteria and tool implementation criteria.

Ontology criteria

- Ontology language – the ontology language supported by the method: OWL-DL, OWL-QL, OWL-RL, OWL-EL, RDF(S), DAML, etc.

Relational schema criteria

- Structure normalization – the relational schema normal form? : 3NF, BCNF, 5NF or 6NF. This criterion can be deduced from the conversion rules.
- Structure scope – the form of the predicate represented by the relvars of the generated relational: schema generic [G] (*RDF generic style* <predicate, subject, object>) or specific [S] (a specific predicate per class or association).
- Domains[2] – does the method convert the ontology data types and their constraints into domains (e.g. CREATE DOMAIN in PostgreSQL)?
- Primary keys – does the method generate [G] or calculate [C] the primary keys? A generated key is an artificial key defined independently of the set of axioms. A calculated key is a key deduced from the set of axioms.
- Secondary keys – does the method generate the secondary key from the set of axioms?
- Foreign keys – does the method convert the appropriate axioms into foreign keys?
- General constraints – does the method convert cardinalities into general constraints?
- Modification procedures – does the method define the procedures for modifying the data (insert, delete, and update triggers)?
- Supported target RDBMS – such as PostgreSQL, MySQL, Oracle, MSSQL, etc.

Conversion process criteria

- Axiom normalization – does the conversion process deal with composite axiom?
- Intermediate structure – the intermediate data structure used for the conversion of OWL into a relational schema: MOF (Meta-Object Facility) FOL (First order logic), RDF, Jena model, etc.
- Type conversion – does the conversion process specify or configure the conversion rules between ontology types and SQL types?
- Restriction conversion – does the conversion process convert the restrictions to general constraints? If yes: is explicit conversion [E] or metadata (implicit conversion [I]).
- Annotation conversion – does the conversion process convert the annotations to document the relational schema?
- Structural reversibility – does the conversion process make it possible to refer to the ontology construct?
- Structural reversibility algorithm – does the method describe the algorithm and propose an implementation of structural reversibility?
- Tuples reversibility – does the conversion process make it possible to import tuples stored in the DB in their full ontological expression?
- Tuples reversibility algorithm – does the method describe the algorithm and propose an implementation of tuples reversibility?

[1] http://www.obofoundry.org/ontology/iao.html

[2] A domain as defined by the type theory is a finite set of values and a type is a constrained domain that restricts the accepted values.

Implementation criteria

- Existence of an implementation – has the method been implemented?
- Tool availability – is the tool publicly available?

C. Literature selection process for the evaluated methods

The search process started in February 2018. The first intent of the search was to find a publicly available tool. To identify methods implemented or updated with current technologies, only papers from the year 2010 and up were retained. The search was conducted with Google Scholar and Engineering Village using the following keywords: "Ontology to relational schema", "Ontology to relational database", "Relation schema from ontology". From 178 papers, 10 were selected and evaluated. The selection was based on the completeness of the paper. The completeness was defined regarding the number of criteria. A paper was selected if at least 15 (over 23) criteria can be evaluated. Moreover, authors of the papers were contacted to validate the evaluation or complete the missing values in the evaluation table and 6 out of 10 responded.

III. RESULTS

This section presents the analysis of the results. The analysis is divided into 2 categories: general observation and, criteria observation. The appendix I presents specific criteria results for each method.

A. General observation

The OWL language is the most popular ontology language. However, the methods differ from one another according to the ontology constructs taken into account in the conversion process. The common ontology constructs used are: classes, objects properties, data properties, subclass axioms with simple restriction (some, exactly, min and max), functional properties characteristic as well as the domain and the range of properties. The methods also differ from one another in several other aspects: the conversion rules, the conversion steps, the quality of the relational schema generated, the availability of the tool, etc.

First, there seems to be a consensus on some conversions rules, especially the class conversion rule is the same for all methods, a class is transformed into a relation. But, properties and axioms are generated differently, i.e. a property can be transformed into an attribute or into a relation depending on the granularity of the conversion rules. In addition, several conversions are suggested to increase the integrity of the data, including the generation of secondary keys, general constraints, modification procedures, and so on. Furthermore, no tool supporting multiple ontologies with an advanced conversion process that was publicly available has been identified. It is thus very difficult to reproduce or share the results, let alone reuse it.

B. Criteria observation

Ontology criteria

OWL defines four profiles: EL, RL, QL, and DL. The DL profile being the superset of the three others [26]. It is important to use DL profile to benefit from a higher degree of expressivity, and to cover more general modeling construct (i.e. universal and existential quantification, cardinality restrictions, functional properties, etc.). While some reasoning operations may be undecidable [26] the gain in expressivity is notable. On the other sides, optimized translation schema may be derived taking the restrictions of EL, RL, or QL into account.

DL is the most supported (7/10) and just one method supports specifically EL, QL and RL profiles, but no indication is given on the available optimizations, if any.

Relational schema criteria

The generated relational databases differ with respect to their structure and their constraints. On the one hand, all methods generate an ontology-specific relational schema except the method presented in [27]. The latter uses generic representations as a "triple store" where all the data are stored in a single large table (subject, predicate, object). This representation accepts any combination of "fact" but makes it difficult to verify data integrity and to process a large volume of data [13]. In addition, an object property is often converted to a join table, and a data property is always converted to an attribute. These conversions rarely take into account the value of the cardinality. On the other hand, regarding the schema constraint, primary key constraints are generated by most of the methods (7/10), and secondary keys are calculated only by few of them (3/10) using property characteristics (functional and inverse functional). In addition, only two methods generate general constraints, yet this generation is limited to an enumeration restriction (i.e. value enumeration inside a check constraint).

Furthermore, modification procedures are only generated by one method [27] and are limited to the insertion procedure. The generation is based on cardinality restrictions and on all the property characteristics (functional, inverse functional, transitive, symmetric, asymmetric, reflexive and, irreflexive).

Finally, the main RDBMS targeted are PostgreSQL, Oracle, MySQL, and MSSQL. Method [28] supports multiple RDBMS and method [29] uses an ontology database system called OntoDB.

Conversion process criteria

The conversion process is unique to each method. The methods differ in terms of the sequence of steps or conversion rules. A noteworthy point is that all methods only deal with simple axioms or do not give an explicit indication of the treatment of composite axioms.

Type conversion: few methods define conversion of ontology types to SQL types (4/10). In the case where a definition is presented, the conversion often depends on an internal configuration or a specific RDBMS. Thus, manual adaptation is needed to reuse the generated relational database in different RDBMS. In addition, some ontology-type does not have a direct mapping to SQL-type, a more advanced mapping mechanism is required.

Restriction conversion: for the majority of the methods (9/10) the participation applicable to an object property in an axiom are converted into referential keys. In addition, a few methods (2/10) generate general constraints (explicit conversion) and some of the methods (4/10) keep information about the participation in metadata tables (implicit conversion).

The latter case implies that cardinalities are not verifiable by the RDBMS unless there are constraints or automatisms that take advantage of them. However, these constraints or automatisms are not generated by these methods. In this case, the RDBMS cannot guarantee intrinsically the integrity of the data as described by the ontology. Yet, the integrity of the database can be evaluated externally if the information is present in the metadata, and internally if explicit constraints were generated.

Individual conversion: individuals are converted into tuples by five methods (5/10) so the initial database schema can contain tuples. This is an interesting feature as the database can be ready for querying. In addition, to benefit from it tuples reversibility can be valuable.

Annotation conversion: annotations are rarely used by methods (2/10) to document the database or the automate some conversion rules. Thus, a complementary mechanism is needed to fetch information from the ontology and the relational database to take full advantage of semantics.

Structure reversibility: none of the methods defines the reversibility explicitly. But, according to the overall conversion process, structure reversibility can be easily defined by six of the methods. Only two of the six methods present algorithms for this use.

Tuples reversibility: none of the methods defines the reversibility explicitly. But, according to the overall conversion process, tuples reversibility can be easily defined by five of the methods. Three over five methods present algorithms for this use.

Implementation criteria

Most methods have an implementation (7/10) but unfortunately only one is publicly available [30] and its use is limited to one ontology. Furthermore, the tool evaluation is rarely detailed which made it difficult to have a clear view of the evaluation scale.

IV. DISCUSSION

Ontologies are used as knowledge models to define axiomatically the application domain while the relational schema is used as a logical data model to store, modify, and retrieve in a secure way a large amount of data. In the context of data integration, the role of ontologies is twofold. First, an ontology presents a consensual knowledge model [31], thus more suitable for semantic interoperability and for defining a unique access point. Secondly, an ontology offers a formal definition of the database enabling automatic structure and data consistency verification (that can be done automatically by most of the reasoners). Both the ontological and the relational model must be used together to take advantage of the abstraction and expressiveness of ontologies as well as the operational functionalities of databases built using relational schema [16].

As illustrated here, more work is still needed to improve the overall database quality especially to consolidate the schema integrity. First, the relational schema must be normalized and general constraints must be defined based on the cardinality in the axiom. This implies better management of missing information, and participation of individuals according to the property. In addition, the normalization is even more important in the context of physical data warehousing (especially for temporal database and big data) or virtual data warehousing (mediation) where the data extracted from multiple sources are heterogeneous, highly fragmented and context dependent [17], [32]. A “high” normal form (like 5NF and 6NF) reduce uncontrolled redundancy and facilitate schema extension as every part of the schema represents one predicate [33]. Even more, the resulting structure (after normalization) is closer to the set-theory foundation of ontologies and databases, thus facilitating query formulation [34] as the query formulated using ontology entities can have a more direct mapping. Second, preserving axiom expressivity is important to guarantee lossless conversions. Some ontologies may be built with simple axioms definitions but in the biomedical field, this hypothesis is not guaranteed, and as mentioned earlier, multiple ontologies in the OBO foundry use this approach. Describing conversion rules according to axioms definition is, therefore, an important aspect to cover ontologies produced in a large number of domains. Third, the structure of relations in a database differs from the structure of ontological entities. "Ontological" views can be generated to provide a view of the ontology structure. Even more, through these views, modification procedures can be generated to provide better access and standardize data manipulation. Therefore, a method that allows the generation of views and modification procedures can be very beneficial for application development in the sense that data access and modification can be handled at the database level. Fourth, another interesting advantage of using ontologies is the capability to document entities using annotation. However, the methods do not use annotations to document the relational schema (actually documentation is rarely available). For example, ontology label annotation can be used to define different views with different languages increasing the schema accessibility. Finally, regarding ontology-SQL type compatibility, when the RDBMS allows it, domains must be created for ontology types to make it easier to change types during the life cycle of the database.

Maybe more importantly, knowledge and schema evolution are not mentioned in any paper. This subject may be out of the scope of the selected paper, but it is an important one to ensure a long-term solution. Sustainability is a challenge faced by every system, and explicit approaches to address this challenge are required.

Regarding structure reversibility and tuples reversibility, interesting work has been done in the reversibility aspect. That means that the methods are used in a more global context such as ontology data access, data acquisition, and data extraction.

Finally, what slowed down the adoption of ontologies is the lack of publicly accessible tools. Ontologies are by their very nature (shared understanding of a domain) public and shared resources. Thus, to encourage the reuse of ontologies or the use of common database schemas, tools must be publicly accessible and usable with multiple ontologies. Once this is the case, further work to assert that the method is usable at a certain scale will require several evaluations (tests) that are accepted by the community, easily used and reproducible.

V. Conclusion

Ontologies are definitely starting to be "popular" and they are now used in different forms. They are indeed a very promising approach to bring formal semantic to relational databases in order to enhance data interoperability. Nevertheless, in the context of using ontologies to generate a relational database, many issues remain to maintain the full expressivity of ontologies, among them: preserving property cardinalities and axiom expressivity, handling knowledge and schema evolution, handling missing information, and maintaining schema documentation. Complementary, relational approaches have much potential for bringing performance and power to ontology-based data operations. Finally, more work must be done to decreases the gap between ontologies and relation databases but the work presented is a step in this direction.

Acknowledgment

We thank the authors of the selected papers that revised the evaluation table, especially colleagues who took the time to reply and greatly enhance the scientific value of this work. We also thank master student Samuel Dussault for his help to fill the evaluation table. Finally, we would like to acknowledge the important support of the Unité de Soutien SRAP du Québec.

References

[1] P. P.-S. Chen, "The Entity-relationship Model—Toward a Unified View of Data," *ACM Trans Database Syst*, vol. 1, no. 1, pp. 9–36, 1976.

[2] OMG, "Unified Modeling Language Specification Version 2.0," *Object Management Group*, 2005. [Online]. Available: http://www.omg.org/spec/UML/2.0/. [Accessed: 31-Jan-2018].

[3] R. Elmasri and S. Navathe, *Fundamentals of database systems*, 6th ed. Boston: Addison-Wesley, 2011.

[4] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, R. Rosati, and M. Ruzzi, "Using OWL in data integration," pp. 397–424, 2010.

[5] G. D. Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati, "Using Ontologies for Semantic Data Integration," in *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*, Springer, Cham, 2018, pp. 187–202.

[6] T. Gruber, "Ontology," *Encyclopedia of Database Systems*. Springer US, Boston, MA, pp. 1963–1965, 2009.

[7] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati, "Linking Data to Ontologies," *J. Data Semant. X*, vol. LNCS, no. 4900, pp. 133–173, 2008.

[8] B. Motik, P. F. Patel-Schneider, and B. Parsia, "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)," 2012. [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/. [Accessed: 03-Apr-2016].

[9] A. Gali, C. X. Chen, K. T. Claypool, and R. Uceda-Sosa, "From Ontology to Relational Databases," in *Conceptual Modeling for Advanced Application Domains*, S. Wang, K. Tanaka, S. Zhou, T.-W. Ling, J. Guan, D. Yang, F. Grandi, E. E. Mangina, I.-Y. Song, and H. C. Mayr, Eds. Springer Berlin Heidelberg, 2004, pp. 278–289.

[10] D. Krneta, V. Jovanovic, and Z. Marjanovic, "A direct approach to physical Data Vault design," *Comput. Sci. Inf. Syst.*, vol. 11, no. 2, pp. 569–599, 2014.

[11] J. Lee *et al.*, "SAP HANA distributed in-memory database system: Transaction, session, and metadata management," 2013, pp. 1165–1173.

[12] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach, "SW-Store: a vertically partitioned DBMS for Semantic Web data management," *VLDB J.*, vol. 18, no. 2, pp. 385–406, 2009.

[13] A. A. Tzacheva, T. S. Toland, P. H. Poole, and D. J. Barnes, "Ontology Database System and Triggers," in *Advances in Intelligent Data Analysis XII*, 2013, pp. 416–426.

[14] I. Astrova, N. Korda, and A. Kalja, "Storing OWL ontologies in SQL relational databases," *Int. J. Electr. Comput. Syst. Eng.*, vol. 1, no. 4, pp. 242–247, 2007.

[15] L. Al-Jadir, C. Parent, and S. Spaccapietra, "Reasoning with large ontologies stored in relational databases: The OntoMinD approach," *Data Knowl. Eng.*, vol. 69, no. 11, pp. 1158–1180, 2010.

[16] D.-E. Spanos, P. Stavrou, and N. Mitrou, "Bringing Relational Databases into the Semantic Web: A Survey," *Semant Web*, vol. 3, no. 2, pp. 169–209, 2012.

[17] C. Khnaisser, L. Lavoie, H. Diab, and J.-F. Ethier, "Data Warehouse Design Methods Review: Trends, Challenges and Future Directions for the Healthcare Domain," in *New Trends in Databases and Information Systems*, T. Morzy, P. Valduriez, and L. Bellatreche, Eds. Springer International Publishing, 2015, pp. 76–87.

[18] C. Pinkel *et al.*, "RODI: A Benchmark for Automatic Mapping Generation in Relational-to-Ontology Data Integration," in *The Semantic Web. Latest Advances and New Domains*, F. Gandon, M. Sabou, H. Sack, C. d'Amato, P. Cudré-Mauroux, and A. Zimmermann, Eds. Springer International Publishing, 2015, pp. 21–37.

[19] F. Baader, Ed., *The description logic handbook: theory, implementation, and applications*, 2. ed., paperback ed. Cambridge: Cambridge Univ. Press, 2010.

[20] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[21] H. Darwen and C. J. Date, "The Third Manifesto," *SIGMOD Rec*, vol. 24, no. 1, pp. 39–49, 1995.

[22] H. Darwen, "How to Handle Missing Information without Using NULL," Warwick University, 2005.

[23] C. J. Date and H. Darwen, *Database Explorations: essays on the Third Manifesto and related topics*. Trafford Publishing, 2010.

[24] OBO, "The Open Biological and Biomedical Ontology Foundry," *The OBO Foundry*, 2018. [Online]. Available: http://www.obofoundry.org/. [Accessed: 06-May-2018].

[25] C. Curino, G. Orsi, E. Panigati, and L. Tanca, "Accessing and Documenting Relational Databases through OWL Ontologies," in *Flexible Query Answering Systems*, 2009, pp. 431–442.

[26] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, "OWL 2 Web Ontology Language Profiles (Second Edition)." [Online]. Available: https://www.w3.org/TR/owl2-profiles/#OWL_2_RL. [Accessed: 10-May-2018].

[27] S. Achpal, V. Bannihatti Kumar, and K. Mahesh, "Modeling Ontology Semantic Constraints in Relational Database Management System," presented at the IMECS International Multiconference of Engineers and Computer Scientists, Hong Kong, 2016.

[28] T. Podsiadły-Marczykowska, T. Gambin, and R. Zawiślak, "Rule-Based Algorithm Transforming OWL Ontology Into Relational Database," in *Beyond Databases, Architectures, and Structures*, 2014, pp. 148–159.

[29] L. Bellatreche, Y. Ait-Ameur, and C. Chakroun, "A design methodology of ontology based database applications," *Log. J. IGPL*, vol. 19, no. 5, pp. 648–665, 2010.

[30] T. Hornung and W. May, "Experiences from a TBox Reasoning Application: Deriving a Relational Model by OWL Schema Analysis," in *Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013) co-located with 10th Extended Semantic Web Conference (ESWC 2013), Montpellier, France, May 26-27, 2013*, 2013, vol. 1080.

[31] G. Pierra, "Context Representation in Domain Ontologies and Its Use for Semantic Integration of Data," in *Journal on Data Semantics X*, Springer, Berlin, Heidelberg, 2008, pp. 174–211.

[32] N. Golov and L. Rönnbäck, "Big Data normalization for massively parallel processing databases," *Comput. Stand. Interfaces*, vol. 54, Part 2, pp. 86–93, Nov. 2017.

[33] C. J. Date, *Database Design & Relational Theory*. Sebastopol, Calif.: O'Reilly Media, 2012.

[34] P. LePendu, D. Dou, Z. M. Ariola, and C. Wilson, *Ontology-based Relational Databases*. 2007.

[35] D. Dou, H. Qin, and P. Lependu, "OntoGrate : Towards Automatic Integration For Relational Databases And The Semantic Web Through An Ontology-Based Framework," *Int. J. Semantic Comput.*, vol. 04, no. 01, pp. 123–151, 2010.

