GROUPE DE RECHERCHE INTERDISCIPLINAIRE
EN INFORMATIQUE DE LA SANTÉ

Spécification des exigences système du module « OntoRelA »

Équipe du GRIIS

2024-02-10

Table des matières

Liste des illustrations

Liste des tableaux

Historique

Nom	Modification	Date
Ameni Souid	ménage des dépendances	2024-12-06
Ameni Souid	Correction mineur	2024-12-06
Ameni Souid	renommer en syrs et modification selon notes BF	2024-11-01
Ameni Souid	déplacer sqlGen	2024-10-30
Ameni Souid	Modification	2024-08-27
Christina Khnaisser	Révision	2024-07-20
Ameni Souid	Modification	2024-07-31
Ameni Souid	Modification	2024-06-10
Christina Khnaisser	Conception initiale	2024-02-10

Introduction

Le composant OntoRela est l'atelier de génération d'une base de données relationnelle à partir d'une ontologie.

Objet et portée du document

Le présent document permet de décrire le produit OntoRela. Ses principaux objectifs sont de:

- présenter le contexte dans lequel s'inscrivent le but du projet, les objectifs du développement du produit et les besoins qu'il doit satisfaire;
- présenter et motiver la modélisation nécessaire et suffisante du domaine d'application;
- définir les solutions envisagées;
- définir les exigences applicables à la solution retenue;
- démontrer la rencontre des besoins à l'origine du produit.

Le présent document, une fois complété et approuvé, est la seule référence fonctionnelle applicable sur laquelle l'architecture et la conception du produit pourront être établies. Il s'adresse au maitre d'ouvrage, au maitre d'œuvre, au groupe de l'assurance de la qualité, aux responsables des essais et l'ensemble des membres de l'équipe de développement.

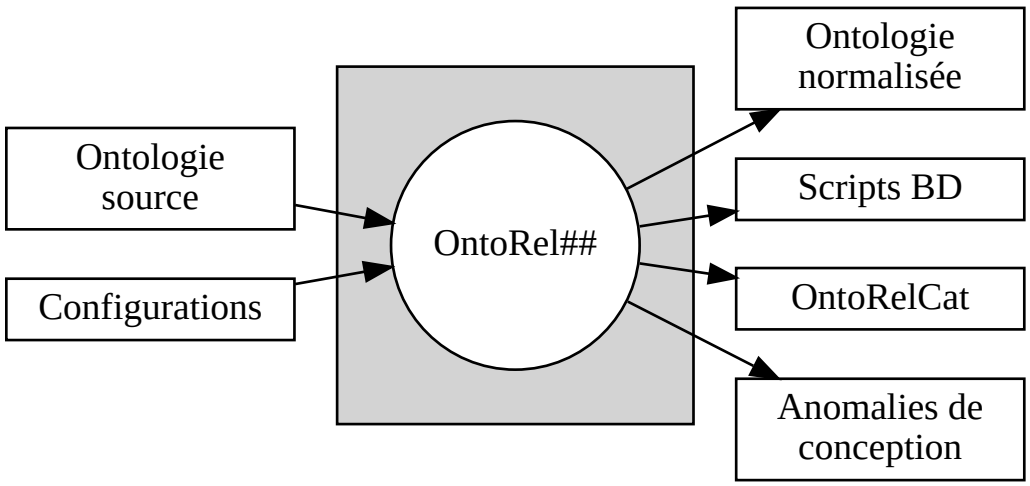
Présentation du problème

Mise en contexte

Pour obtenir l'interopérabilité de modèles de données, il est nécessaire d'élaborer une méthode automatisée de construction de modèle de données à partir d'un modèle de connaissance. 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 permet l'uniformisation de la structure du modèle de données, l'intégration des contraintes du domaine qui proviennent des ontologies.

OntoRelα génère à partir d'une ontologie et une configuration : (1) des scripts pour une base de données relationnelle, (2) des listes d'anomalies de conception, (3) le catalogue du modèle ontologique-relationnel (OntoRelCat), et (4) une ontologie normalisée formalisée selon µOnto. La figure 1 illustre les entrées et les sorties d'OntoRelα.

Figure 1. Diagramme de contexte



Besoins

BE.100 L'utilisateur doit pouvoir utiliser l'atelier de manière automatiquement en s'appuyant sur des algorithmes sur la théorie relationnelle et la logique descriptive.