[36] D. d B. Saccol, T. d C. Andrade, and E. K. Piveta, "Mapping OWL ontologies to relational schemas," in *2011 IEEE International Conference on Information Reuse Integration*, 2011, pp. 71–76.

[37] E. Vyšniauskas, L. Nemuraitė, and B. Paradauskas, "Preserving Semantics of Owl 2 Ontologies in Relational Databases Using Hybrid Approach," *Inf. Technol. Control*, vol. 41, no. 2, pp. 103–115, 2012.

[38] E. Jiménez-Ruiz *et al.*, "BootOX: Practical Mapping of RDBs to OWL 2," in *The Semantic Web - ISWC 2015*, 2015, pp. 113–132.

[39] L. T. T. Ho, C. P. T. Tran, and Q. Hoang, "An Approach of Transforming Ontologies into Relational Databases," in *Intelligent Information and Database Systems*, 2015, pp. 149–158.

[40] H. Afzal, M. Waqas, and T. Naz, "OWLMap: Fully Automatic Mapping of Ontology into Relational Database Schema," *Int. J. Adv. Comput. Sci. Appl. IJACSA*, vol. 7, no. 11, 2016.

APPENDIX I

The table below presents the evaluation result of the criteria and of the requirement for the selected paper respectively. The value "?" means that the information was not found in the article. The evaluated papers are:

A1. Dou et al. [35]
A2. Bellatreche et al. [29]
A3. Saccol et al. [36]
A4. Vyšniauskas et al. [37]
A5. Hornung and May [30]
A6. Podsiadły-Marczykowska et al [28]
A7. Jiménez-Ruiz et al. [38]
A8. Ho et al. [39]
A9. Afzal et al. [40]
A10. Achpal et al. [27]

TABLE I. CRITERIA EVALUATION TABLE

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
Ontology										
Ontology language	OWL ?	OWL ?	OWL ?	OWL ?	OWL DL	OWL DL	OWL QL,RL,EL	OWL ?	OWL DL	OWL DL
Schema										
Structure	? S	BCNF S	3NF S	BCNF S	BCNF S	3NF S	BCNF S	? S	? S	? G
Domains	No	No	No	No	No	No	No	No	No	No
Primary Keys	G	C	G	G	C	G	C	G	G	G
Secondary Key	No	?	No	Yes	?	?	Yes	?	?	Yes
Foreign keys	?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Participation constraints	?	?	No	No	No	No	Yes	No	No	Yes
General constraints	?	?	Yes	Yes	?	Enum	Yes	Enum	No	Yes
Modification procedure	No	No	No	No	No	No	No	No	No	Yes
Target DBMS	?	OntoDB	PostgreSQL	?	?	Multi	?	MySQL	MSSQL	?
Process										
Axiom normalization	?	?	No	?	?	No	No	?	?	?
Intermediate Structure	FOL	MOF	No	OWL W3C	RDF	No	?	Jena	Jena	?
Type conversion	?	?	No	?	?	?	Yes	Yes	Yes	?
Restriction conversion	?	I	No	I	No	No	E*	I	I	E
Individual conversion	Yes	No	No	No	Yes	No	Yes	Yes	No	Yes
Annotation conversion	No	No	No	Yes	Yes	No	No	No	No	No
Structural reversibility	Yes Yes	Yes ?	No No	Yes ?	No No	? ?	No No	Yes No	Yes ?	? ?
Tuples reversibility	Yes Yes	Yes ?	No No	Yes ?	Yes Yes	? ?	Yes Yes	? ?	? ?	? ?
Tool										
Existence	Yes	Yes	Yes	?	No	Yes	Yes	Yes	Yes	?
Availability	No	?	?	?	Yes*	?	?	?	?	?

A5. Tool availability: The tool is a web application and works with Mondial ontology: http://www.semwebtech.org/rdf2sql/
A7. Restriction conversion: CHECK constraints are generated according to individual values

Annexe 4
Past Indeterminacy in Data Warehouse Design

KHNAISSER, C., LAVOIE, L., BURGUN, A., AND ETHIER, J.-F. 2017b. Past Indeterminacy in Data Warehouse Design. Database and Expert Systems Applications, Springer, Cham, 90–100.

Past Indeterminacy in Data Warehouse Design

Christina Khnaisser[1,2], Luc Lavoie[1], Anita Burgun[2], and Jean-François Ethier[1,2,3] (✉)

[1] Département d'informatique, Université de Sherbrooke, Sherbrooke, Canada
{christina.khnaisser, luc.lavoie}@usherbrooke.ca
[2] INSERM UMR 1138 team 22 Centre de Recherche des Cordeliers, Université Paris Descartes
anita.burgun@aphp.fr
[3] Département de médecine, Université de Sherbrooke, Sherbrooke, Canada
jf.ethier@usherbrooke.ca

Abstract. Traditional data warehouse design methods do not fully address some important challenges, particularly temporal ones. Among them past indeterminacy is not handled systematically and uniformly. Furthermore, most methods published until now present transformation approaches by providing examples rather than general and systematic transformation rules. As a result, real-world applications require manual adaptations and implementations. This hinders scaleability, long-term maintenance and increases the risk of inconsistency in case of manual implementation. This article extends the Unified Bitemporal Historicization Framework with a set of specifications and a deterministic process that defines simple steps for transforming a non-historical database schema into a historical schema allowing data evolution and traceability, including past and future indeterminacy. The primary aim of this work is to help data warehouse designers to model historicized schema based on a sound theory ensuring a sound temporal semantic, data integrity and query expressiveness.

Keywords. Data warehouse design, Temporal data warehouse, Temporal indeterminacy, Missing information.

1 Introduction

Historicization is the process that consists of transforming a non-historical (database) schema into a historicized one. There are complex design problems that need to be addressed [2]. A recent survey paper [4] did not identify new innovative models successfully addressing the following gaps. First, when applied to historical data warehouses (HDW), the published methods require multiple manual steps (both in terms of design and implementation) [12]. Second, the existing solutions are based on different data models (relational, entity-relationship, object-oriented, multidimensional) and use different structures and semantics. This diminishes generalizability and large-scale adoption. Finally, many solutions for temporal data warehouses do not handle missing information.

adfa, p. 1, 2011.

This paper extends the Unified Bitemporal Historicization Framework[1] (UBHF), to cope with past indeterminacy. UBHF and it's time model were formally defined in [11].

In 2006, Rizzi et al. [13] wrote: "Though a lot has been written about how data warehouse should be designed, there is no consensus on a design method yet". According to our last survey of data warehouse design (DWD) methods [12] this is still valid.

Two major temporal models have emerged in the literature and in our domain of interest (clinical data warehousing). The first one is the Bitemporal conceptual data model (BCDM) a bitemporal model based on SQL. Snodgrass presents design "best practices" to build a bitemporal schema starting from a conceptual model (entity relationship) ending with SQL code (and TSQL2 [22] a temporal SQL extension). BCDM was initially defined in [9], a more recent presentation can be found in [14] and an extension to temporal indeterminacy in [3]. The second one is the Date-Darwen-Lorentzos Model (DDLM) which is based on the relational theory. It proposes three submodels, two unitemporal model and one bitemporal model based on the third manifesto relational model [5] and Allen's interval logic [1].In previous work, a comparative study [10] showed interesting similarities (and some differences) between BCDM and DDLM when using UBHF to express them [11] but none can express past indeterminacy.

Regularly, patients cannot provide exact dates regarding important health events that happened sometimes more than 40 years ago. In the event of a fracture, the exact onset moment was known at some point in time but might have been forgotten, leading to a missing date in the electronic medical record. A related but different problem occurs when a new diagnosis is made (e.g. diabetes). While we know that the disease is present at the time of the first diagnosis, for most diseases, the exact onset moment is unknown and it could be days, weeks, months or even years (e.g. hypertension) before. Currently, temporal data related to the examples above is often not included in the database. In the case where the application forces the user to enter a precise value, the clinicians will input an approximate date. This can lead to significant inconsistencies in the treatment of temporal operations during queries or the exclusion of relevant information.

2 UBHF concepts

UBHF is a conceptual framework that defines specifications and deterministic transformations based on fundamental relational and temporal concepts [7, 14]. This paper, presents a synthesis of these concepts, a more elaborated version can be found in [11].

2.1 Time model

UBHF uses a discrete time model based on points and intervals derived from the one defined by Allen in [1] and used in [7]. TIMPEPOINT and PERIOD will be used as a representative time point type and time interval type in the following sections. The re-

[1] Technical report can be found at http://griis.ca/surl/ubfh-dexa

tained timelines are transaction time (@T) and valid time (@V) as defined in the consensus glossary [8]. In the context of a HDW, the distinction between future indeterminacy and past indeterminacy allows easier processing of the retrospective and perspective data.

Future indeterminacy (indeterminate end) concerns facts with a known beginning and an unknown ending. In BCDM, this endpoint is represented by the constant « until changed » for a valid-time period and with « forever » for a transaction-time period. These constants are encoded in the database tables as the maximal value of TIMESTAMP type (9999-12-31 when granularity is one day). In DDLM and UBHF, this is represented with the function *ufn* « until further notice » and is only used in query expression (not for storage purpose).

Past indeterminacy (indeterminate begin) concerns facts with a known ending and an unknown beginning. This indeterminacy is overlooked by most models including DDLM and BCDM. In UBHF, this is represented with the function saw « since a while » and is only used in query expression (not for storage purpose).

2.2 Timelines

UBHF represents a timeline by an attribute (called timeline attribute). A timeline attribute can have different types and values defined as follows:

- A period timeline attribute (be): where the beginning and the end point values are known.
- A point timeline attribute with unknown end value (bx): where the beginning point is known and the end is unknown..
- A point timeline attribute with unknown beginning value (xe): where the beginning point is unknown and the end is known.

The table below defines the notation used. Note that, for transaction timelines the xe-type is not defined, as in a DBMS the beginning point of a transaction is always known.

Table 1. Timeline attribute definition and notation.

Notation	Definition	Timeline
@Vbe	Valid time period	Valid
@Vbx	Valid time begin point	
@Vxe	Valid time end point	
@Tbe	Transaction time period	Transaction
@Tbx	Transaction time point	

2.3 Attributes and relations

In a non-historical schema, we conventionally distinguish between key and non-key attributes. A non-historicized relation R is denoted R(K, A), where:

- $K = \{k_1,\ldots,k_{|K|}\}$ is the set of key attributes ($|K| \geq 1$). Without loss of generality, we consider that each relation contains only one key - although this key may have more than one attribute.
- $A = \{a_1,\ldots,a_{|A|}\}$ is the set of non-key attributes ($|A| \geq 0$).

A historicized relation R is denoted R'(K, B, C, D_V, D_T) with $A = B \cup C$ where:

- $B = \{b_1,\ldots,b_{|B|}\}$ is the set of non-key attributes ($|B| \geq 0$) associated with a valid timeline attribute (called historicized attributes); B is a subset of A.
- $C = \{c_1,\ldots,c_{|C|}\}$ is the set of non-key attributes ($|C| \geq 0$) **not** associated with a valid timeline attribute(called non-historicized attributes); C is a subset of A.
- $D_V = \{@V, b_1@V,\ldots, b_{|B|}@V\}$ is the set of valid timeline attributes, with the following notations @V is associated with K, and $b_i@V$ is associated with $b_i \in B$.

$D_T = \{@T, a_1@T,\ldots, b_{|A|}@T\}$ is the set of transaction timeline attribute where @T is associated with K, and $a_i@T$ is associated with $a_i \in A$.In the context of R', K and "key" do not refer to the same entity. K is the key set of the original relation R where the "key" of R' is the union of K with D_V and D_T.

Keeping history changes may introduce redundancy, contradiction, circumlocution and non-denseness when attributes in the same relation are modified independently. DDLM studied these problems in detail [7] (chap. 5 and 13) and proposed constraints to avoid them. See section 4 for a generalized form of constraint definitions.

3 Historicization process

Data modifications may introduce data inconsistency as described in previous sections. Data inconsistency is usually addressed by schema normalization and constraint specification. The normalization process uses lossless relational decomposition to split a relation into smaller relations ("relparts" for short). Two types of such decomposition are used hereafter: projection-join decomposition (PJ) and the restriction-union decomposition (RU) [11]. The historicization process consists of the following activities which are presented in the remaining of the present section:

- Schema annotation (guided).
- Historical schema construction (automated).
- Temporal constraints specification (automated).

3.1 Schema annotation

The aim of the schema annotation is to "parameterize" the algorithm of the schema transformation according to the domain needs. For each relation R(K, A) of the initial schema previously normalized in 5th normal form (5NF) : Split A into B (attributes with valid time) or C (attributes without).

3.2 Modelling

For each relation R(K, B, C) the following steps are required:

1. Decompose the relation R into 6NF using the PJ decomposition[2]
   PJ(R_K{K}, R_b_1{K, b_1}, …, R_b_n{K, b_n}, R_c_1{K, c_1}, …, R_c_m{K, c_m}).

To facilitate constraint definition, these (numerous) relparts are conceptually grouped into three types of groupings [11]:

- The *K-grouping* of a relation R is the set of all the K relparts. A K-relpart, denoted R_K, is a relation with K and the associated timeline attributes only.
- A *b_i-grouping* of a relation R is the set of all b_i-relparts. Collectively, the b_i-groupings are called *B-groupings*.
- A *c_i-grouping* of a relation R is the set of all c_i-relparts. Collectively, the c_i-groupings are called *C-groupings*.

For each relation R(K, B, C) the following steps are required to produce the historical representation (i.e. R!VT(K, B, C, D_V, D_T)).

2. For each relpart in K-grouping and B-groupings: Add the timeline attribute @V.
3. For each relpart in K-grouping and B-groupings: Decompose each resulting relparts, using RU decomposition over the @V timeline attribute to separate timeline between @Vxe, @Vbe and @Vbx types, renaming the relparts accordingly.
4. For all relparts (in K-, B- and C-groupings): Add two transaction time relparts: one with @Tbx and one with @Tbe.

In the following, without loss of generality, all relations are considered bitemporal. If a relation is a transaction time relation, only the last step is required.

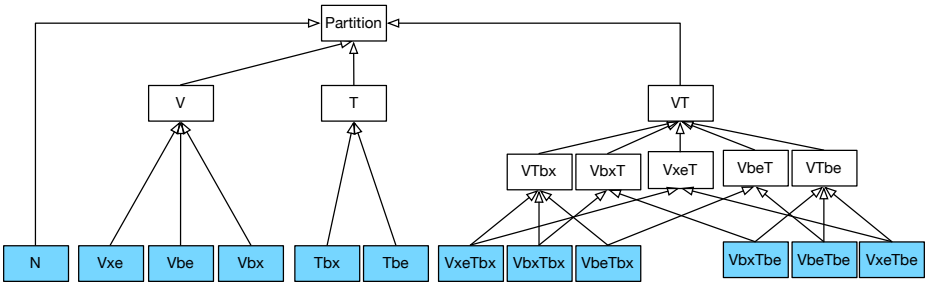


Figure 1. Hierarchy of all potential relational partitions

3.3 Relational partitions

Relparts are conceptually split into "partitions" to facilitate constraint definition and query expressiveness [11]. A partition is defined regarding the relation category and timeline category. Figure 1 shows the hierarchy of potential relational partitions that

[2] A PJ decomposition of a relation R is defined by its contributive projection subsets, each of them containing the common key and the union of them being equals to the attributes of R. Here, the subsets are precisely to the 6NF relparts of R (including the key relpart).

can be found in a historical schema depending on the decomposition. In UBFH, the structure and the constraints are defined over the leaf partitions (in blue).

In the following section a relational partition of R is denoted R@S where S is one of the categories illustrated in the previous figure (white and blue boxes). We may also refer to a specified grouping in a specific partition denoted R_g@S.

4 Constraints

This section presents constraint specifications that are automatically generated for a specified relation. Constraints are defined on V partitions because they can be modified by the user. No constraints are defined for T partitions because they cannot be modified by the user as they are managed by the DBMS using any of the following solutions: (a) as views when the values of the transaction timeline attribute are obtained by a function call to the DBMS journal as in DDLM or (b) as base relations (table) when the values of the transaction timeline attribute are set by the DBMS (as in SQL:2011 with SYSTEM TIME), or by triggers (as in BCDM [14]). In UBHF, all solutions are supported if they satisfy the transaction timeline semantic.

The following constraints must be defined regarding each relation of the initial schema. A constraint is defined with a unique identifier and a boolean expression.

4.1 Candidate keys

For each grouping R_g in partition S{N, Vxe, Vbx, Tbx, VxeTbx, VbxTbx}, the key constraint is the same as the initial relation. The constraint of R_g@S is:

```
RELATION R_g@S (K, B, C, Dv, Dt)  KEY {K};
```

For historicized relparts with be-type timeline, the key constraint must be applied to each time point in the period. For each grouping R_g in partition S{Vbe,VbeTbx}, add:

```
RELATION R_g@S (K, B, C, Dv, Dt)  USING(@Vbe): KEY {K, @Vbe};
```

For each grouping R_g in partition S{Tbe, VbxTbe, VxeTbe}, add:

```
RELATION R_g@S (K, B, C, Dv, Dt)  USING(@Tbe): KEY {K, @Tbe};
```

For each grouping R_g in partition S{VbeTbe}, add:

```
RELATION R_g@S (K, B, C, Dv, Dt)  USING(@Vbe, @Tbe): KEY {K, @Vbe, @Tbe};
```

4.2 Temporal denseness

The temporal denseness is defined over all relparts of an initial relation to ensure the history completeness. Each initial relation R has been decomposed in one K-grouping and some B- and C-groupings by historicization. Those groupings must stay consistent, i.e.: each key defined in the K-relpart must also be defined in the same period in one of its related present or missing relpart of B- and C-groupings. This constraint is inspired by the requirements 3 and 6 in [7] and by the equality dependency defined in [6].

First, a shorthand operator, **gSpace**, defined in [11], is extended to cover Vxe partition. The operator extracts the history (if any) of a specified grouping by calculating the union of all relparts of a grouping R_g with respect to the applicable partition[3] (@N or @V):

```
OPERATOR gSpace(R_g GROUPING) RETURNS RELATION;
  IF R_g is a C-grouping THEN
    R_g@N
  ELSE // R_g is a K-grouping or a B-grouping
    WITH (
      r_xe := (EXTEND R_g@Vxe : {@V:=[saw:@Vxe]}) {ALL BUT @Vxe}),
      r_bx := (EXTEND R_g@Vbx : {@V:=[@Vbx:ufn]}) {ALL BUT @Vbx}),
      r_be := R_g@Vbe RENAME {@Vbe AS @V}
    ): USING (@V): r_xe UNION r_bx UNION r_be
  END IF
END OPERATOR
```

The temporal denseness may now be expressed using the two following rules:

For each b_i-grouping, the constraint is defined as follows:

```
CONSTRAINT R_b_i_denseness
 USING(@V): gSpace(R_K) = gSpace(R_b_i){K,@V}
```

For each c_i-grouping, the constraint is defined as follows:

```
CONSTRAINT R_c_i_denseness
  gSpace(R_K){K} = gSpace(R_c_i){K}
```

4.3 Foreign keys

A foreign key is evaluated regarding related attributes in different relparts. In a historicized relation, the associated timeline attributes must be considered to guarantee that their related values are asserted at the same time (at each moment of the unpacked relation). On the one hand, the foreign key in the initial schema must be maintained. Let Rs{X}→Rd be a foreign key in the initial schema where Rs is the source relation with X being any subset of attributes of Rs equivalent to the key (K) of Rd, the destination relation. The constraint guarantees temporal referential consistency between groupings by verifying that the projection of Rs on X is included in the projection of Rd on K (with suitable renaming). Using gSpace, another shorthand operator, **gUnpack**, is defined, extracting the unpacked history of an attribute x of a specified relation R:

```
OPERATOR gUnpack(R RelationName, x AttributeName) RETURNS RELATION;
  IF (x in K of R) THEN    UNPACK (@V): (gSpace(R_K){x,@V})
  ELSIF (x in B of R) THEN UNPACK (@V): (gSpace(R_x){x,@V})
  ELSE                     UNPACK (@V): gSpace(R_x){x} JOIN gSpace(R_K)
  END IF
END OPERATOR
```

Each foreign key Rs{X}→Rd induce the following constraint:

```
CONSTRAINT Rs_Rd_@V_fk
  gUnpack(Rs, x_1) RENAME {x_1 AS k_1} ⊆ gUnpack(Rd_K, k_1)
  AND … AND
  gUnpack(Rs, x_n) RENAME {x_n AS k_n} ⊆ gUnpack(Rd_K, k_n);
```

[3] gSpace is defined to deal with @T and @VT partitions. In this paper only @N and @V are represented.

4.4 Key history uniqueness

The key history uniqueness is defined over the value of the timeline associated with K. It ensures non-redundancy and non-contradiction of the key attributes values over time. In other words, it ensures consistency of the history of a tuple by verifying that the same proposition is represented only once. This constraint is similar to requirement 1 in [7]. For the K-relpart, the following constraint must be applied:

```
CONSTRAINT R_K_history
  IS_EMPTY (R_K@Vbx JOIN R_K@Vbe WHERE @Vbx < POST(@Vbe)) AND
  IS_EMPTY (R_K@Vbe JOIN R_K@Vxe WHERE @Vxe > PRE(@Vbe)) AND
  IS_EMPTY (R_K@Vxe JOIN R_K@Vbx WHERE @Vxe > PRE(@Vbx));
```

4.5 Attribute history uniqueness

The attribute history uniqueness is defined over the value of the timeline attribute associated with a b_i attribute. It ensures non-redundancy and non-contradiction of b_i values over time. This constraint is similar to requirement 4 in [7]. For each grouping R_g in $\{R\_b_1, \ldots, R\_b_n\}$, the following constraint must be applied:

```
CONSTRAINT R_g_uniqueness
  IS_EMPTY (R_g@Vbx {K, @Vbx} JOIN R_g@Vbe {K, @Vbe}
    WHERE @Vbx < POST(@Vbe)) AND
  IS_EMPTY (R_g@Vbe {K, @Vbe} JOIN R_g@Vxe {K, @Vxe}
    WHERE @Vxe > PRE(@Vbe)) AND
  IS_EMPTY (R_g@Vxe {K, @Vxe} JOIN R_g@Vbx {K, @Vbx}
    WHERE @Vxe > PRE(@Vbx));
```

4.6 Key history non-circumlocution

The key history non-circumlocution is defined over the value of the timeline attribute associated with K. It ensures non-circumlocution of the key attributes values over time. This constraint is similar to requirement 2 in [7]. For the K-relpart, the following constraint must be applied:

```
CONSTRAINT R_K_circumlocution
  IS_EMPTY (R_K@Vbx JOIN R_K@Vbe WHERE @Vbx = POST(@Vbe)) AND
  IS_EMPTY (R_K@Vbe JOIN R_K@Vxe WHERE @Vxe = PRE(@Vbe)) AND
  IS_EMPTY (R_K@Vxe JOIN R_K@Vbx WHERE @Vxe = PRE(@Vbx));
```

The two key history constraints (uniqueness and non-circumlocution) may be combined in one.

```
CONSTRAINT R_K_key_history
  IS_EMPTY (R_K@Vbx JOIN R_K@Vbe WHERE @Vbx <= POST(@Vbe)) AND
  IS_EMPTY (R_K@Vbe JOIN R_K@Vxe WHERE @Vxe >= PRE(@Vbe)) AND
  IS_EMPTY (R_K@Vxe JOIN R_K@Vbx WHERE @Vxe >= PRE(@Vbx));
```

4.7 Attribute history non-circumlocution

The attribute history non-circumlocution is defined over the value of the timeline attribute associated to a b_i attribute. It ensures non-circumlocution of b_i values over time. This constraint is similar to requirement 5 in [7]. For each present value relpart R_b_i, the following constraint must be applied:

```
CONSTRAINT R_bi_circumlocution
  IS_EMPTY (R_bi@Vbx {K, bi, @Vbx} JOIN R_bi@Vbe {K, bi, @Vbe}
    WHERE @Vbx = POST(@Vbe)) AND
  IS_EMPTY (R_bi@Vbe {K, bi, @Vbe} JOIN R_bi@Vxe {K, bi, @Vxe}
    WHERE @Vxe = PRE(@Vbe)) AND
  IS_EMPTY (R_bi@Vxe {K, bi, @Vxe} JOIN R_bi@Vbx {K, bi, @Vbx}
    WHERE @Vxe = PRE(@Vbx));
```

5 Conclusion

Most proposed DW design methods define transformation rules "by-example" and must largely be adapted and applied manually. More specifically, regarding past indeterminacy, some DW design methods propose "ideas" but none presents a completely integrated deterministic process. UBHF defines (a) relation, attribute and timeline categorization to provide unique semantic; (b) a unified temporal structure and general constraints to be independent of the domain and context (yet providing formal definition and superior automation capabilities); (c) historicization processes with past indeterminacy ensuring traceability over the transformation steps without losing the initial conceptual view.

A model based on UBHF satisfies (1) data integrity with the definition of the described constraints (2), sound temporal schema design using relational concepts and well-defined temporal concepts (3) data schema evolution due to the normalization in 6NF (4) schema traceability and guided automation. This paper has extended UBHF which is now suitable to guided automated historicization of a database schema including past indeterminacy. This will enable improved modelling and query possibilities in many domains, especially in healthcare.

Despite all UBHF concepts, the DW schema still not "fully" historicized. In other words, adding the past indeterminacy induces a fourth timeline category, denoted eb, that represents events that occur "certainly" at some period (from e to b) but the beginning and the end point are not known. This situation may appear when two tuples having the same a_i values and two different timelines xe and bx that merge.

Future work is required to offer a fully applicative solution:

- To add the eb-type timeline.
- To model missing information.
- To implement a tool for designing historicized schema and translating it into standard SQL code or TutorialD code so existing DBMS may be used directly.
- To evaluate UBFH approach in real applications.
- To propose a generalized set of data modification operators (insert, delete, update).

References

1. Allen, J.F.: Maintaining Knowledge About Temporal Intervals. Commun ACM. 26, 11, 832–843 (1983).
2. Anselma, L., Piovesan, L., Terenziani, P.: A 1NF temporal relational model and algebra coping with valid-time temporal indeterminacy. J. Intell. Inf. Syst. 1–30 (2015).

3. Anselma, L., Terenziani, P., Snodgrass, R.T.: Valid-Time Indeterminacy in Temporal Relational Databases: Semantics and Representations. IEEE Trans. Knowl. Data Eng. 25, 12, 2880–2894 (2013).
4. Arora, S.: A comparative study on temporal database models: A survey. 2015 International Symposium on Advanced Computing and Communication (ISACC). pp. 161–167 (2015).
5. Darwen, H., Date, C.J.: The Third Manifesto. SIGMOD Rec. 24, 1, 39–49 (1995).
6. Date, C.J., Darwen, H.: Database Explorations: essays on the Third Manifesto and related topics. Trafford Publishing (2010).
7. Date, C.J., Darwen, H., Lorentzos, N.A.: Time and relational theory: temporal databases in the relational model and SQL. Morgan Kaufmann, Waltham, MA (2014).
8. Jensen, C.S., Dyreson, C.E., Bohlen, M., Cliford, J., Elmasri, R., Gadia, S.K., Grandi, F., Hayes, P., Jajodia, S., Kafer, W., Kline, N., Lorentzos, N., Mitsopoulos, Y., Montanari, A., Nonen, D., Peressi, E., Pernici, B., Roddick, J.F., Sarda, N.L., Scalas, M.R., Segev, A., Snodgrass, R.T., Soo, M.D., Tansel, A., Tiberio, P., Wiederhold, G.: The consensus glossary of temporal database concepts-February 1998 version. Proceedings of Seminar Temporal Databases: Research and Practice, 23-27 June 1997. pp. 367–405 Springer-Verlag (1998).
9. Jensen, C.S., Soo, M.D., Snodgrass, R.T.: Unifying Temporal Data Models via a Conceptual Model. Inf. Syst. 19, 513–547 (1993).
10. Khnaisser, C.: Méthode de construction d'entrepôt de données temporalisé pour un système informationnel de santé. Faculté des sciences, Université de Sherbrooke (2016).
11. Khnaisser, C., Lavoie, L., Burgun, A., Ethier, J.-F.: Unified Bitemporal Historicization Framework. Université de Sherbrooke (GRIIS), Sherbrooke, Québec, Canada (2017).
12. Khnaisser, C., Lavoie, L., Diab, H., Ethier, J.-F.: Data Warehouse Design Methods Review: Trends, Challenges and Future Directions for the Healthcare Domain. In: Morzy, T., Valduriez, P., and Bellatreche, L. (eds.) New Trends in Databases and Information Systems. pp. 76–87 Springer International Publishing (2015).
13. Rizzi, S., Abello, A., Lechtenborger, J., Trujillo, J.: Research in data warehouse modeling and design: Dead or alive? 9th ACM International Workshop on Data Warehousing and OLAP - DOLAP'06. pp. 3–10 Association for Computing Machinery, New York, NY, USA (2006).
14. Snodgrass, R.T.: Developing time-oriented database applications in SQL. Morgan Kaufmann Publishers, San Francisco, California (2000).

Annexe 5
Combining ontology and temporal databases for data reuse: the example of hospital organizational structures

Article soumis à JAMIA le 13 aout 2019. Suite aux commentaires reçus, à resoumettre en octobre 2019 à IJMI.

Combining ontology and temporal databases for data reuse: the example of hospital organizational structures

Christina Khnaisser[1,2], Vincent Looten[1,3], Jean-François Ethier[1,2], Luc Lavoie[2], Anita Burgun[1,3]

[1] Université Paris Descartes, Centre de Recherche des Cordeliers, INSERM, URMS1138,Team 22, Paris, France

[2] Université de Sherbrooke, Groupe de recherche interdisciplinaire en informatique de la santé, Sherbrooke, Canada

[3] Hôpital Universitaire Européen Georges-Pompidou - Assistance Publique-Hôpitaux de Paris, Paris, France

Corresponding author : jean-francois.ethier@usherbrooke.ca Faculté de médecine et des sciences de la santé, Département de médecine +1 819 821-8000 #74977; 3001, 12e Avenue Nord Sherbrooke (Québec) J1H 5N4

Keywords: Temporal Database, Ontology, Hospital Administration, Health Information Interoperability.

Word count (excluding title page, abstract, references, figures and tables): 4330

ABSTRACT (250 MOTS)

Objective

Healthcare data need to be semantically defined and stored in a way that ensures data evolution, traceability and the accuracy of query results. Our objective is to link ontologies and the temporal relational database to take advantage of the ontologies expressivity richness and the relational database maturity.

Materials and Methods

We propose a generic method that generates a temporal relational database from an ontology to store the data in a way that preserves data integrity, semantic and temporal data evolution. To demonstrate the usability of the method, the hospital organizational structure is defined using an OWL ontology from which a temporal relational database is generated. Then, the data quality and the potential impact of

organizational changes in healthcare activities and data access rights is illustrated with two use cases.

Results

The use cases highlighted the impact of the evolution of structures on the secondary use of healthcare data and data policy access. The analysis of the data quality and the evolution of organizational structures over a long period of time and across multiple hospitals has been made easier combining an ontology that describes a formal representation of data and the temporal relational database that tracks the evolution of this data. This work is a contribution towards a better data stewardship and data reuse.

Conclusion

The paper demonstrated the usefulness of the ontology to provide a formal, interoperable and reusable definition of entities and their relationships and the adequacy of the temporal database to store, trace and query data over time.

INTRODUCTION

Real world data collected during health care are a key component of learning health systems [1]. However, data within various institutions are managed by different systems using various data structures and semantics [2]. In order to integrate and access such heterogeneous data in a semantically interoperable sound way, a unified logical data model may be defined using a knowledge model [3] and a temporal model to ensure data consistency over time. More precisely, ontologies can be used to provide formal representations of the relevant entities and their relations. While significant research efforts in the medical informatics community have produced shared ontologies of diseases, procedures, and other medical entities [4–6], less has been done regarding hospital organizational structures. Moreover, a temporal model is need to keep track of changes of these structures over time as, for example, services are merged, new departments are created or hospital are united [7].

Indeed, temporal data are acquiring importance in the healthcare field [8]: sound temporal model plays an essential role in minimizing temporal indeterminacy and improving query expressiveness. However, many challenges remain in modelling time in ontologies [9] and storing temporal data when ontologies are used on a large scale [10,11]. On the other hand, temporal data models relying on relational databases exist. While some methods rely on *ad hoc* models and may have more limited scopes and carry interoperability limitations [12]. Robust ones have been proposed [13,14] and offer design guidelines as well as constraints regarding temporal representations.

Given the scope, wide perimeter and unpredictability of possible uses cases to be supported in context of healthcare data reuse for research, management and care delivery, it becomes essential to have a temporal model which stands on its own, provides intrinsic computability soundness, and gives transformation rules automation independently of a specific domain. This paper therefore proposes a generic method that generates a temporal relational database (TRDB) to store the data and track their evolution in a way that preserves data semantics and data integrity according to a domain ontology (that formalizes concepts of a specific domain).

The method was applied to changes of hospital organizational structures across time. The rationale for choosing such an application was twofold. First, many studies in health services research rely on clinical data warehouses [15]. These studies evaluate among other things the effectiveness of clinical procedures, healthcare processes and management of healthcare facilities [16]. All these studies require fine-grained organizational information rather than hospital-level analysis [17,18] to be able to compare outcomes and costs associated with practice patterns, variations in care processes and organization [19,20]. Secondly, access, sharing and reuse of health data for research purposes are subject to strict rules. Data access policies are based either on organization-level, on patient-level, or on a combination of these. Many hospitals in Europe have adopted standard data access policies based on the organizational structures. By default, doctors are allowed to access and reuse the data of all patients that have been admitted in their department; the services being part of a department will vary through time. At patient-level, the data access by project is adopted by a data warehouse in the United States [21] and in Europe [15]. At organization-level, the data access is granted by certified structures or an ethics committee [22]. As a result, for a retrospective study, such rules cannot be implemented without a deep understanding of hospital organizational structures and their relations across time [15]. This is largely done with *ad hoc* methods, sometimes even manually. These two use cases, namely (1) variations in care processes, and (2) application of data access rules for retrospective data, were chosen to illustrate our approach.

RELATED WORK

Data reuse takes advantage of solutions based on mediation systems [23] and data warehouses (like i2b2 [24]) that integrate various kinds of information that are routinely generated during patient care. These systems may take advantage of the numerous ontologies developed in the medical domain [25] to generate more explicit semantics of the data. However, they do not use a temporal data model uniformly across facts. Thus, finding the right data, tracing data evolution and understanding its semantics is still a difficult task. On one hand, temporalizing current ontologies requires substantial work, decreases reasoning efficiency and may introduce semantic errors due to its complexity [26,11,9]. On the other hand, designing and

implementing a temporalized data model require multiple manual steps and a generalized temporalization process has not been developed until now [27,28].

Time representation and databases

One of the most influential publications on the representation of temporal primitive concepts and temporal relations was Allen's interval logic [29]. This work defined twelve relations (*before/after, meets/met_by, overlaps/overlapped_by, starts/started_by, during/contains, finishes/finished_by*) and one symmetric relation (equal) between temporal intervals, as well as a series of axioms to hold between these relations. Following that work, time representation has been extensively addressed in many disciplines such as databases, semantic web, natural language processing, and artificial intelligence [7].

Several ontologies of time have been defined, including: (1) the *OWL-Time ontology* developed by the World Wide Web Consortium (W3C) [30] (defines time primitive concept taxonomy for expressing facts about relations between instants and intervals, along with information about duration, and temporal position including date-time information), (2) the *SWRL Temporal Ontology* [31] and SOWL [32] (define various approaches to represent temporal concepts; use SWRL rules to enable reasoning features), and (3) the *Clinical Narrative Temporal Relation Ontology* [33] field and *TimeML* [34] (for events and temporal expressions in natural language). However, their usability in a practical context and scalability in large-scale applications are still to be demonstrated [35]. Moreover, a cross-disciplinary effort is required to harmonize the representations [9] and make them work in a practical context.

In the database domain, several models and temporal languages have been defined since 1970 in order to simplify time management in a TRDB [36]. The two predominant models are the *Bitemporal Conceptual Data Model* [13] based on SQL and the *Date, Darwen and Lorentzos Model* [14] based on the relational theory and temporal algebra using interval logic [37]. A more recent work of Khnaisser *et al.* [28] proposes a *Unified Historicization Framework* (UHF) that combines the models of BDCM and DDLM to enable automation [28,38,39]. UHF is a conceptual framework that defines temporal stereotypes based on fundamental relational and temporal concepts to provide a uniform and systematic approach to represent temporal data in the context of a relational database. However, an automatized process is needed to take advantage of the ontology and UHF to generate an axiomatic temporal data schema.

Organizational structures and databases

Ontologies for administrative data and especially organizational structures of healthcare are rare [40] or very specific. Some related concepts appear in the *Ontology of organizational structures of trauma centers and trauma systems* [41] that describes a trauma center structure and the *Ontologized minimum information about biobank data sharing* [40] that describes a biobank structure. Nevertheless, a

reference ontology describing generic organizational structure, the *Organization Ontology* (ORG) [42], has been developed as part of W3C recommendation. The ORG ontology contains upper-level concepts related to organizations, but, to our knowledge, has not been tailored to represent hospital organizational structures.

Besides ontology design, important research has focused on methods that translate ontologies into database models [12,36]. The same approach can be applied to organizational structures. In addition, administrative structures frequently change over time, e.g., a medicine department may include a unit for children and a unit for adults, and in the next decade be restricted to adult care only. Such changes have to be traced in the databases to improve the quality of retrospective studies, prevent false conclusions to be drawn and give proper access (or not) to the data. To achieve that goal, the model must integrate time aspects along with data modeling. Noticeably, although the literature on ontology-based generation of data models is abundant, none of these articles [12,36] report on generating a TRDB.