BE.110 L'utilisateur, en tant qu'architecte de bases de données, doit pouvoir configurer la génération du modèle reltionnel.

BE.120 L'utilisateur doit pouvoir demander à l'atelier de générer un modèle relationnel à partir des entités ontologiques d'intérêts.

BE.130 L'utilisateur doit pouvoir exécuter sur un SGBDR des fichiers SQL générés par l'atelier.

BE.140 L'utilisateur doit pouvoir générer des script SQL selon plusieurs SGBDR.

BE.200 L'utilisateur doit pouvoir établir la correspondance entre un composant de l'ontologie et un composant relationnel via l'atelier.

BE.210 L'utilisateur doit pouvoir exiger que l'atelier normalise les axiomes complexes.

BE.220 L'utilisateur doit pouvoir établir la correspondance entre les types de données OWL et SQL via l'atelier.

BE.230 L'utilisateur doit pouvoir interpréter la documentation du modèle.

Contraintes

CO.10 L'atelier doit être mis en œuvre par des technologies accessibles en source ouvert.

Hypothèses

HY.10 L'utilisateur possède des connaissances minimales des ontologies.

HY.20 L'utilisateur possède des connaissances minimales en normalisation d'une base de données.

Prolongement

PR.10 L'utilisateur doit pouvoir choisir plusieurs SGBDs cibles.

PR.20 L'atelier doit pouvoir mettre à jour un modèle ontologique-relationnel existant.

Exclusions

S.O.

Définitions:

- : C’est un graphe ontologique- relationnel orienté avec les tables comme sommets et (source, déstination et clé de jointure) comme arrêtes.

Présentation de la solution

L'ontologie doit être analysée et filtrée selon une configuration définie (Configuration Ontologie) pour produire une ontologie normalisée. Chaque composant de cette ontologie sera ensuite converti en un composant relationnel, formant ainsi un modèle ontologique relationnel (OntoRel), selon une configuration spécifique (Configuration BDR). Ce modèle relationnel (µRel) servira à générer le code SQL correspondant au SGBD cible.

Les artéfacts externes:

- Configuration de l'ontologie : un fichier de configuration contenant les paramètres d'analyse pour une ontologie.
- Configuration BDR : un fichier de configuration contenant les paramètres de modélisation et de connexion à une base de données relationnelle.
- Ontologie source : une ontologie source est une ontologie au format OWL.
- Anomalies ontologie : un ensemble d'information qui présente des anomalies potentielles provenant de l'ontologie.
- Ontologie normalisée : l'ontologie normalisée en format OWL.
- Anomalies BDR : un ensemble d'information qui présente des anomalies potentielles causées par le mécanisme de conversion.
- OntoRelCat : le catalogue de correspondance entre un composant ontologique et un composant relationnel.
- BDR : un ensemble de scripts exécutable sur un SGBD pour construire la base de données.

Analyse ontologie

Voir la spécification de MOnto [../../monto-library/srs/monto_library-srs.adoc] pour plus de détails.

Intrants

Configuration de l'ontologie Ontologie source

Extrants

µOnto Anomalies ontologie

Génération MOnto

Voir la spécification de MOnto [../../monto-library/srs/monto_library-srs.adoc] pour plus de détails.

Intrants

µOnto

Extrants

Ontologie normalisée

Construction OntoRel

Le processus de conversion OntRel se déroule en cinq étapes :

- Création de l'OntoRel: traduction de l'ontologie µOnto en une représentation conforme au modèle µRel.
- 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.
- Création du graphe relationnel (RelGraph)
- Création du graphe ontologique-relationnel (OntoRelGraph).

Intrants

Configuration BDR µOnto

Extrants

µRel OntoRel

Algorithmes

Règles générales de conversion des composants µOnto en µRel

Les composants ontologiques sont convertis en composants relationnels selon les règles décrites ci-dessous.

Classe

Une classe ontologique est convertie en une table 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 table Thing qui contient les identifiants de tous les individus ontologiques représentés dans la base de données.

Type

Un type de données ontologique est converti en une table 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.

Propriété de classe

Une propriété de classe est convertie en une table 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 table qui représente le domaine et une clé référentielle vers la table 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 table de la sous-classe vers la table de la superclasse.