Objective

In this article, we propose a method to generate a TRDB for tracing organizational changes in healthcare facilities via a linkage between ontologies and the relational model to take advantage of the ontology expressiveness and the relational databases maturity. To our knowledge, our paper is the first that proposes a reusable method to generate an ontology-based TRDB. We implemented our method on a use case and a set of data from the *Assistance Publique-Hôpitaux de Paris* (AP-HP), the French public health institution located in Paris, France.

MATERIALS AND METHODS

Designing an ontology of the hospital organizational structures

The organizational structure in French hospitals (Figure 1) is based on the principles of the French national management accounting for hospitals [43] and is linked to the Program of Medicalization of Information Systems, a legacy system based on the diagnosis related groups that is used for billing in all French hospitals [44].

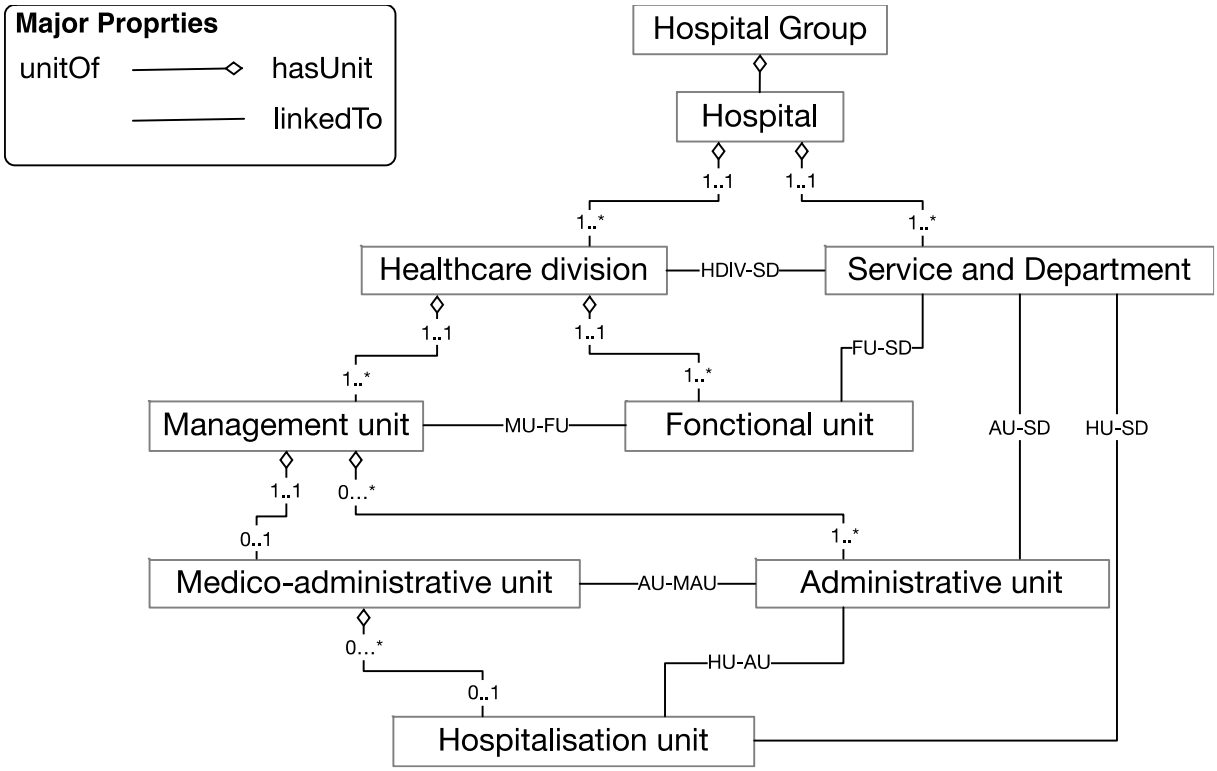


Figure 1. Conceptual view of organizational structures in French hospitals. The hospital organizational structure has two elementary structures: the hospitalization unit (HU) and the functional unit (FU). The HU has an administrative responsibility scope, and the FU has a medical responsibility scope. The other levels are aggregates of HU or FU elements according to different perspectives (administrative, medical or mixed). Three main properties were used to relate the defined units: 'unitOf' and 'hasUnit' (inverse of 'unitOf') represent the hierarchy between units and 'linked to' represents a link between two units.

For the purpose of this demonstration, we designed an ontology to provide a unified formal definition of such hospital organizational data [45]. As the ORG ontology was developed to serve as a reference ontology that could be tailored to any specific application, it was used as a starting point to design the *Hospital ORGanization Ontology in France* (HORGO-FR).

Thus, HORGO-FR ontology was designed according to the following principles. HORGO-FR is formalized in OWL and provides a formal definition of the organizational structures enabling reasoning about these structures. HORGO-FR is created via the reuse of the ORG ontology [42] that describes the core concepts of organizational structures (e.g. 'formal organization', 'organizational unit', 'membership', etc.), and was developed to support the sharing of organizational data across a number of domains. The reference ontology is extended and refined to encompass entities and relations that are specific to hospitals and even specific to the French hospitals if needed. HORGO-FR mainly extended 'formal organization' and 'organizational unit' to define their descendants, i.e., the Hospital and its units, respectively. The hierarchy of units is described using the object properties 'unitOf'

and 'hasUnit', and the more specific links are described using the 'linkedTo' object properties. Moreover, some classes were added to describe items such as 'unit identification' (described as a 'unit identifier' and a 'unit label'), 'medical specialty' associated with a 'functional unit'.

Creating the temporal relational database

The method consists of four steps (Figure 2): (1) generating a temporal relational schema from an ontology; (2) building the TRDB; (3) feeding the TRDB from the data sources; and (4) exploring and extracting for analysis the data loaded in the TRDB.

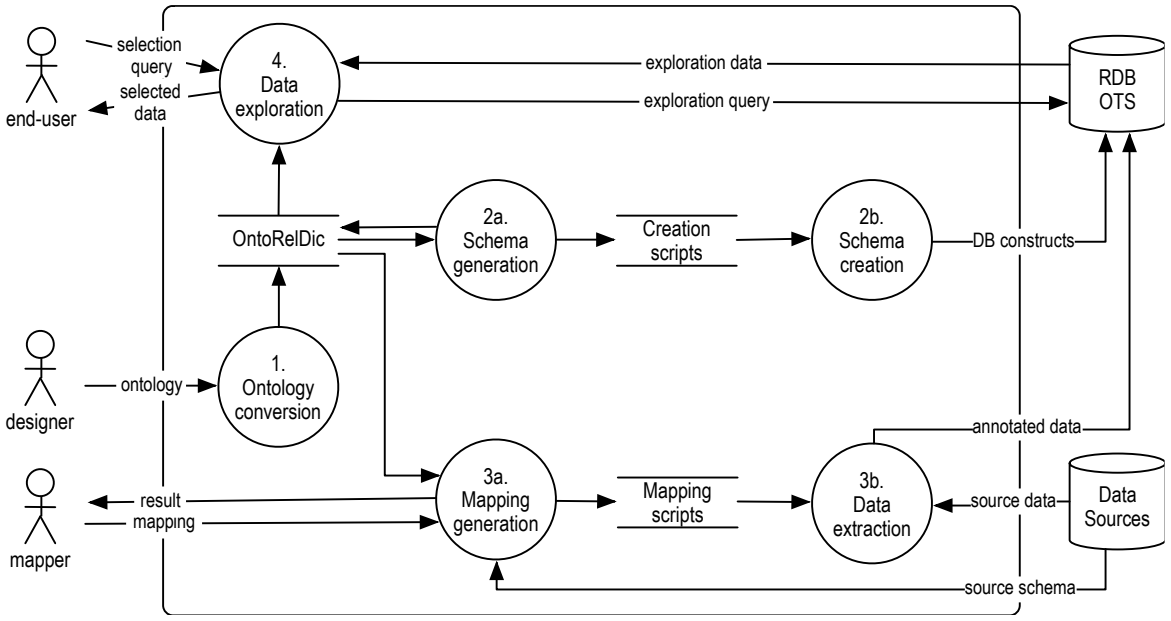
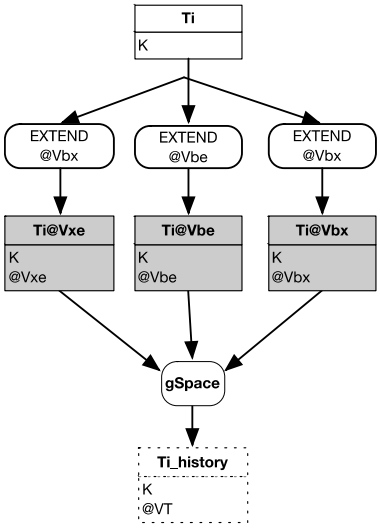


Figure 2. **Data flow diagram of the method**. The method includes 4 steps: (1) The ontology conversion process generates from an ontology a schema dictionary (OntoRelDic) according to ontological-relational conversion rules. The OntoRelDic describes the mapping between an ontological construct and the related relational constructs; (2) The TRDB construction includes 2 steps (2a) the schema generation and (2b) the schema creation. The schema generation process temporalizes the relation constructs (tables and constraints) according to the UHF and generates a set of SQL schema creation scripts of the relational constructs defined in OntoRelDic. The schema creation process executes the creation scripts in the RDB to create the ontology-temporal schema (OTS); (3) The TRDB feeding includes 2 steps (3a) mapping generation and (3b) the data extraction. The mapping generation process generates a set of SQL mapping scripts using OntoRelDic and data sources. The data extraction populates the OTS with annotated data from the data sources; (4) The data exploration process offers an interface to extract and explore data through the ontology (via OntoRelDic) stored in the OTS.

First, a set of conversion rules have been defined for the ontology to convert the ontological constructs into relational constructs, based on our previous work [46]. Then, the temporalization of relational constructs is done according to UHF [28]. UHF

defines deterministic conversion rules based on fundamental relational and temporal concepts. The temporalization uses a discrete time model based on time points and intervals [begin point..end point] derived from the one defined by Allen [47] and a timeline called the valid time that represents the period of validity of a fact in the modeled reality [48]. The temporalization (Figure 3) consists of generating from each table (Ti) three temporal tables with temporal constraints and a "*history view* ".



Legend
Ti: a table representing a class or an axiom
K: set of attribute forming the key
EXTEND: relational operator that adds a named attribute to a table initialized with an expression
Ti@Vxe: until table
Ti@Vbe: during table
Ti@Vbx: since table
Ti_history: history table
gSpace : relational operator that merges the 3 temporal tables of a same table.

```
OPERATOR gSpace(Ti RELATION) RETURNS RELATION;
    WITH (
      t_xe := (EXTEND Ti@Vxe : {@VT:=[faw..@Vxe]}) {ALL BUT @Vxe}),
      t_bx := (EXTEND Ti@Vbx : {@VT:=[@Vbx..ufn}) {ALL BUT @Vbx}),
      t_be := Ti@Vbe RENAME {@Vbe AS @VT}
    ): USING (@VT): t_xe UNION t_bx UNION t_be
END OPERATOR
```

Figure 3. Illustration of the generic temporalization process of a table. A non-temporal table is temporalized by generating three temporal tables, one for each valid time category. (1) Ti@Vbe *During table* is the extension of Ti table with a valid time interval attribute (@Vbe) where the beginning and the end point values are known. Ti@Vbe represents propositions that are valid during a period of time (the proposition is considered valid for each point included in the period); (2) Ti@Vbx *Since table* is the extension of Ti table with a valid time begin point attribute @Vbx = [*since*..*ufn*] where *since* is the known begin point of a period and *ufn* (until further notice) is the unknown end point of a period. Ti@Vbx represents propositions that are valid since a point in time; (3) Ti@Vxe *Until table* is the extension of Ti table with a valid time end point attribute @Vxe = [*faw*..*until*] where *faw* (from a while) is the unknown begin point and *until* is the known end point. Ti@Vxe represents propositions that are until a point in time. Ti_history the temporal history of Ti is a view that shows all data modification over time of Ti. The view is built over the three temporal table using a gSpace function (PU) where *faw* and *ufn* are parametrized variable.

Furthermore, keeping history changes may introduce redundancy, contradiction, circumlocution and non-denseness when attributes in the same table are modified independently. These problems were studied in detail in [14] and in UHF we described generalized constraints that must be implemented in a relational database to detect them. Thus, for each temporal table four constraints are generated to handle the following problems:

- Redundancy occurs when two value-equivalent tuples overlap in time (Two tuples are value-equivalent when all their non-temporal attributes have the same values).
- Contradiction occurs when two tuples having identical key values but different non-key attributes values overlap in time.
- Circumlocution occurs when two value-equivalent tuples meet in time (one follows immediately the other).
- Denseness guarantees that when an attribute value is known for a key value at some time point, the values of all other attributes which are dependent on the same key are known at the same time point.

Evaluation

The approach was evaluated at AP-HP, a public health establishment with seven affiliated universities which comprises 39 hospitals. Among them, the Georges-Pompidou European Hospital (*Hôpital Européen Georges Pompidou* HEGP) [49] has an i2b2 clinical data warehouse (CDW) that has been in operation since 2009. The CDW contains almost all the EHR data produced in the hospital since 2000 [15], resulting in a total of 1,164,525 patients and 2,830,351 encounters in 2019-01-08. Yet, the data related to the hospital organizational structures and its evolution is not part of the CDW. Thus, the evaluation presented here aims at exploring the possibility of the future integration of clinical and organizational structure data. The approach is evaluated using HORGO-FR and the data of organizational structures extracted from the institutional AP-HP *Sirius database* [50] through the *IBM InfoSphere MDM*(TM) solution which holds the organizational structure data.

Firstly, the quality of organizational structure data over time was validated using a set of 21 validation queries developed according to the ontology axioms. Secondly, two use cases were designed to evaluate the potential impact of organizational changes of the HEGP hospital on healthcare activities and access rights.

Ethics and reproducibility

The use cases were conducted under the methodological guidelines MR-005 of the French national data privacy authority (*CNIL, Délibération n° 2018-256 du 7 juin 2018).* All scripts are available at https://github.com/equipe22/HORGO-FR.

RESULTS

The method generates an ontology-temporal schema for the hospital organizational structure from HORGO-FR. Then a PostgreSQL database is used to store the source data structured using mapping scripts defined according to OntoRelDic.

The HORGO-FR was designed using Protégé and contains 288 definitions (Table 1). Consistency checking was performed using HermiT and Pellet reasoner with success.

Table 1. **Ontology entities counts.** The HORG-FR ontology contains 46 classes, 39 object properties, 6 data properties and 197 logical axioms.

	ORG base	HORGO-FR specific	HORGO-FR total
Nb. Classes	16	30	46
Nb. Object properties	36	3	39
Nb. Data properties	2	4	6
Nb. Logical axioms	110	87	197

The TRDB contains 656 relational constructs (Table 2) and 2985 units with 3529 different relationships. The TRDB offers a coherent axiomatic structure for temporal data aligned with the entities and axioms from the ontology and temporal validation rules using temporal constraints. This structure allows the definition of temporal queries to reconstruct organizational structures for specific periods.

Table 2. Database construct counts. The hospital organizational structure TRDB contains 153 temporalized tables, 70 history views, 80 procedures for data verification and data insertion. On the 5th of January, the TRDB contained 1887 units and 2254 different relationships in HEGP, and 1098 units and 1275 different relationships in *Ambroise-Paré* hospital.

Number of relational constructs	
Nb. Tables	153
Nb. Constraints	343
Nb. Views	70
Nb. Procedures	80
Nb. Functions	10
Number of specific tuples	
Nb. HEGP units	1887
Nb. HEGP different relationships	2254
Nb. *Ambroise-Paré* hospital unit	1098
Nb. *Ambroise-Paré* hospital different relationships	1275

Data quality assessment using ontology axioms and temporal features

The data was validated using a set of validation queries (Table 3). The validation queries are defined according to the ontology axioms using the following properties: "hasUnit", "unitOf" and "linkedTo". Each validation query tests (A) the property restriction of the unit lifetime, and (B) the temporal inclusion between related units. The temporal inclusion test verifies if a sub-unit valid time period is included in the super-unit valid time period using Allen operators (equals, during, begins and finishes).

These tests demonstrated the usefulness of the ontology as a base model to formulate validation queries and a temporal database to verify the restrictions over time. The detected anomalies were used to notify the persons in charge of the reorganization and the data access definition as well as the researchers so that they can take these anomalies into consideration while querying the data. Finally, the

implementation of validation queries in the database could be an interesting way to detect potential errors generated by a reorganization of the hospital structures.

Table 3. Result of the validation queries. A validation query (VQ) is an axiom of HORGO-FR ontology. (A) The count of valid and invalid property restriction. (B) The count of temporal inclusion (inc.) and temporal exclusion (exc.). The results show that the majority of the restrictions on “unit Of” and “has Unit” are respected. For "linkedTo" property, only 3 over 7 validation queries confirm the property restriction. Moreover, a high number of temporal exclusion (non-inclusion) relations is detected. This can be explained by (1) the fact that the data sources are not fully temporalized and (2) the lack of integrity constraints to track the reorganization of units.

VQi	Axiom	A		B	
		#valid	#invalid	#inc.	#exc.
VQ1	'Healthcare division' 'unit Of' exactly 1 'Hospital'	29	0	29	0
VQ2	'Service and department' 'unit Of' exactly 1 'Hospital'	98	0	91	7
VQ3	'Functional unit' 'unit Of' exactly 1 'Healthcare division'	284	0	129	155
VQ4	'Management unit' 'unit Of' exactly 1 'Healthcare division'	632	0	344	288
VQ5	'Medico-administrative unit' 'unit Of' exactly 1 'Management unit'	676	0	485	191
VQ6	'Administrative unit' 'unit Of' some 'Management unit'	736	0	615	121
VQ7	'Hospitalization unit' 'unit Of' exactly 1 'Medico-administrative unit'	808	0	367	441
VQ8	'Hospital' 'has Unit' some 'Healthcare division'	2	0	2	0
VQ9	'Hospital' 'has Unit' some 'Service and department'	2	0	1	1
VQ10	'Healthcare division' 'has Unit' some 'Functional unit',	26	3	14	12
VQ11	'Healthcare division' 'has Unit' some 'Management unit'	27	2	14	13
VQ12	'Management unit' 'has Unit' max 1 'Medico-administrative unit'	498	134	215	151
VQ13	'Management unit' 'has Unit' some 'Administrative unit'	507	125	401	106
VQ14	'Medico-administrative unit' 'has Unit' some 'Hospitalization unit'	625	51	244	381
VQ15	'Service and department' 'linked to' max 1 'Healthcare division'	91	7	16	76
VQ16	'Functional unit' 'linked to' max 1 'Management unit'	157	127	152	470
VQ17	'Functional unit' 'linked to' max 1 'Service and department'	284	0	136	34
VQ18	'Administrative unit' 'linked to' max 1 'Medico-administrative unit'	660	76	516	141
VQ19	'Administrative unit' 'linked to' max 1 'Service and department'	736	0	369	51
VQ20	'Hospitalization unit' 'linked to' exactly 1 'Administrative unit'	778	32	305	473
VQ21	'Hospitalization unit' 'linked to' max 1 'Service and department'	810	0	390	268

Querying organizational structure using ontology axioms and temporal features

We have considered the current approach from two perspectives: data reuse for clinical epidemiology, and data access rights.

The analysis of the evolution of organizational structures and healthcare activities over a long period of time and across multiple hospitals was straightforward. The ontology and the temporal relational database allowed us to construct semantic queries to extract the hierarchical structure over different period.

The epidemiological use case

This use case is focused on the evolution of the organizational structure of the units related to the nutrition specialty at HEGP and *Ambroise Paré* hospital and the prevalence of obese patients in the hospitals.

The results were obtained as following (Figure 4):

1. The organizational changes were extracted from the TRDB using a temporal query over the unit's hierarchy as defined in the ontology. The query builds the hierarchical structure from 2013-01-01 to 2019-01-01 of all functional units having nutrition as the medical specialty for both hospitals. The hierarchy is built according to ontology axioms using "unitOf" and "linkedTo".
2. The data of healthcare activities were extracted from the French hospitalization summary discharges database. The query used all the discharge summaries timestamps of the AP-HP with an E66 (“obesity and overweight”) ICD-10 code between 2013-01-01 to 2019-01-01.

The results show a significant increase of the prevalence of obesity among the HEGP patients after July 2016. Thanks to the temporalized organizational data and the ontology, this phenomenon can be easily interpreted with regard to changes in the organization: new units specialized in nutrition opened at HEGP in July 2016 and, at the same time, changes were made in the *Ambroise Paré* hospital (Figure 5B), but in the latter the activity pertaining to obesity remained stable despite the changes in its structure. Actually, these changes correspond to the transfer of the head of the department of nutrition from the *Ambroise Paré* hospital to HEGP in July 2016. This example demonstrates that (1) data interpretation requires that the organizational structure history be integrated in the CDW, (2) storing the data in a TRDB provides an adequate solution for automated queries that take into account time aspects, e.g., search for all units that exhibit a significant increase of the *prevalence of some ICD code* and any organizational change at some *period.*

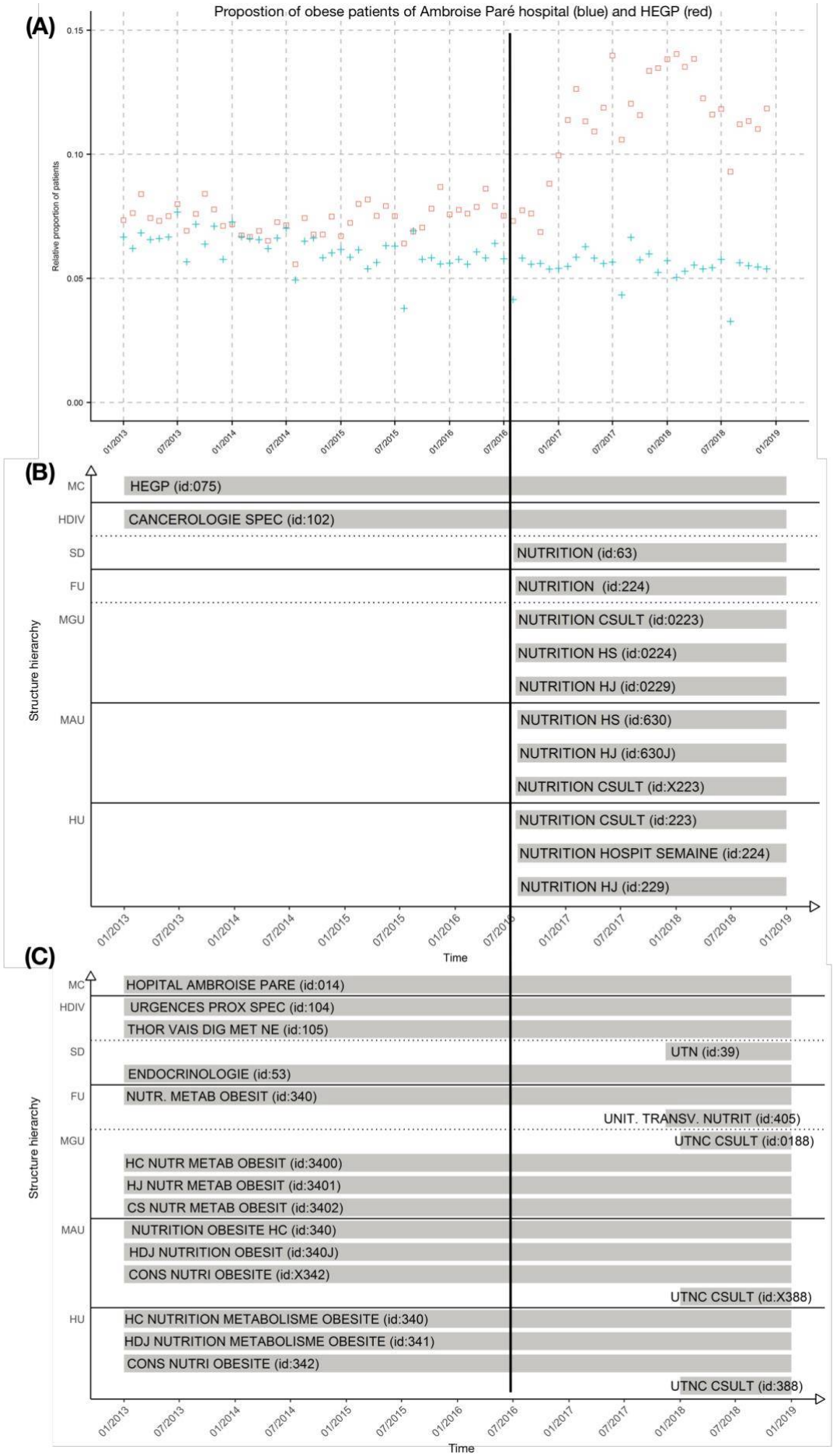

(A)
Propostion of obese patients of Ambroise Paré hospital (blue) and HEGP (red)
Relative proportion of patients
0.15
0.10
0.05
0.00
01/2013
07/2013
01/2014
07/2014
01/2015
07/2015
01/2016
07/2016
01/2017
07/2017
01/2018
07/2018
01/2019
(B)
Structure hierarchy
MC
HDIV
SD
FU
MGU
MAU
HU
HEGP (id:075)
CANCEROLOGIE SPEC (id:102)
NUTRITION (id:63)
NUTRITION (id:224)
NUTRITION CSULT (id:0223)
NUTRITION HS (id:0224)
NUTRITION HJ (id:0229)
NUTRITION HS (id:630)
NUTRITION HJ (id:630J)
NUTRITION CSULT (id:X223)
NUTRITION CSULT (id:223)
NUTRITION HOSPIT SEMAINE (id:224)
NUTRITION HJ (id:229)
Time
(C)
Structure hierarchy
MC
HDIV
SD
FU
MGU
MAU
HU
HOPITAL AMBROISE PARE (id:014)
URGENCES PROX SPEC (id:104)
THOR VAIS DIG MET NE (id:105)
UTN (id:39)
ENDOCRINOLOGIE (id:53)
NUTR. METAB OBESIT (id:340)
UNIT. TRANSV. NUTRIT (id:405)
UTNC CSULT (id:0188)
HC NUTR METAB OBESIT (id:3400)
HJ NUTR METAB OBESIT (id:3401)
CS NUTR METAB OBESIT (id:3402)
NUTRITION OBESITE HC (id:340)
HDJ NUTRITION OBESIT (id:340J)
CONS NUTRI OBESITE (id:X342)
UTNC CSULT (id:X388)
HC NUTRITION METABOLISME OBESITE (id:340)
HDJ NUTRITION METABOLISME OBESITE (id:341)
CONS NUTRI OBESITE (id:342)
UTNC CSULT (id:388)
Time

Figure 4. Evolution of healthcare activity and structures related to obese patients in the *Ambroise Paré* hospital and HEGP. Proportion of obese patients of the *Ambroise Paré* hospital (blue) and HEGP (red). **(A)** The relative proportion of obese patients per month was computed as the ratio between the number of obese patients seen in HEGP (resp. *Ambroise Paré hospital*) and the total number of obese patients seen at AP-HP. The results contain respectively 16,899 and 26,212 hospitalization stays from the *Ambroise Paré* hospital and the HEGP with the E66 code corresponding to 8,087 and 16,921 distinct patients (resp.). Six months after the creation of nutrition units (black transversal line), we observed a temporal shift of the relative proportion of patients in the HEGP's activity (from 9% to 12%). In the *Ambroise Paré* hospital, the activity remained stable.**(B)** The evolution of the organizational structure related to the nutrition in the HEGP. **(C)** The evolution of the organizational structure related to the nutrition in the *Ambroise Paré* hospital. Of note, while it may seem that new units are appearing also at *Ambrois Paré* hospital at the beginning of 2018, UTN units ("Unité Transversale de Nutrition" in French) are transversal units created to improve the quality of care by proposing to others specialties of *Ambroise Paré* hospital advices of nutrition specialists, hence the lack of increased activity with their creation as they do not represent hospitalizations.

The access right use case

This use case reports on the organizational changes in cardiovascular surgery and reanimation divisions at HEGP for a seven-year period.

The results were obtained as follows:

1. The organizational changes were extracted from the TRDB using a temporal query over the unit's hierarchy defined in the ontology. The query builds the hierarchical structure of the cardiovascular surgery department and the anesthesia & resuscitation department (query results are presented in appendix).
2. The number of stays in four specific hospitalization units (HU) was calculated based on the CDW. Two new HU were opened: HU 681 and HU 682, and two HU were "physically" closed, HU 339 and HU 331. However, the changes were neither recorded as such nor documented in the system and so were considered open in the system.

The data visualization of the query results (Figure 5) reveals a potential transfer of activities between HUs from 2018-08-01 to 2019-02-01. Thus, this information was only available in the human mind and was limited to some administrative staff responsible for maintaining the organizational structure data set. This situation has an impact on automated management of data access rights. First, if the structure data was correctly updated, adding patient stays after 2018-09-01 should have been refused. Second, the cardiovascular surgery and the reanimation team members working in HUs 681 or 682 could neither access nor reuse the EHR data generated in HU 339 and HU 331 before 2018-09-01. This use case illustrates that, if the history

and the hierarchy of the organization structures are not explicitly available, then a data access policy based on unit hierarchy cannot be correctly implemented.

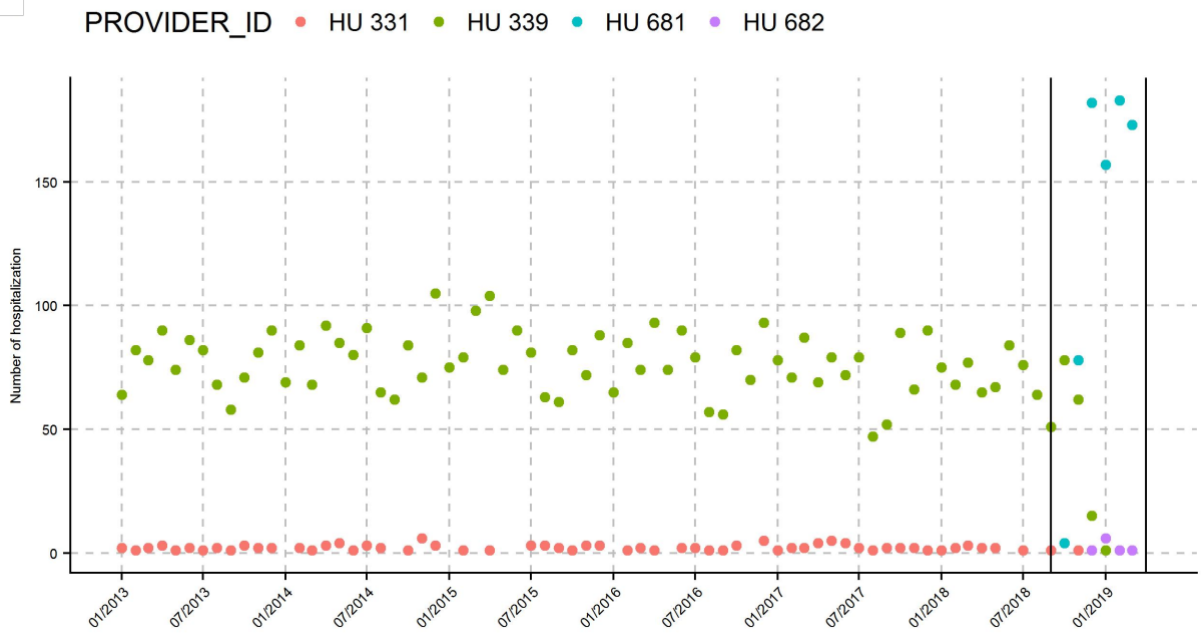


Figure 5. Evolution of activity related to four reanimation units in HEGP and the cardiology and the anesthesia service and department structure. The evolution of the number of hospitalizations in HEGP hospitals for the HU 331 (surgical reanimation unit for transplantation), HU 339 (cardiovascular surgical reanimation unit), HU 681 (general surgical reanimation unit) and HU 682 (surgical reanimation unit for transplantation). The data visualization of the query results reveals a potential transfer of activities from HU 331 to HU 682 and HU 339 to HU 681 from 2018-08-01 to 2019-02-01. This information was confirmed by the administrative staff responsible for maintaining the organizational structure data set. However, notice that, the number of hospitalizations in HU 331 and HU 339 was still changing several months after the structure changes, so data access to these units must be adapted consequently to avoid hospitalization to be recorded in the wrong units.

This is one among many use cases that demonstrates that the clinical data must be described with rich metadata, including the complex relations between organizational structures across time.

DISCUSSION

Recently developed guidelines for improved data reusability [51] that are referred to as the FAIR Data Principles (Findability, Accessibility, Interoperability, and Reusability) put specific emphasis on enhancing the ability of machines to automatically find and use the data, in addition to supporting its reuse by individuals. This work is a contribution to the adoption of such principles towards good data management leading to good data reuse. Beyond data collection, data stewardship includes the notion of ‘long-term care’ of the data (to quote the authors of the FAIR

principles) and consequently the temporalization of both the data and their metadata. The proposed method combines an ontology and a temporal database. The ontology provides a formal, interoperable, reusable and format definition of entities and their relationships. The TRDB is used to store, trace and find all data changes. Concretely, the method enables data managers to easily create, manage, and maintain data model with the least possible resources while ensuring the fidelity, integrity, and traceability of its evolution. The use cases show that it is possible to formalize the hospital organizational structure using an ontology HORGO-FR and to generate a TRDB. The resulted artifacts allow highlighting the impact of the evolution of structures on the secondary use of healthcare data and data policy access as demonstrated. To our knowledge, it is the first time a method linking ontology and temporal database is implemented.

Clinical Significance

Contrarily to the medical informatics community paying a limited attention to the issues raised by the representation of organizational structures in databases, we believe that this topic is crucial to analyze “big longitudinal data” repositories Some terminologies integrate such entities, e.g., the *Medical Subject Headings* (MeSH) defines “health facilities” as “institutions which provide medical or health-related services”. However, the MesH hierarchy does not reflect the granularity of the entities nor the hierarchy constraints e.g., “Hospitals“ and “Hospital Units“ are siblings in MeSH. Therefore, more formal representation is needed.

Similarly, time aspects must be integrated. The first use case demonstrated that analyzing the number of obese patients regarding the evolution of organizational structures can bring relevant results. The second use case demonstrated the difficulty of implementing data access policies across time without understanding the reorganization of the structures. According to Collins *et al.* [52], the data access policies of healthcare organizations could indirectly impact the analysis of data contained in patient health records and consequences could be beyond a specific institution. A holistic approach could combine organizational-level and patient-level for data policy access. This approach is possible if we integrate an ontological and temporal description of the hospital organizational structure in CDW with consent registries. Thus, our method could make data access management easier, more controlled and more automated to ensure that the rules governing data access rights are rigorously followed even when hospital organization changes.

Technical Significance

The method takes advantage of the extensibility of shared ontologies as well as the RDB maturity and the temporal models to build a conversion process. We demonstrated that the ORG ontology developed by the W3C Consortium could be used to support a specific application and that the resulting ontology was consistent. However, our approach has two main limitations: (1) the ORG ontology is not aligned on an upper level ontology like the *Basic Formal Ontology* (BFO) [53] or the *General*

Formal Ontology (GFO) [54]. This may restrict its ability to be assembled with other biomedical ontologies to ensure proper linkage as many of them are based on the BFO. A BFO compatible ontology dedicated to organizational structures would therefore be a means to facilitate consistency across subdomains in medicine.; (2) the HORGO-FR ontology has a limited coverage, as we modeled organizations for French hospitals. Further work is needed to adapt the model to other countries and, consequently, to make the ontology consistent across countries. Moreover, HORGO-FR should be refined to include low-level units such as operating room, beds as well as healthcare providers roles and functions allowing the use of the ontology in a wider variety of clinical studies. However, it is important to note that the proposed method is domain-independent. Since no assumptions were made on the choice of ontologies, no intrinsic design choices would preclude the method from being used with the ontologies to fill these gaps or even in other domains.

Structuring the data according to the ontology and storing it in TRDB allowed the definition of temporal queries that reflected the organizational structure hierarchy. The temporalization must be founded on a sound temporal model that defines uniform structure patterns and constraints. In ontologies, there is currently no standardization of the temporalization process of classes and their relationships nor consensual temporal model as in the database community [48]. Many techniques have been proposed (e.g., reification, 4D-fluents, and versioning) but they are complex to define, they offer limited OWL reasoning capabilities and they can be costly to implement and maintain (i.e., an additional class must be added for every temporal relation) [11]. Therefore, the use of TRDB to store data evolution over time is actually a more adequate system. In our method, the temporal model is implemented in an RDB and uses already existing temporal features without further development. Moreover, the temporal model represents three types of valid time to handle future indeterminacy, period determinacy and past indeterminacy. This representation can be easily extended to add another timeline, such as transaction time, without impacting the current structure [14].

Perspectives

The work presented in this article is part of two main convergent initiatives: (1) integrating this method into a mediation architecture and in data warehouse design processes. The temporal relational schema construction from an ontology is systematic and rule driven – it might be fully automatized in a near future. Moreover, a fully integrated system including a user-friendly interface for data exploration is under development and will make data visually accessible to healthcare providers through the PARS3 platforms [55]; (2) having a data quality assessment plan for the huge sets of routine care data generated in hospitals. Previous studies conducted at AP-HP concluded that data quality control was crucial when data from the whole AP-HP, i.e., more than 30 hospitals were integrated [56], and that several temporal events, that remained unrecorded in the data warehouse until now, might have impact for data reuse [57].