Axiome d'héritage de propriété

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

Axiome d'association de classe

Un axiome d'association de classe est converti en une table comportant deux attributs : la clé candidate de la table qui représente le domaine de l'axiome (table du domaine) et la clé candidate de la table qui représente la portée de l'axiome (table de la portée). La clé candidate est composée des deux attributs. De plus, trois clés référentielles sont définies : une clé référentielle vers la table du domaine, une clé référentielle vers la table de la portée et une clé référentielle vers la table 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 table comportant deux attributs : la clé candidate de la table du domaine et la clé candidate de la table de la portée. La clé candidate est composée des deux attributs. De plus, deux clés référentielles sont définies : une clé référentielle vers la table du domaine et une clé référentielle vers la table 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 converti 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 table de classe. La projection inclut le renommage de nom des attributs de classe table 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 (table) 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 », « ($\beta \cap \gamma$) = Z » ou « Z = ($\beta \cup \gamma$) » selon le cas, assurera de facto la contrainte correspondante « p $\beta \subseteq$ Z », « ($\beta \cap \gamma$) $\subseteq$ Z » ou « Z $\subseteq$ ($\beta \cup \gamma$) ». Les contraintes peuvent donc être ignorées lors de la conversion.

Règles spécifiques de conversion des composants µOnto en µRel

Règles spécifiques de conversion sans la génération de la Table Thing

La non-conversion de la table Thing implique des changements aux règles générales de conversion. Les différences sont présentées ci-dessous :

- Axiome d'héritage de classe: quand la superclasse est Thing, la contrainte référentielle avec propagation depuis la table de la sous-classe vers la table de la superclasse ne sera pas créée.
- Axiome d'héritage de propriété: quand la super-propriété est Thing, la contrainte référentielle avec propagation depuis la table de la sous-propriété vers la table de la super-propriété ne sera pas créée.
- Axiome d'association de classe : Si Thing fait partie d'un axiome d'une association de classe (domaine ou portée) la clé référentielle vers la table ne sera pas créée. Donc le nombre des clés référentielles ne sera pas forcément trois.

Règles spécifiques de conversion 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. Plus spécifiquement, pour chaque axiome d'association : (1) la clé primaire de la table de la portée doit être ajoutée à la table du domaine et (2) une contrainte référentielle doit être ajoutée depuis la table du domaine vers la table de la portée.

Règles spécifiques de conversion des types (normalizeDatatype= false)

Une conversion spécifique est possible pour les types de données avec la configuration normalizeDatatype= false 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 table d'association de données. Plus précisément, pour chaque axiome d'association de données, il s'agit d'ajouter à la table 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.

Génération OntoRelCat

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.

Intrants

OntoRel

Extrants

OntoRelCat

Mise en correspondance entre les identifiants ontologiques(IRIs) et les identifiants relationnels (id de tables)

(voir OntoRelCat-ldm_SES)

Génération 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 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).

1. Traduction des restrictions µRel en PostgreSQL

µRel	SQL
Schema	CREATE SCHEMA
Description	COMMENT ON
Base relvar (Table)	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

Intrants

μRel

Extrants

BDR Anomalies BDR

Spécification des exigences

Génération d'ontologie

EX.100 Satisfait: [BE.100], [BE.210] Dépendance: Aucune Le composant doit générer une ontologie normalisée et filtrée selon les classes sélectionnées.

Génération OntoRel

EX.001 Satisfait: [BE.110], [BE.210] Dépendance: Aucune Le composant doit permettre de choisir la génération ou non des propriétés d'objet.

EX.002 Satisfait: [BE.210] Dépendance: Aucune Le composant doit permettre de choisir le mode de normalisation des types.

EX.004 Satisfait: [BE.120] Dépendance: Aucune Le composant doit permettre de sélectionner le mode de génération de la table «Thing».

EX.005 Satisfait: [BE.110] Dépendance: Aucune Le composant doit permettre de sélectionner le mode de génération des identifiants des tables.

EX.006 Satisfait: [BE.220] Dépendance: [EX.100] Le composant doit permettre de faire la correspondance entre les types OWL et les types SQL selon le SGBD (???, ???).

EX.007 Satisfait: [BE.140] Dépendance: Aucune Le composant doit permettre de générer des scripts indépendants des dialectes SQL.

Génération OntoRelCat

EX.010 Satisfait: [BE.200] Dépendance: [EX.006] Le composant doit fournir un dictionnaire d'arrimage entre un composant ontologie et un ou plusieurs composants relationnels.

EX.020 Satisfait: [BE.200], [BE.220] Dépendance: [EX.100] Le composant doit générer un graphe ontologique relationnel ([OntoRelGraph]).