CONCLUSION

This paper presented a method that generates a TRDB and integrates a formal representation of organizational structures. A proof of concept implementing this method was evaluated on hospital organizational structures domain with data from AP-HP hospital. The ontology allowed the formalization of the definitions of organizational structures and the temporal databases allowed modeling the evolution of data. In a context where both data sharing and data quality assessment are considered priorities, shared representations of health facilities and organizational structures combined with TRDB will facilitate reuse of retrospective data and improve the development of sound federated studies.

COMPETING INTERESTS

The authors have no competing interests to declare.

CONTRIBUTORS

Christina Khnaisser contributed in all the sections. Vincent Looten contributed in the use case definition, evaluation, data acquisition and interpretation. Anita Burgun supervised the work and contributed in all the sections. All authors contributed to draft and revision of the content of the paper and in the approval of the final version.

REFERENCES

1 Budrionis A, Bellika JG. The Learning Healthcare System: Where are we now? A systematic review. *Journal of Biomedical Informatics* 2016;**64**:87–92. doi:10.1016/j.jbi.2016.09.018

2 Ainsworth J, Buchan I. Combining Health Data Uses to Ignite Health System Learning. *Methods Inf Med* 2015;**54**:479–87. doi:10.3414/ME15-01-0064

3 Pacaci A, Gonul S, Sinaci AA, *et al.* A Semantic Transformation Methodology for the Secondary Use of Observational Healthcare Data in Postmarketing Safety Studies. *Front Pharmacol* 2018;**9**. doi:10.3389/fphar.2018.00435

4 Bodenreider O. Biomedical Ontologies in Action: Role in Knowledge Management, Data Integration and Decision Support. *Yearb Med Inform* 2008;:67–79.http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2592252/ (accessed 9 Dec 2015).

5 Santana da Silva F, Jansen L, Freitas F, *et al.* Ontological interpretation of biomedical database content. *J Biomed Semantics* 2017;**8**. doi:10.1186/s13326-017-0127-z

6 Haendel MA, Chute CG, Robinson PN. Classification, Ontology, and Precision Medicine. *New England Journal of Medicine* 2018;**379**:1452–62. doi:10.1056/NEJMra1615014

7 Combi C, Keravnou ET, Shahar Y. *Temporal Information Systems in Medicine*. Springer 2010. http://www.springer.com/computer/database+management+%26+information+retrieval/book/978-1-4419-6542-4 (accessed 29 May 2014).

8 Adlassnig K-P, Combi C, Das AK, *et al.* Temporal representation and reasoning in medicine: Research directions and challenges. *Artificial Intelligence in Medicine* 2006;**38**:101–13. doi:10.1016/j.artmed.2006.10.001

9 Ermolayev V, Batsakis S, Keberle N, *et al.* Ontologies of Time: Review and Trends. *ResearchGate* 2014;**11**:57–115.https://www.researchgate.net/publication/271906680_Ontologies_of_Time_Review_and_Trends (accessed 1 Aug 2016).

10 Spanos D-E, Stavrou P, Mitrou N. Bringing Relational Databases into the Semantic Web: A Survey. *Semant web* 2012;**3**:169–209. doi:10.3233/SW-2011-0055

11 Batsakis S, Tachmazidis I, Antoniou G. Representing Time and Space for the Semantic Web. *International Journal on Artificial Intelligence Tools* Published Online First: 2017. doi:10.1142/S0218213017600156

12 Khnaisser C, Lavoie L, Diab H, *et al.* Data Warehouse Design Methods Review: Trends, Challenges and Future Directions for the Healthcare Domain. In: Morzy T, Valduriez P, Bellatreche L, eds. *New Trends in Databases and Information Systems*. Springer International Publishing 2015. 76–87.http://link.springer.com/chapter/10.1007/978-3-319-23201-0_10 (accessed 15 Sep 2015).

13 Snodgrass RT. *Developing time-oriented database applications in SQL*. San Francisco, California: : Morgan Kaufmann Publishers 2000.

14 Date CJ, Darwen H, Lorentzos NA. *Time and Relational Theory: Temporal Databases in the Relational Model and SQL*. Waltham, MA: : Morgan Kaufmann 2014.

15 Jannot A-S, Zapletal E, Avillach P, *et al.* The Georges Pompidou University Hospital Clinical Data Warehouse: A 8-years follow-up experience. *International Journal of Medical Informatics* 2017;**102**:21–8. doi:10.1016/j.ijmedinf.2017.02.006

16 National Academy of Medicine. *The Future of Health Services Research: Advancing Health Systems Research and Practice in the United States*. Washington (DC): : National Academies Press (US) 2018. http://www.ncbi.nlm.nih.gov/books/NBK535995/ (accessed 3 May 2019).

17 Jha AK, Prasopa-Plaizier N, Larizgoitia I, *et al.* Patient safety research: an overview of the global evidence. *Quality and Safety in Health Care* 2010;**19**:42–7. doi:10.1136/qshc.2008.029165

18 Olsen KR, Street A. The analysis of efficiency among a small number of organisations: How inferences can be improved by exploiting patient-level data. *Health*

Economics 2008;**17**:671–81. doi:10.1002/hec.1281

19 Donabedian A. The quality of care. How can it be assessed? *JAMA* 1988;**260**:1743–8.

20 Hearld LR, Alexander JA, Fraser I, *et al.* Review: how do hospital organizational structure and processes affect quality of care?: a critical review of research methods. *Med Care Res Rev* 2008;**65**:259–99. doi:10.1177/1077558707309613

21 Danciu I, Cowan JD, Basford M, *et al.* Secondary use of clinical data: The Vanderbilt approach. *Journal of Biomedical Informatics* 2014;**52**:28–35. doi:10.1016/j.jbi.2014.02.003

22 Tuppin P, Rudant J, Constantinou P, *et al.* Value of a national administrative database to guide public decisions: From the système national d'information interrégimes de l'Assurance Maladie (SNIIRAM) to the système national des données de santé (SNDS) in France. *Revue d'Épidémiologie et de Santé Publique* 2017;**65**:S149–67. doi:10.1016/j.respe.2017.05.004

23 Ethier J-F, Dameron O, Curcin V, *et al.* A unified structural/terminological interoperability framework based on LexEVS: application to TRANSFoRm. *J Am Med Inform Assoc* 2013;**20**:986–94. doi:10.1136/amiajnl-2012-001312

24 Murphy SN, Weber G, Mendis M, *et al.* Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *Journal of the American Medical Informatics Association* 2010;**17**:124–30. doi:10.1136/jamia.2009.000893

25 Smith B, Ashburner M, Rosse C, *et al.* The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat Biotechnol* 2007;**25**:1251. doi:10.1038/nbt1346

26 Grewe N. A generic reification strategy for n-ary relations in DL. In: *Proceedings of the 2nd workshop of the GI-Fachgruppe "Ontologien in Biomedizin und Lebenswissenschaften."* Mannhiem, Germany: : Herre H, et al. 2010.

27 Arora S. A comparative study on temporal database models: A survey. In: *2015 International Symposium on Advanced Computing and Communication (ISACC)*. 2015. 161–7. doi:10.1109/ISACC.2015.7377335

28 Khnaisser C, Lavoie L, Burgun A, *et al.* Past Indeterminacy in Data Warehouse Design. In: *Database and Expert Systems Applications*. Springer, Cham 2017. 90–100. doi:10.1007/978-3-319-64471-4_9

29 Allen JF. Time and time again: The many ways to represent time. 1991. http://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=32CC39C34FFEC1BA82E3C8BEDE2101F9?doi=10.1.1.103.8212 (accessed 27 Jul 2016).

30 W3C. Time Ontology in OWL (v.2016). 2016.http://www.w3.org/TR/2016/WD-owl-time-20160712/ (accessed 27 Jul 2016).

31 O'Connor MJ, Das AK. A Method for Representing and Querying Temporal

Information in OWL. In: Fred A, Filipe J, Gamboa H, eds. *Biomedical Engineering Systems and Technologies*. Springer Berlin Heidelberg 2010. 97–110. doi:10.1007/978-3-642-18472-7_8

32 Batsakis S, Petrakis E, Tachmazidis I, *et al.* Temporal Representation and Reasoning in OWL 2. *Semantic Web journal* Published Online First: 2009.http://www.semantic-web-journal.net/ (accessed 29 Jul 2016).

33 Tao C, Wei W-Q, Solbrig HR, *et al.* CNTRO: A Semantic Web Ontology for Temporal Relation Inferencing in Clinical Narratives. *AMIA Annu Symp Proc* 2010;**2010**:787–91.http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3041418/ (accessed 26 Jul 2016).

34 Pustejovsky J, Castaño J, Ingria R, *et al.* TimeML: Robust Specification of Event and Temporal Expressions in Text. 2003. http://www.aaai.org/Library/Symposia/Spring/2003/ss03-07-005.php (accessed 16 Jun 2017).

35 Stravoskoufos K, Petrakis EGM, Mainas N, *et al.* SOWL QL: Querying Spatio-Temporal Ontologies in OWL. *J Data Semant* 2016;**5**:249–69. doi:10.1007/s13740-016-0064-5

36 Ozsoyoglu G, Snodgrass RT. Temporal and real-time databases: a survey. *IEEE Transactions on Knowledge and Data Engineering* 1995;**7**:513–32. doi:10.1109/69.404027

37 Lorentzos NA, Johnson RG. Extending relational algebra to manipulate temporal data. *Information Systems* 1988;**13**:289–96. doi:10.1016/0306-4379(88)90040-3

38 Khnaisser C. *Méthode de construction d'entrepôt de données temporalisé pour un système informationnel de santé*. 2016.http://savoirs.usherbrooke.ca/handle/11143/8386 (accessed 22 Jul 2016).

39 Khnaisser C, Lavoie L, Burgun A, *et al.* Unified Bitemporal Historicization Framework. Sherbrooke, Québec, Canada: : Université de Sherbrooke (GRIIS) 2017. http://griis.ca/en/2017/03/24/english-unified-bitemporal-historicization-framework/

40 Brochhausen M, Fransson MN, Kanaskar NV, *et al.* Developing a semantically rich ontology for the biobank-administration domain. *J Biomed Semantics* 2013;**4**:23. doi:10.1186/2041-1480-4-23

41 Utecht J, Judkins J, Otte JN, *et al.* OOSTT: a Resource for Analyzing the Organizational Structures of Trauma Centers and Trauma Systems. *CEUR Workshop Proc* 2016;**1747**.https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5312685/ (accessed 23 Apr 2019).

42 Reynolds D. The Organization Ontology. World Wide Web Consortium. 2014.https://www.w3.org/TR/vocab-org/ (accessed 7 Apr 2019).

43 Ministère du travail, de l'emploi et de la santé. Guide méthodologique de comptabilité

analytique hospitalière. 2011.https://solidarites-sante.gouv.fr/IMG/pdf/GUIDE_CAH___BOS_2011-3.pdf

44 Agence technique de l'information sur l'hospitalisation. Autorisations des unités médicales. 2014.https://www.atih.sante.fr/nomenclatures-de-recueil-de-l-information/autorisations-des-unites-medicales (accessed 3 May 2019).

45 Noy NF. Semantic Integration: A Survey of Ontology-based Approaches. *SIGMOD Rec* 2004;**33**:65–70. doi:10.1145/1041410.1041421

46 Khnaisser C, Lavoie L, Burgun A, *et al.* Generating a relational database for heterogeneous data using an ontology. (research report available through the URL and scientific article currently submitted). 2019. http://griis.ca/surl/rr-ontorel

47 Allen JF. Maintaining Knowledge About Temporal Intervals. *Commun ACM* 1983;**26**:832–843. doi:10.1145/182.358434

48 Dyreson C, Grandi F, Käfer W, *et al.* A Consensus Glossary of Temporal Database Concepts. *SIGMOD Rec* 1994;**23**:52–64. doi:10.1145/181550.181560

49 Degoulet P. The HEGP component-based clinical information system. *International Journal of Medical Informatics* 2003;**69**:115–26. doi:10.1016/S1386-5056(02)00101-6

50 TIcSante.com. Un référentiel d'organisation unique pour faciliter le pilotage de l'AP-HP. L'actualité des nouvelles technologies de la santé. 2009.http://www.ticsante.com/Un-referentiel-d-organisation-unique-pour-faciliter-le-pilotage-de-l-AP-HP-NS_341.html (accessed 17 Mar 2019).

51 Wilkinson MD, Dumontier M, Aalbersberg IjJ, *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data* 2016;**3**:160018.https://doi.org/10.1038/sdata.2016.18

52 Collins SA, Vawdrey DK, Kukafka R, *et al.* Policies for patient access to clinical data via PHRs: current state and recommendations. *Journal of the American Medical Informatics Association* 2011;**18**:i2–7. doi:10.1136/amiajnl-2011-000400

53 Arp R, Smith B, Spear AD. *Building Ontologies with Basic Formal Ontology*. MIT Press 2015. http://www.jstor.org/stable/j.ctt17kk7vw (accessed 10 Jan 2018).

54 Herre H. General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling. In: Poli R, Healy M, Kameas A, eds. *Theory and Applications of Ontology: Computer Applications*. Dordrecht: : Springer Netherlands 2010. 297–345. doi:10.1007/978-90-481-8847-5_14

55 GRIIS. PARS3. Groupe de recherche interdisciplinaire en informatique de la santé de l'Université de Sherbrooke. 2019.https://griis.ca/en/pars3/ (accessed 9 Jul 2019).

56 Daniel C, Serre P, Orlova N, *et al.* Initializing a hospital-wide data quality program. The AP-HP experience. *Computer Methods and Programs in Biomedicine*

2018;:S0169260718306242. doi:10.1016/j.cmpb.2018.10.016

57 Looten V, Kong Win Chang L, Neuraz A, *et al.* What can millions of laboratory test results tell us about the temporal aspect of data quality? Study of data spanning 17 years in a clinical data warehouse. *Computer Methods and Programs in Biomedicine* 2018;:S0169260718307089. doi:10.1016/j.cmpb.2018.12.030

APPENDIX

In Table 1, we show units valid during the interval [2013-01-01..2019-01-01]. The hierarchy is built according to ontology axioms using "unitOf" and “linkedTo”.

Each row represents a unit with the following columns:

- level: is the level in the hierarchy (level 0 is the hospital, level 1 is the healthcare division, etc.
- unitCategory: represent the acronym of the unit category.
- supId: is the super unit id, label is the current unit label.
- validTime: is the period when a unit is *supposed* to be operational in the modelled reality

Table 4A represents the hierarchical structure of unit HU 339 and HU 681 and Table 4B represents the hierarchical structure of unit HU 331 and HU 682. Note that the “old” HU 339 [HU 331] supposed to *administratively* replaced by the HU 681 [HU 682], but the valid time period of HU 339 is still open. This indicated the data about this structure was not updated. As a result, the data access right cannot be changed accordingly causing disruptions between administrative decision and medical activities. Using this example, we can conclude that when the data are not managed correctly [e.g. not update], a TRDB can help detect access rights overlaps more easily.

Table 1. Part of the hierarchical structure of the cardiovascular surgery department and the anesthesia & resuscitation department valid during [2013-01-01..2019-01-01].

(A)

level	unitCategory	id	supId	label	validTime
0	MC	75	-	HEGP	[2001-01-01,)
1	HDIV	103	75	CVRM	[2012-01-01,)
1	HDIV	105	75	ANEST REA TRAUMATO	[2012-01-01,)
2	MGU	613	105	REANIMATION CHIR	[2007-11-28,)
2	MGU	332	103	Ne plus util. REACCV	[2007-11-28,)

3	MAU	556	613	REA CHIRURGICALE	[2007-01-01,)
3	MAU	386	332	REA CHIR CARDIOVASC	[2005-01-01,)
4	HU	339	386	REA CHIR CARDIO VASCULAIRE	[2005-01-01,)
4	HU	681	556	REA CHIR POLY	[2018-09-01,)

(B)

level	unitCategory	id	supId	label	validTime
0	MC	75	-	HEGP	[2001-01-01,)
1	HDIV	103	75	CVRM	[2012-01-01,)
1	HDIV	105	75	ANEST REA TRAUMATO	[2012-01-01,)
2	MGU	613	105	REANIMATION CHIR	[2007-11-28,)
2	MGU	332	103	Ne plus util. REACCV	[2007-11-28,)
3	MAU	556	613	REA CHIRURGICALE	[2007-01-01,)
3	MAU	386	332	REA CHIR CARDIOVASC	[2005-01-01,)
4	HU	331	386	REA CHIR GREFFE	[2005-01-01,)
4	HU	682	556	REA CHIR GREFFE	[2018-09-01,)

Bibliographie

AADIL, B., WAKRIME, A.A., KZAZ, L., AND SEKKAKI, A. 2016. Automating Data warehouse design using ontology. *2016 International Conference on Electrical and Information Technologies (ICEIT)*, 42–48.

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*, 2, 385–406.

ABRIAL, J.-R. 1974. Data semantics. In: *Data Base Management*. North-Holland, Amsterdam, The Netherlands.

ACHPAL, S., BANNIHATTI KUMAR, V., AND MAHESH, K. 2016. Modeling Ontology Semantic Constraints in Relational Database Management System. Proceedings of the International MultiConference of Engineers and Computer Scientists.

ADAMSON, C. 2010. *The complete reference star schema*. McGraw-Hill, New York.

AFZAL, H., WAQAS, M., AND NAZ, T. 2016. OWLMap: Fully Automatic Mapping of Ontology into Relational Database Schema. *International Journal of Advanced Computer Science and Applications (IJACSA) 7*, 11.

ALLEN, J.F. 1983. Maintaining Knowledge About Temporal Intervals. *Commun. ACM 26*, 11, 832–843.

ALLEN, J.F. AND FERGUSON, G. 1994. *Actions and Events in Interval Temporal Logic*. The University of Rochester, Computer Science Department.

ANAGNOSTOPOULOS, E., BATSAKIS, S., AND PETRAKIS, E.G.M. 2013. CHRONOS: A Reasoning Engine for Qualitative Temporal Information in OWL. *Procedia Computer Science 22*, 70–77.

ANSELMA, L., BOTTRIGHI, A., MONTANI, S., AND TERENZIANI, P. 2013a. Extending BCDM to Cope with Proposals and Evaluations of Updates. *IEEE Transactions on Knowledge and Data Engineering 25*, 3, 556–570.

ANSELMA, L., PIOVESAN, L., SATTAR, A., STANTIC, B., AND TERENZIANI, P. 2016. A Comprehensive Approach to‘Now’in Temporal Relational Databases: Semantics and Representation. *IEEE Transactions on Knowledge and Data Engineering 28*, 10, 2538–2551.

ANSELMA, L., PIOVESAN, L., AND TERENZIANI, P. 2015. A 1NF temporal relational model and algebra coping with valid-time temporal indeterminacy. *Journal of Intelligent Information Systems*, 1–30.

ANSELMA, L., STANTIC, B., TERENZIANI, P., AND SATTAR, A. 2013b. Querying now-relative data. *Journal of Intelligent Information Systems 41*, 2, 285–311.

ANSELMA, L., TERENZIANI, P., AND SNODGRASS, R.T. 2010. Valid-Time Indeterminacy in Temporal Relational Databases: A Family of Data Models. *2010 17th International Symposium on Temporal Representation and Reasoning (TIME)*, 139–145.

D'AQUIN, M., SCHLICHT, A., STUCKENSCHMIDT, H., AND SABOU, M. 2009. Criteria and Evaluation for Ontology Modularization Techniques. In: H. Stuckenschmidt, C. Parent and S. Spaccapietra, eds., *Modular Ontologies*. Springer Berlin Heidelberg, Berlin, Heidelberg, 67–89.

ARTALE, A. AND FRANCONI, E. 1999. Introducing Temporal Description Logics. *Proceedings of the Sixth International Workshop on Temporal Representation and Reasoning*, IEEE Computer Society, 2–.

ARTALE, A. AND FRANCONI, E. 2000. A survey of temporal extensions of description logics. *Annals of Mathematics and Artificial Intelligence 30*, 1, 171–210.

ARTALE, A., KOVTUNOVA, A., KONTCHAKOV, R., RYZHIKOV, V., WOLTER, F., AND ZAKHARYASCHEV, M. 2015. First-order rewritability of temporal ontology-mediated queries. In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. AAAI Press, Palo Alto, U.S., 2706–2712.

BAADER, F., ED. 2010. *The description logic handbook: theory, implementation, and applications*. Cambridge Univ. Press, Cambridge.

BARTON, A., KHNAISSER, C., LAVOIE, L., AND ETHIER, J.-F. 2017. Ambiguities in temporalized relational databases: A referent-tracking view. *Proceedings of Ontology and Data in Life Sciences (ODLS 2017)*, CEUR Workshop proceedings.

BATSAKIS, S. AND PETRAKIS, E.G.M. 2011. SOWL: A Framework for Handling Spatio-temporal Information in OWL 2.0. In: N. Bassiliades, G. Governatori and A. Paschke, eds., *Rule-Based Reasoning, Programming, and Applications*. Springer Berlin Heidelberg, 242–249.

BATSAKIS, S., TACHMAZIDIS, I., AND ANTONIOU, G. 2017. Representing Time and Space for the Semantic Web. *International Journal on Artificial Intelligence Tools*.

BAUMANN, R., LOEBE, F., AND HERRE, H. 2014. Axiomatic Theories of the Ontology of Time in GFO. *Appl. Ontol. 9*, 3–4, 171–215.

BELLATRECHE, L., AIT-AMEUR, Y., AND CHAKROUN, C. 2010. A design methodology of ontology based database applications. *Logic Journal of the IGPL 19*, 5, 648–665.

BEN-ZVI, J. 1982. The Time Relational Model. .

BERGSON, H. 1910. *Time and free will: an essay on the immediate data of consciousness.* S. Sonnenschein & co., lim., London, New York.

BLAISURE, J.C. AND CEUSTERS, W.M. 2018. Improving the "Fitness for Purpose" of Common Data Models through Realism Based Ontology. *AMIA Annual Symposium Proceedings 2017*, 440–447.

BÖHLEN, M.H., DIGNÖS, A., GAMPER, J., AND JENSEN, C.S. 2018. Temporal Data Management – An Overview. *Business Intelligence and Big Data*, Springer International Publishing, 51–83.

BONIFATI, A., FURNISS, P., GREEN, A., HARMER, R., OSHURKO, E., AND VOIGT, H. 2019. Schema Validation and Evolution for Graph Databases. *arXiv:1902.06427 [cs]*.

BONO, B. DE, HOEHNDORF, R., WIMALARATNE, S., GKOUTOS, G., AND GRENON, P. 2011. The RICORDO approach to semantic interoperability for biomedical data and models: strategy, standards and solutions. *BMC Research Notes 4*, 1, 313.

BOOCH, G., REESE, F., REESE, L., AND BONNETAIN, P.-Y. 1997. *Analyse et conception orientées objets*. Éd. Addidon-Wesley France, Paris; Reading (Mass.); Amsterdam.

BORGWARDT, S., LIPPMANN, M., AND THOST, V. 2015. Temporalizing rewritable query languages over knowledge bases. *Journal of Web Semantics 33*, 50–70.

BRUCE, T.A. 1992. *Designing Quality Databases with IDEF1X Information Models*. Dorset House Publishing Co., Inc., New York, NY, USA.

BRUIJN, J. DE, EHRIG, M., FEIER, C., MARTINS-RECUERDA, F., SCHARFFE, F., AND WEITEN, M. 2006. Ontology Mediation, Merging, and Aligning. In: J. Davies, R. Studer and P. Warren, eds., *Semantic Web Technologies*. John Wiley & Sons, Ltd, Chichester, UK, 95–113.

BUDRIONIS, A. AND BELLIKA, J.G. 2016. The Learning Healthcare System: Where are we now? A systematic review. *Journal of Biomedical Informatics 64*, 87–92.

BUREAU INTERNATIONAL DES POIDS ET MESURES, ED. 2006. *Le système international d'unités (SI)*. BIPM, Sèvres.

BURGUN, A., BERNAL-DELGADO, E., KUCHINKE, W., ET AL. 2017. Health Data for Public Health: Towards New Ways of Combining Data Sources to Support Research Efforts in Europe. *Yearbook of Medical Informatics 26*, 01, 235–240.

CALVANESE, D., DE GIACOMO, G., LEMBO, D., LENZERINI, M., ROSATI, R., AND RUZZI, M. 2010. Using OWL in Data Integration. In: R. de Virgilio, F. Giunchiglia and L. Tanca, eds., *Semantic Web Information Management: A Model-Based Perspective*. Springer Berlin Heidelberg, Berlin, Heidelberg, 397–424.

CALVANESE, D., GIACOMO, G.D., LEMBO, D., LENZERINI, M., POGGI, A., AND ROSATI, R. 2007. Ontology-based database access. .

CALVANESE, D., KALAYCI, E.G., RYZHIKOV, V., XIAO, G., AND ZAKHARYASCHEV, M. 2017. Metric Temporal Logic for Ontology-Based Data Access over Log Data. *arXiv:1701.00976 [cs]*.

CARDOSO, J., ED. 2007. *Semantic Web Services: Theory, Tools and Applications*. IGI Global.

CARROLL, J.J., BIZER, C., HAYES, P., AND STICKLER, P. 2005. Named graphs. *Journal of Web Semantics 3*, 4, 247–267.

CEUSTERS, W. AND BLAISURE, J. 2017. A Realism-Based View on Counts in OMOP's Common Data Model. *Studies in Health Technology and Informatics 237*, 55–62.

CEUSTERS, W. AND SMITH, B. 2006. Strategies for referent tracking in electronic health records. *Journal of Biomedical Informatics 39*, 3, 362–378.

CHAMBERS, D.A., FEERO, W.G., AND KHOURY, M.J. 2016. Convergence of Implementation Science, Precision Medicine, and the Learning Health Care System: A New Model for Biomedical Research. *JAMA 315*, 18, 1941–1942.

CHEN, P.P.-S. 1976. The Entity-relationship Model—Toward a Unified View of Data. *ACM Trans. Database Syst. 1*, 1, 9–36.

CHOMICKI, J. AND SAAKE, G. 1998. *Logics for Databases and Information Systems*. Springer Science & Business Media.

CHROMIAK, M. AND STENCEL, K. 2014. A Data Model for Heterogeneous Data Integration Architecture. In: S. Kozielski, D. Mrozek, P. Kasprowski, B. Małysiak-Mrozek and D. Kostrzewa, eds., *Beyond Databases, Architectures, and Structures*. Springer International Publishing, 547–556.

CLIFFORD, J. AND CROKER, A. 1987. The historical relational data model (HRDM) and algebra based on lifespans. *1987 IEEE Third International Conference on Data Engineering*, 528–537.

CODD, E.F. 1979. Extending the Database Relational Model to Capture More Meaning. *ACM Trans. Database Syst. 4*, 4, 397–434.

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

COMBI, C., KERAVNOU, E.T., AND SHAHAR, Y. 2010. *Temporal Information Systems in Medicine*. Springer.

CURINO, C., ORSI, G., PANIGATI, E., AND TANCA, L. 2009. Accessing and Documenting Relational Databases through OWL Ontologies. *Flexible Query Answering Systems*, Springer, Berlin, Heidelberg, 431–442.

DANCIU, I., COWAN, J.D., BASFORD, M., ET AL. 2014. Secondary use of clinical data: The Vanderbilt approach. *Journal of Biomedical Informatics 52*, 28–35.

DANESE, M.D., HALPERIN, M., DURYEA, J., AND DURYEA, R. 2019. The Generalized Data Model for clinical research. *BMC Medical Informatics and Decision Making 19*, 1, 117.

DARWEN, H. 1998. Valid time and transaction time proposals: Language design aspects. In: O. Etzion, S. Jajodia and S. Sripada, eds., *Temporal Databases: Research and Practice*. Springer Berlin Heidelberg, 195–210.

DARWEN, H. AND DATE, C.J. 1995. *Databases, Types and the Relational Model : The Third Manifesto*. Addison Wesley.

DATE, C.J. 2015. *SQL and relational theory: how to write accurate SQL code*. O'Reilly, Sebastopol, Calif.

DATE, C.J. 2016. *Type inheritance and relational theory: subtypes, supertypes, and substitutability*. O'Reilly Media, Sebastopol, CA.

DATE, C.J. AND DARWEN, H. 2010. *Database Explorations: essays on the Third Manifesto and related topics*. Trafford Publishing.

DATE, C.J., DARWEN, H., AND LORENTZOS, N.A. 2003. *Temporal data and the relational model: a detailed investigation into the application of interval and relation theory to the problem of temporal database management*. Morgan Kaufmann Publishers, San Diego, CA.

DATE, C.J., DARWEN, H., AND LORENTZOS, N.A. 2014. *Time and Relational Theory: Temporal Databases in the Relational Model and SQL*. Morgan Kaufmann, Waltham, MA.

DE NICOLA, A. AND MISSIKOFF, M. 2016. A Lightweight Methodology for Rapid Ontology Engineering. *Commun. ACM 59*, 3, 79–86.

DEFOSSEZ, G., ROLLET, A., DAMERON, O., AND INGRAND, P. 2014. Temporal representation of care trajectories of cancer patients using data from a regional information system: an application in breast cancer. *BMC Medical Informatics and Decision Making 14*, 1, 24.

DEPUTY CHIEF INFORMATION OFFICER. 2015. DODAF - DOD Architecture Framework Version 2.02. http://dodcio.defense.gov/TodayinCIO/DoDArchitectureFramework.aspx.

DIGNÖS, A., BÖHLEN, M., AND GAMPER, J. 2013. Query time scaling of attribute values in interval timestamped databases. *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, 1304–1307.

DIGNÖS, A., BÖHLEN, M.H., AND GAMPER, J. 2014. Overlap interval partition join. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data - SIGMOD '14*, ACM Press, 1459–1470.

DIGNÖS, A., GLAVIC, B., NIU, X., BÖHLEN, M., AND GAMPER, J. 2019. Snapshot Semantics for Temporal Multiset Relations. *Proc. VLDB Endow. 12*, 6, 639–652.

DOU, D., QIN, H., AND LEPENDU, P. 2010. OntoGrate : Towards Automatic Integration For Relational Databases And The Semantic Web Through An Ontology-Based Framework. *International Journal of Semantic Computing 04*, 01, 123–151.

DYRESON, C., GRANDI, F., KÄFER, W., ET AL. 1994. A Consensus Glossary of Temporal Database Concepts. *SIGMOD Rec. 23*, 1, 52–64.

ELMASRI, R. AND NAVATHE, S.B. 2016. *Fundamentals of database systems*. Pearson.

ERMOLAYEV, V., BATSAKIS, S., KEBERLE, N., TATARINTSEVA, O., AND ANTONIOU, G. 2014. Ontologies of Time: Review and Trends. *International Journal of Computer Science and Applications 11*, 3, 57–115.

ERMOLAYEV, V., KEBERLE, N., AND MATZKE, W.E. 2008. An Ontology of Environments, Events, and Happenings. *2008 32nd Annual IEEE International Computer Software and Applications Conference*, 539–546.

ETHIER, J., MCGILCHRIST MARK, BARTON ADRIEN, ET AL. 2017. The TRANSFoRm project: Experience and lessons learned regarding functional and interoperability requirements to support primary care. *Learning Health Systems*.

ETHIER, J.-F., DAMERON, O., CURCIN, V., ET AL. 2013. A unified structural/terminological interoperability framework based on LexEVS: application to TRANSFoRm. *Journal of the American Medical Informatics Association 20*, 5, 986–994.

FAGIN, R. 1979. Normal Forms and Relational Database Operators. *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, ACM, 153–160.

FALKENBERG, E.D. 1976. Concepts for modelling information. *Proc. 1976 IFIP Working Conf. on Modelling in Data Base Management Systems*, G.M. Nijssen, Freudenstadt, Germany, North-Holland Publishing, 95–109.

GADIA, S.K. AND YEUNG, C.-S. 1988. A Generalized Model for a Relational Temporal Database. *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, ACM, 251–259.

GALI, A., CHEN, C.X., CLAYPOOL, K.T., AND UCEDA-SOSA, R. 2004. From Ontology to Relational Databases. In: S. Wang, K. Tanaka, S. Zhou, et al., eds., *Conceptual Modeling for Advanced Application Domains*. Springer Berlin Heidelberg, 278–289.

GALTON, A. 2018. The Treatment of Time in Upper Ontologies. *Frontiers in Artificial Intelligence and Applications*, 33–46.

GRAU, B.C., HORROCKS, I., KAZAKOV, Y., AND SATTLER, U. 2007. Just the Right Amount: Extracting Modules from Ontologies. *Proceedings of the 16th International Conference on World Wide Web*, ACM, 717–726.

GROSSMANN, C., POWERS, B., MCGINNIS, J.M., MCGINNIS, J.M., AND INSTITUTE OF MEDICINE (U.S.), EDS. 2011. *Digital infrastructure for the learning health system: the foundation for continuous improvement in health and health care: workshop series summary*. Institute of Medicine of The National Academies, Washington, D.C.

GRUBER, T. 2009. Ontology. *Encyclopedia of Database Systems*, 1963–1965. http://www.springerlink.com/index/10.1007/978-0-387-39940-9_1318.

GUARINO, N. 1997. Understanding, Building and Using Ontologies. *Int. J. Hum.-Comput. Stud. 46*, 2–3, 293–310.

GUARINO, N. 1998. Formal ontology in information systems. *46*.

GUTIERREZ, C., HURTADO, C., AND VAISMAN, A. 2005. Temporal RDF. *Proceedings of the Second European Conference on The Semantic Web: Research and Applications*, Springer-Verlag, 93–107.

HAENDEL, M.A., CHUTE, C.G., AND ROBINSON, P.N. 2018. Classification, Ontology, and Precision Medicine. *New England Journal of Medicine 379*, 15, 1452–1462.

HAI, R., GEISLER, S., AND QUIX, C. 2016. Constance: An Intelligent Data Lake System. *Proceedings of the 2016 International Conference on Management of Data*, ACM, 2097–2100.

HALPERN, J.Y. AND SHOHAM, Y. 1991. A Propositional Modal Logic of Time Intervals. *J. ACM 38*, 4, 935–962.

HALPIN, T. 1998. Object-Role Modeling (ORM/NIAM). In: P. Bernus, K. Mertins and G. Schmidt, eds., *Handbook on Architectures of Information Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 81–103.

HARPER, J., HAUCK, K., AND STREET, A. 2001. Analysis of costs and efficiency in general surgery specialties in the United Kingdom. *The European Journal of Health Economics (HEPAC) 2*, 4, 150–157.

HEARLD, L.R., ALEXANDER, J.A., FRASER, I., AND JIANG, H.J. 2008. Review: how do hospital organizational structure and processes affect quality of care?: a critical review of research methods. *Medical care research and review: MCRR 65*, 3, 259–299.

HEIMBIGNER, D. AND MCLEOD, D. 1985. A Federated Architecture for Information Management. *ACM Trans. Inf. Syst. 3*, 3, 253–278.

HERRE, H. 2010. General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling. In: R. Poli, M. Healy and A. Kameas, eds., *Theory and Applications of Ontology: Computer Applications*. Springer Netherlands, Dordrecht, 297–345.

HO, L.T.T., TRAN, C.P.T., AND HOANG, Q. 2015. An Approach of Transforming Ontologies into Relational Databases. *Intelligent Information and Database Systems*, Springer, Cham, 149–158.

HOBBS, J.R. AND PAN, F. 2004. An Ontology of Time for the Semantic Web. *ACM Transactions on Asian Language Information Processing 3*, 66–85.

HOBBS, J.R. AND PUSTEJOVSKY, J. 2002. Annotating and Reasoning about Time and Events. .

HORNUNG, T. AND MAY, W. 2013. Experiences from a TBox Reasoning Application: Deriving a Relational Model by OWL Schema Analysis. *Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013) co-located with 10th Extended Semantic Web Conference (ESWC 2013), Montpellier, France, May 26-27, 2013*, CEUR-WS.org.

INMON, W.H. 2002. *Building the data warehouse*. J. Wiley, New York.

ISO. 2004. ISO 8601:2004 - Data elements and interchange formats -- Information interchange -- Representation of dates and times. *ISO*. http://www.iso.org/iso/catalogue_detail?csnumber=40874.

JANNOT, A.-S., ZAPLETAL, E., AVILLACH, P., MAMZER, M.-F., BURGUN, A., AND DEGOULET, P. 2017. The Georges Pompidou University Hospital Clinical Data Warehouse: A 8-years follow-up experience. *International Journal of Medical Informatics 102*, 21–28.

JARKE, M., JEUSFELD, M., AND QUIX, C. 2014. Data-centric intelligent information integration—from concepts to automation. *Journal of Intelligent Information Systems 43*, 3, 437–462.

JENSEN, C.S. AND SNODGRASS, R.T. 1996. Semantics of time-varying information. *Information Systems 21*, 4, 311–352.

JENSEN, C.S., SOO, M.D., AND SNODGRASS, R.T. 1993. Unifying Temporal Data Models via a Conceptual Model. *Information Systems 19*, 513–547.

JENSEN, P.B., JENSEN, L.J., AND BRUNAK, S. 2012. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics 13*, 6, 395–405.

JIANG, B. 2011. *Constructing data warehouses with metadata-driven generic operators, and more architecture, methodology, and paradigm, concepts, algorithms, and operators, principles, recommendations, and exercices*. DBJ Publishing, Niederglatt (8172.

JIMENEZ-RUIZ, E., KHARLAMOV, E., ZHELEZNYAKOV, D., ET AL. 2015. BootOX: Practical Mapping of RDBs to OWL 2. *The Semantic Web - ISWC 2015*, Springer, Cham, 113–132.

JOHN, S., SHAH, N., AND STEWART, C. 2018. Towards a Software Centric Approach for Ontology Development: Novel Methodology and its Application. *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)*, IEEE, 139–146.

JOVANOVIC, V. AND BOJICIC, I. 2012. Conceptual data vault model. *Proceedings of the Southern Association for Information Systems Conference*.