Génération SQL

EX.200 Satisfait: [BE.130], [BE.140] Dépendance: [EX.006], [EX.007] L'application doit permettre la génération de scripts SQL complets couvrant la création de schémas, de tables, de vues, de fonctions, et de contraintes référentielles.

EX.210 Satisfait: [BE.130] Dépendance: [EX.200] Les scripts SQL générés doivent être exportables dans des fichiers `.sql`.

Génération des représentations

EX.300 Satisfait: [BE.230] Dépendance: [EX.020] L'application doit permettre de générer des représentations graphiques de l'ontologie et de la base de données.

Génération des anomalies

EX.400 Satisfait: [BE.210], [BE.220] Dépendance: [EX.100], [EX.300] L'application doit permettre de générer les anomalies de l'ontologie et de la base de données.

Matrice exigences et besoins

Tableau 1. Tableau de matrice exigences/besoins

EX/BE	BE.100	BE.110	BE.120	BE.130	BE.140	BE.200	BE.210	BE.220	BE.230
EX.001		X					X		
EX.002							X		
EX.004			X						
EX.005		X							
EX.006								X	
EX.007					X				
EX.010						X		X	
EX.020						X		X	
EX.100	X						X		
EX.200				X	X				
EX.210				X					
EX.300									X
EX.400							X	X	

Annexe

Exemples de conversion

Figure 2. Exemple de conversion

Diagramme d'activité

Figure 3. Diagramme d'activité OntoRelA

Correspondance entre les types OWL et postgreSQL

```
datatypes:
  # Numbers
  "real": REAL
  "double": "DOUBLE PRECISION"
  "float": REAL
  "rational": REAL
  "decimal": DECIMAL
  "integer": INTEGER
  "nonNegativeInteger": INTEGER
  "nonPositiveInteger": INTEGER
  "positiveInteger": INTEGER
  "negativeInteger": INTEGER
  "long": "DOUBLE PRECISION"
  "int": INTEGER
  "short": SMALLINT
  "byte": BYTEA
  "unsignedLong": "DOUBLE PRECISION"
  "unsignedInt": INTEGER
  "unsignedShort": SMALLINT
  "unsignedByte": BYTEA
  # Text
  "string": TEXT
  "Literal": TEXT
  # Bool
  "boolean": BOOLEAN
  # Date and time
  "date": DATE
  "dateTime": TIMESTAMP
  "dateTimeStamp": TIMESTAMP
```

Correspondance entre les types OWL et MSSQL

```
datatypes:
  # Numbers
  "real": REAL
  "double": BIGINT
  "float": FLOAT
```

```
"rational": REAL
"decimal": DECIMAL
"integer": INT
"nonNegativeInteger": INT
"nonPositiveInteger": INT
"positiveInteger": INT
"negativeInteger": INT
"long": BIGINT
"int": INT
"short": SMALLINT
"byte": TINYINT
"unsignedLong": BIGINT
"unsignedInt": INT
"unsignedShort": SMALLINT
"unsignedByte": TINYINT
# Text
"string": VARCHAR
"Literal": TEXT
# Bool
"boolean": BIT
# Date and time
"date": DATE
"dateTime": DATETIME
"dateTimeStamp": DATETIME
```

Représentation du modèle relationnel(µRel)

La librairie doit pouvoir définir les composants essentiels du modèle relationnel, notamment les tables, les domaines, les clés primaires et étrangères, et les contraintes de validation. Cela permettra une création cohérente et standardisée des modèles relationnels à travers différents projets.

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 l'entête 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 définis. Une base de données est valide si et seulement si toutes les contraintes sont avérées.

Références

Khnaisser, Christina. « Construction de modèles de données relationnels temporalisés guidée par les ontologies ». Université de Paris cotutelle Université de Sherbrooke, 2019. http://www.theses.fr/s177273.

Khnaisser, Christina, Luc Lavoie, Benoit Fraikin, Adrien Barton, Samuel Dussault, Anita Burgun, et Jean-François Ethier. « Using an Ontology to Derive a Sharable and Interoperable Relational Data Model for Heterogeneous Healthcare Data and Various Applications ». Methods of Information in Medicine, 16 juin 2022, a-1877-9498. https://doi.org/10.1055/a-1877-9498.

Spécification d'architecture d'ontorela ontorela-application-sas [ontrela-application-sas.adoc]. Spécification des exigences de MOnto MOnto [../../monto-library/srs/monto_library-srs.adoc].