KHNAISSER, C. 2016. Méthode de construction d'entrepôt de données temporalisé pour un système informationnel de santé. http://savoirs.usherbrooke.ca/handle/11143/8386.

KHNAISSER, C., LAVOIE, L., BURGUN, A., AND ETHIER, J.-F. 2017a. *Unified Bitemporal Historicization Framework*. Université de Sherbrooke (GRIIS), Sherbrooke, Québec, Canada.

KHNAISSER, C., LAVOIE, L., BURGUN, A., AND ETHIER, J.-F. 2017b. Past Indeterminacy in Data Warehouse Design. *Database and Expert Systems Applications*, Springer, Cham, 90–100.

KHNAISSER, C., LAVOIE, L., BURGUN, A., AND ETHIER, J.-F. 2018. Generating Relational Database using Ontology Review : Issues, Challenges and Trends. *International Journal of Advanced Computer Science and Applications (IJACSA) 9*, 6.

KHNAISSER, C., LAVOIE, L., DIAB, H., AND ETHIER, J.-F. 2015. Data Warehouse Design Methods Review: Trends, Challenges and Future Directions for the Healthcare Domain. In: T. Morzy, P. Valduriez and L. Bellatreche, eds., *New Trends in Databases and Information Systems*. Springer International Publishing, 76–87.

KHOURI, S. AND LADJEL, B. 2010. A Methodology and Tool for Conceptual Designing a Data Warehouse from Ontology-based Sources. *Proceedings of the ACM 13th International Workshop on Data Warehousing and OLAP*, ACM, 19–24.

KIM, S.-K., SONG, M.-Y., KIM, C., YEA, S.-J., JANG, H.C., AND LEE, K.-C. 2008. Temporal Ontology Language for Representing and Reasoning Interval-Based Temporal Knowledge. *The Semantic Web*, Springer Berlin Heidelberg, 31–45.

KIMBALL, R. 2004. *The Data Warehouse ETL Toolkit : practical techniques for extracting, cleaning, conforming, and delivering data*. Wiley, Indianapolis, IN.

KIMBALL, R. 2013. *The data warehouse toolkit: the definitive guide to dimensional modeling*. John Wiley & Sons, Inc, Indianapolis, IN.

KIRRANE, S., MILEO, A., AND DECKER, S. 2017. Access control and the Resource Description Framework: A survey. *Semantic Web 8*, 2, 311–352.

KLEIN, M. AND FENSEL, D. 2001. Ontology Versioning on the Semantic Web. *Proceedings of the First International Conference on Semantic Web Working*, CEUR-WS.org, 75–91.

KONTCHAKOV, R., LUTZ, C., TOMAN, D., WOLTER, F., AND ZAKHARYASCHEV, M. 2011. The combined approach to ontology-based data access. In: T. Walsh, ed., *Twenty-Second International Joint Conference on Artificial Intelligence*. AAAI Press, California, U.S., 2656–2661.

KOUBARAKIS, M. 1992. Dense Time and Temporal Constraints with ≠. *In Proc. 3rd International Conference on Principles of Knowledge Representation and Reasoning (kr-92*, Morgan Kaufmann Publishers, 24–35.

KRIEGER, H.-U. 2010. A General Methodology for Equipping Ontologies with Time. *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*, European Language Resources Association.

KULKARNI, G.S., LAUPACIS, A., URBACH, D.R., FLESHNER, N.E., AND AUSTIN, P.C. 2009. Varied definitions of hospital volume did not alter the conclusions of volume–outcome analyses. *Journal of Clinical Epidemiology 62*, 4, 400–407.

KULKARNI, K. AND MICHELS, J.-E. 2012. Temporal Features in SQL:2011. *SIGMOD Rec. 41*, 3, 34–43.

LALECI, G.B., YUKSEL, M., AND DOGAC, A. 2013. Providing semantic interoperability between clinical care and clinical research domains. *IEEE journal of biomedical and health informatics 17*, 2, 356–369.

LAMB, A., FULLER, M., VARADARAJAN, R., ET AL. 2012. The Vertica Analytic Database: C-store 7 Years Later. *Proc. VLDB Endow. 5*, 12, 1790–1801.

LAMPORT, L. 1978. Time, Clocks, and the Ordering of Events in a Distributed System. *Commun. ACM 21*, 7, 558–565.

LEDIEU, T., BOUZILLE, G., THIESSARD, F., ET AL. 2018. Timeline representation of clinical data: usability and added value for pharmacovigilance. *BMC Medical Informatics and Decision Making 18*, 1, 86.

LINSTEDT, D. AND OLSCHIMKE, M. 2016. *Building a scalable data warehouse with Data Vault 2.0*. Morgan Kaufmann, Waltham, Massachusetts.

LOOTEN, V., KONG WIN CHANG, L., NEURAZ, A., ET AL. 2018. What can millions of laboratory test results tell us about the temporal aspect of data quality? Study of data spanning 17 years in a clinical data warehouse. *Computer Methods and Programs in Biomedicine*, S0169260718307089.

LORENTZOS, N.A. AND JOHNSON, R.G. 1988. Extending relational algebra to manipulate temporal data. *Information Systems 13*, 3, 289–96.

LORENTZOS, N.A. AND MITSOPOULOS, Y.G. 1997. SQL extension for interval data. *IEEE Transactions on Knowledge and Data Engineering 9*, 3, 480–499.

LUTZ, C. 2001. Interval-based Temporal Reasoning with General TBoxes. *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1*, Morgan Kaufmann Publishers Inc., 89–94.

LUTZ, C., WOLTER, F., AND ZAKHARYASCHEV, M. 2008. Temporal Description Logics: A Survey. IEEE, 3–14.

MADKOUR, M., BENHADDOU, D., AND TAO, C. 2016. Temporal data representation, normalization, extraction, and reasoning: A review from clinical domain. *Computer Methods and Programs in Biomedicine 128*, 52–68.

MAHMUDI, K., LIEM, M.M.I., AND AKBAR, S. 2018. Ontology to relational database transformation for web application development and maintenance. *Journal of Physics: Conference Series 971*, 1, 012031.

MANTHEY, R. 2014. Temporal Information Systems - SS 14. http://www.iai.uni-bonn.de/III/lehre/vorlesungen/TemporalIS/SS14/.

MARTINEZ-RODRIGUEZ, J.L., HOGAN, A., AND LOPEZ-AREVALO, I. 2018. Information extraction meets the Semantic Web: A survey. *Semantic Web Preprint*, Preprint, 1–81.

MASOLO, C., BORGO, S., GANGEMI, A., GUARINO, N., OLTRAMARI, A., AND SCHNEIDER, L. 2002. *The WonderWeb Library of Foundational Ontologies*. .

MATE, S., IXCHEL, C., THOMAS, G., HANS-ULRICH, P., AND STEFAN, K. 2017. Standards-Based Procedural Phenotyping: The Arden Syntax on i2b2. *Studies in Health Technology and Informatics*, 37–41.

MATE, S., KÖPCKE, F., TODDENROTH, D., ET AL. 2015. Ontology-Based Data Integration between Clinical and Research Systems. *PLoS ONE 10*, 1.

MATTIOLI, D. AND GUBITOSO, M.D. 2018. Application of Graph Database in the Storage of Heterogeneous Omics Data for the Treatment in Bioinformatics. *Proceedings of the 2018 10th International Conference on Bioinformatics and Biomedical Technology*, ACM, 51–56.

MCDERMOTT, D. 1982. A Temporal Logic for Reasoning about Processes and Plans. *Cognitive Science 6*, 2, 101–155.

MCKENZIE, E. AND SNODGRASS, R.T. 1991. *Supporting valid time in an historical relational algebra : Proofs and extensions*. Dept. of Computer Science, Univ. of Arizona, Tucson.

MEISEN, P. 2016. *Analyzing time interval data: introducing an information system for time interval data analysis*. Springer Vieweg, Wiesbaden.

METKE-JIMENEZ, A., STEEL, J., HANSEN, D., AND LAWLEY, M. 2018. Ontoserver: a syndicated terminology server. *Journal of Biomedical Semantics 9*, 1, 24.

MOTIK, B., PATEL-SCHNEIDER, P.F., AND CUENCA GRAU, B. 2012a. OWL 2 Web Ontology Language Direct Semantics (Second Edition). *W3C*. https://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/.

MOTIK, B., PATEL-SCHNEIDER, P.F., AND PARSIA, B. 2012b. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). *W3C*. https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

NATIONAL ACADEMY OF MEDICINE. 2018. *The Future of Health Services Research: Advancing Health Systems Research and Practice in the United States*. National Academies Press (US), Washington (DC).

NEBOT, V. AND LLAVORI, R.B. 2014. Exploiting semantic annotations for open information extraction: an experience in the biomedical domain. *Knowl. Inf. Syst. 38*, 2, 365–389.

NOY, N., RECTOR, A., HAYES, P., AND WELTY, C. 2006. Defining n-ary relations on the semantic web. *W3C working group note*. https://www.w3.org/TR/swbp-n-aryRelations/.

OBO. 2018. The Open Biological and Biomedical Ontology Foundry. *The OBO Foundry*. http://www.obofoundry.org/.

O'CONNOR, M.J. AND DAS, A.K. 2010. A Method for Representing and Querying Temporal Information in OWL. In: A. Fred, J. Filipe and H. Gamboa, eds., *Biomedical Engineering Systems and Technologies*. Springer Berlin Heidelberg, 97–110.

O'CONNOR, M.J., TU, S.W., AND MUSEN, M.A. 2000. Representation of temporal indeterminacy in clinical databases. *Proceedings of the AMIA Symposium*, 615–619.

OH, S. AND YEOM, H.Y. 2012. A comprehensive framework for the evaluation of ontology modularization. *Expert Systems with Applications 39*, 10, 8547–8556.

OLSEN, K.R. AND STREET, A. 2008. The analysis of efficiency among a small number of organisations: How inferences can be improved by exploiting patient-level data. *Health Economics 17*, 6, 671–681.

OZSOYOGLU, G. AND SNODGRASS, R.T. 1995. Temporal and real-time databases: a survey. *IEEE Transactions on Knowledge and Data Engineering 7*, 4, 513–532.

PINAIRE, J., AZE, J., BRINGAY, S., AND LANDAIS, P. 2017. Patient healthcare trajectory. An essential monitoring tool: a systematic review. *Health Information Science and Systems 5*, 1.

PINTO, J.A. 1994. Temporal Reasoning in the Situation Calculus. .

PODSIADLY-MARCZYKOWSKA, T., GAMBIN, T., AND ZAWISLAK, R. 2014. Rule-Based Algorithm Transforming OWL Ontology Into Relational Database. *Beyond Databases, Architectures, and Structures*, Springer, Cham, 148–159.

POGGI, A., LEMBO, D., CALVANESE, D., GIACOMO, G.D., LENZERINI, M., AND ROSATI, R. 2008. Linking Data to Ontologies. In: *Journal on Data Semantics X*. Springer, Berlin, Heidelberg, 133–173.

PREVENTIS, A., MARKI, P., PETRAKIS, E.G.M., AND BATSAKIS, S. 2012. CHRONOS: A Tool for Handling Temporal Ontologies in Protégé. *2012 IEEE 24th International Conference on Tools with Artificial Intelligence (ICTAI)*, 460–467.

PRIOR, A.N. 1967. *Past, Present and Future*. Oxford University Press, Oxford, New York.

REYNOLDS, D. 2014. The Organization Ontology. *World Wide Web Consortium*. https://www.w3.org/TR/vocab-org/.

ROCHFIELD, A., COLLETTI, R., AND TARDIEU, H. 1989. *La méthode MERISE*. Les Éditions d'Organisation.

RODRIGUEZ-MURO, M., KONTCHAKOV, R., AND ZAKHARYASCHEV, M. 2013. Ontology-Based Data Access: Ontop of Databases. *The Semantic Web – ISWC 2013*, Springer, Berlin, Heidelberg, 558–573.

ROMERO, O. AND ABELLO, A. 2009. A Survey of Multidimensional Modeling Methodologies. *International Journal of Data Warehousing and Mining (IJDWM) 5*, 2, 1–23.

ROMERO, O. AND ABELLO, A. 2010. Automatic validation of requirements to support multidimensional design. *Data & Knowledge Engineering 69*, 9, 917–942.

RÖNNBÄCK, L., REGARDT, O., BERGHOLTZ, M., JOHANNESSON, P., AND WOHED, P. 2010. Anchor modeling — Agile information modeling in evolving data environments. *Data & Knowledge Engineering 69*, 12, 1229–1253.

RUMBAUGH, J., ED. 1991. *Object-oriented modeling and design*. Prentice Hall, Englewood Cliffs, N.J.

SACCOL, D. D B., ANDRADE, T. D C., AND PIVETA, E.K. 2011. Mapping OWL ontologies to relational schemas. *2011 IEEE International Conference on Information Reuse Integration*, 71–76.

SANDEWALL, E. 1995. *Features and Fluents: The Representation of Knowledge About Dynamical Systems*. Oxford University Press, Inc., New York, NY, USA.

SCHMIDT, M., SCHMIDT, S.A.J., SANDEGAARD, J.L., EHRENSTEIN, V., PEDERSEN, L., AND SØRENSEN, H.T. 2015. The Danish National Patient Registry: a review of content, data quality, and research potential. *Clinical Epidemiology*, 449.

SENKO, M.E. 1975. Information systems: Records, relations, sets, entities, and things. *Information Systems 1*, 1, 3–13.

SMITH, B. 2015. *Basic Formal Ontology 2.0 : Specification and user's guide*. .

SMITH, B., CEUSTERS, W., KLAGGES, B., ET AL. 2005. Relations in biomedical ontologies. *Genome Biology 6*, R46.

SMITH, J.M., BERNSTEIN, P.A., DAYAL, U., ET AL. 1981. Multibase: Integrating Heterogeneous Distributed Database Systems. *Proceedings of the May 4-7, 1981, National Computer Conference*, ACM, 487–499.

SNODGRASS, R. 1987. The Temporal Query Language TQuel. *ACM Trans. Database Syst. 12*, 2, 247–298.

SNODGRASS, R.T. 1995. *The TSQL2 Temporal Query Language*. Springer.

SNODGRASS, R.T. 2000. *Developing time-oriented database applications in SQL*. Morgan Kaufmann Publishers, San Francisco, California.

SPANOS, D.-E., STAVROU, P., AND MITROU, N. 2012. Bringing Relational Databases into the Semantic Web: A Survey. *Semant. web 3*, 2, 169–209.

STOREY, V.C., TRUJILLO, J.C., AND LIDDLE, S.W. 2015. Research on conceptual modeling: Themes, topics, and introduction to the special issue. *Data & Knowledge Engineering 98*, 1–7.

STUCKENSCHMIDT, H., PARENT, C., AND SPACCAPIETRA, S., EDS. 2009. *Modular Ontologies*. Springer Berlin Heidelberg, Berlin, Heidelberg.

SUGUMARAN, V. AND STOREY, V.C. 2006. The Role of Domain Ontologies in Database Design: An Ontology Management and Conceptual Modeling Environment. *ACM Trans. Database Syst. 31*, 3, 1064–1094.

TANSEL, A.U. 1986. Adding time dimension to relational model and extending relational algebra. *Information Systems 11*, 4, 343–355.

TAO, C., WEI, W.-Q., SOLBRIG, H.R., SAVOVA, G., AND CHUTE, C.G. 2010. CNTRO: A Semantic Web Ontology for Temporal Relation Inferencing in Clinical Narratives. *AMIA Annual Symposium Proceedings 2010*, 787–791.

TAPPOLET, J. AND BERNSTEIN, A. 2009. Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. *The Semantic Web: Research and Applications*, Springer Berlin Heidelberg, 308–322.

THENMOZHI, M. AND VIVEKANANDAN, K. 2012. An ontology based hybrid approach to derive multidimensional schema for data warehouse. *International Journal of Computer Applications 54*, 8, 36–42.

TICSANTE.COM. 2009. Un référentiel d'organisation unique pour faciliter le pilotage de l'AP-HP. *L'actualité des nouvelles technologies de la santé*. http://www.ticsante.com/Un-referentiel-d-organisation-unique-pour-faciliter-le-pilotage-de-l-AP-HP-NS_341.html.

TUPPIN, P., RUDANT, J., CONSTANTINOU, P., ET AL. 2017. Value of a national administrative database to guide public decisions: From the système national d'information interrégimes de l'Assurance Maladie (SNIIRAM) to the système national des données de santé (SNDS) in France. *Revue d'Épidémiologie et de Santé Publique 65*, S149–S167.

VANASSE, A., COURTEAU, M., AND ETHIER, J.-F. 2018. The "6W" multidimensional model of care trajectories for patients with chronic ambulatory care sensitive conditions and hospital readmissions. *Public Health 157*, 53–61.

VENOT, A., BURGUN, A., AND QUANTIN, C. 2013. *Informatique médicale, e-santé: fondements et applications*. Springer-Verlag France, Paris.

VYAWAHARE, H.R., KARDE, P.P., AND THAKARE, V.M. 2018. A Hybrid Database Approach Using Graph and Relational Database. *2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)*, 1–4.

VYSNIAUSKAS, E., NEMURAITE, L., AND PARADAUSKAS, B. 2012. Preserving Semantics of Owl 2 Ontologies in Relational Databases Using Hybrid Approach. *Information Technology And Control 41*, 2, 103–115.

WEBB, T. AND WILL, D. 2001. Information Integration. In: *Electronic Medical Records: A Guide for Clinicians and Administrators*. ACP Press, 444.

WEEKS, J. AND PARDEE, R. 2019. Learning to Share Health Care Data: A Brief Timeline of Influential Common Data Models and Distributed Health Data Networks in U.S. Health Care Research. *eGEMs 7*, 1.

WELTY, C. AND FIKES, R. 2006. A Reusable Ontology for Fluents in OWL. *Proceedings of the 2006 Conference on Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006)*, IOS Press, 226–236.

WIEDERHOLD, G. 1992. Mediators in the architecture of future information systems. *Computer 25*, 3, 38–49.

WIMALASURIYA, D.C. AND DOU, D. 2010. Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science 36*, 3, 306–323.

WINTRAECKEN, J.V.R. 1990. *NIAM Information Analysis Method: Theory and Practice*. Kluwer Academic Publishers, Norwell, MA, USA.

WYLOT, M., HAUSWIRTH, M., CUDRE-MAUROUX, P., AND SAKR, S. 2018. RDF Data Storage and Query Processing Schemes: A Survey. *ACM Comput. Surv. 51*, 4, 84:1–84:36.

XIAO, G., CALVANESE, D., KONTCHAKOV, R., ET AL. 2018. Ontology-based data access: a survey. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence, 5511–5519.

YANG, C., WANG, X., ZHANG, M., ZHENG, R., WEI, W., AND LOU, Y. 2015. Standardization on bitemporal information representation in BCDM. *2015 IEEE International Conference on Information and Automation*, 2052–2057.

ZAMBORLINI, V. AND GUIZZARDI, G. 2010. On the Representation of Temporally Changing Information in OWL. *2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops*, 283–292.

Table des matières

Table des figures

Table des tableaux