CSRI TECHNICAL REPORTS

The PRAXICON database

Athens, 2014

This document was prepared by:

Dimitris Mavroeidis and Katerina Pastra, Cognitive Systems Research Institute (CSRI).

Aknowledgements:

Panagiotis Dimitrakis, Cognitive Systems Research Institute (CSRI).

Eirini Balta, Cognitive Systems Research Institute (CSRI).

Argiro Vataki, Cognitive Systems Research Institute (CSRI).

Giorgos Karakatsiotis.

schemaSpy[1], which contributed to the creation of the E-R diagrams.

To be cited as:

The PRAXICON database, version 1.0, CSRI Technical Reports, 2014.

This document is available:

On github as part of the PRAXICON database software[2].

On the CSRI website[3].

[1] http://schemaspy.sourceforge.net/

[2] http://www.github.com/.....

[3] http://www.csri.gr/.....

Table of Contents

Abstract **5**
Introduction **5**
The PRAXICON **5**
The PRAXICON database **6**
Implementation **6**
Tables **7**
Relationships 8
Table: Concepts 9
Columns 9
Indexes 10
Relationships 11
Table: LanguageRepresentations 12
Columns 12
Indexes 12
Relationships 14
Table: Compositionality 15
Columns 15
Indexes 15
Relationships 16
Table: Constituents 17
Columns 17
Indexes 17
Relationships 18
Table: Concepts_LanguageRepresentations 19
Columns 19
Relationships 20
Table: VisualRepresentations 21
Columns 21
Indexes 21
Relationships 22
Table: MotoricRepresentations 23
Columns 23
Indexes 23
Relationships 24
Table: Relations 25
Columns 25
Indexes 25
Relationships 26
Table: RelationTypes 27
Columns 30
Indexes 30
Relationships 31
Table: RelationChains 32
Columns 32
Indexes 32
Relationships 33
Table: RelationChains_Relations 34
Columns 34
Indexes 34

Relationships 35
Table: IntersectionsOfRelationChains 36
Columns 36
Indexes 36
Relationships 37
Table: IntersectionsOfRelationChains_RelationChains 38
Columns 38
Indexes 38
Relationships 39
Table: Verbalizations 40
Columns 40
Indexes 40
Relationships 41
Table: LanguageRepresentations_IntersectionsOfRelationChains 42
Columns 42
Indexes 42
Relationships 43
Table: LanguageRepresentations_RelationObjects 44
Columns 44
Indexes 44
Relationships 45
Table: LanguageRepresentations_RelationSubjects 46
Columns 46
Indexes 46
Relationships 47
Table: LanguageRepresentations_RelationChains 48
Columns 48
Indexes 48
Relationships 49
Table: MotoricRepresentations_RelationObjects 50
Columns 50
Indexes 50
Relationships 51
Table: MotoricRepresentations_RelationSubjects 52
Columns 52
Indexes 52
Relationships 53
Table: VisualRepresentations_RelationObjects 54
Columns 54
Indexes 54
Relationships 55
Table: VisualRepresentations_RelationSubjects 56
Columns 56
Indexes 56
Relationships 57
Restrictions 58
Implied (Foreign keys) 58
Imposed 58

Abstract

This document presents the structure of the PRAXICON database. The PRAXICON is a knowledge base designed to be used by artificial agents. The structure of the database directly reflects the theoretical background of the PRAXICON. It is normalised to the third normal form.

Introduction

Long-term memory is divided into declarative and procedural [Anderson, 1976]. Declarative memory refers to memories that are explicitly stored and retrieved, while procedural memory refers to skills acquired by repetition. Declarative memory is further divided into episodic and semantic (explicit knowledge). Episodic memory refers to memories of past events within the context of a particular time and space frame.

Semantic memory is independent of context and encodes abstract knowledge about the world, or –from the linguistic perspective– provides meaning. Thus far, robots have episodic and procedural memory. The PRAXICON tries to fill in the gap by introducing a semantic memory for artificial agents.

The PRAXICON

The PRAXICON is a computational resource which associates symbolic representations (concepts) with corresponding linguistic and sensorimotor representations, and patterns of their combinations that formulate conceptual structures at different levels of abstraction. The resource has been developed to allow artificial agents/systems:

- to tie concepts/words of different levels of abstraction to their sensorimotor instantiations (catering thus for disambiguation), and
- to untie sensorimotor representations from their physical specificities correlating them to conceptual structures of different levels of abstraction (catering thus for intentionality indication). [Praxicon Paper?]

The PRAXICON's core entity is "Concept". A "Concept" is not just a motoric representation; it is an embodied concept representations of perceptual, motoric and/or linguistic/symbolic nature, perceived and stored in memory for behaviour generation and understanding. We consider action ('praxis' in Greek) to be central not only for the motoric system and its representation, but for the integration of the latter with other modules of the cognitive system, such as perception and language.

In PRAXICON, "Concepts" are representations of any type (e.g. visual, symbolic etc.) perceived by a cognitive system and stored in memory. Analysis and reasoning over these representations as they get perceived takes place and its results are also stored in memory. It's a process of meaning and intentionality understanding.

The PRAXICON database

Concepts in PRAXICON can have language, motoric and visual representations. Two concepts can form a relation. Concepts, relations and more complex structures are represented in the database built to provide the computational base on which applications can be built to be used by artificial agents. In what follows, the structure of the database is explained in detail.

First, a list of all the tables in the database is provided and the entity-relationship (E-R) diagram is laid out. For each table in the database, we provide a short description; a table containing the columns and their description; a table containing the indexes; and a partial E-R diagram that depicts the current table and the tables that are directly connected to it. Tables share one-to-many relationships, unless otherwise stated.

Implementation

The database application programming interface (API) was developed using the Java programming language. The Java Persistence API (JPA) v.2.1 is used to create and manage the database entities.

Tables

Name
Concepts
LanguageRepresentations
Compositionality
Constituents
Concepts_LanguageRepresentations
VisualRepresentations
MotoricRepresentations
Relations
RelationTypes
RelationChains
RelationChains_Relations
IntersectionsOfRelationChains
IntersectionsOfRelationChains_RelationChains
Verbalizations
LanguageRepresentations_IntersectionsOfRelationChains
LanguageRepresentations_RelationObjects
LanguageRepresentations_RelationSubjects
LanguageRepresentations_RelationChains
MotoricRepresentations_RelationObjects
MotoricRepresentations_RelationSubjects
VisualRepresentations_RelationObjects
VisualRepresentations_RelationSubjects

Relationships

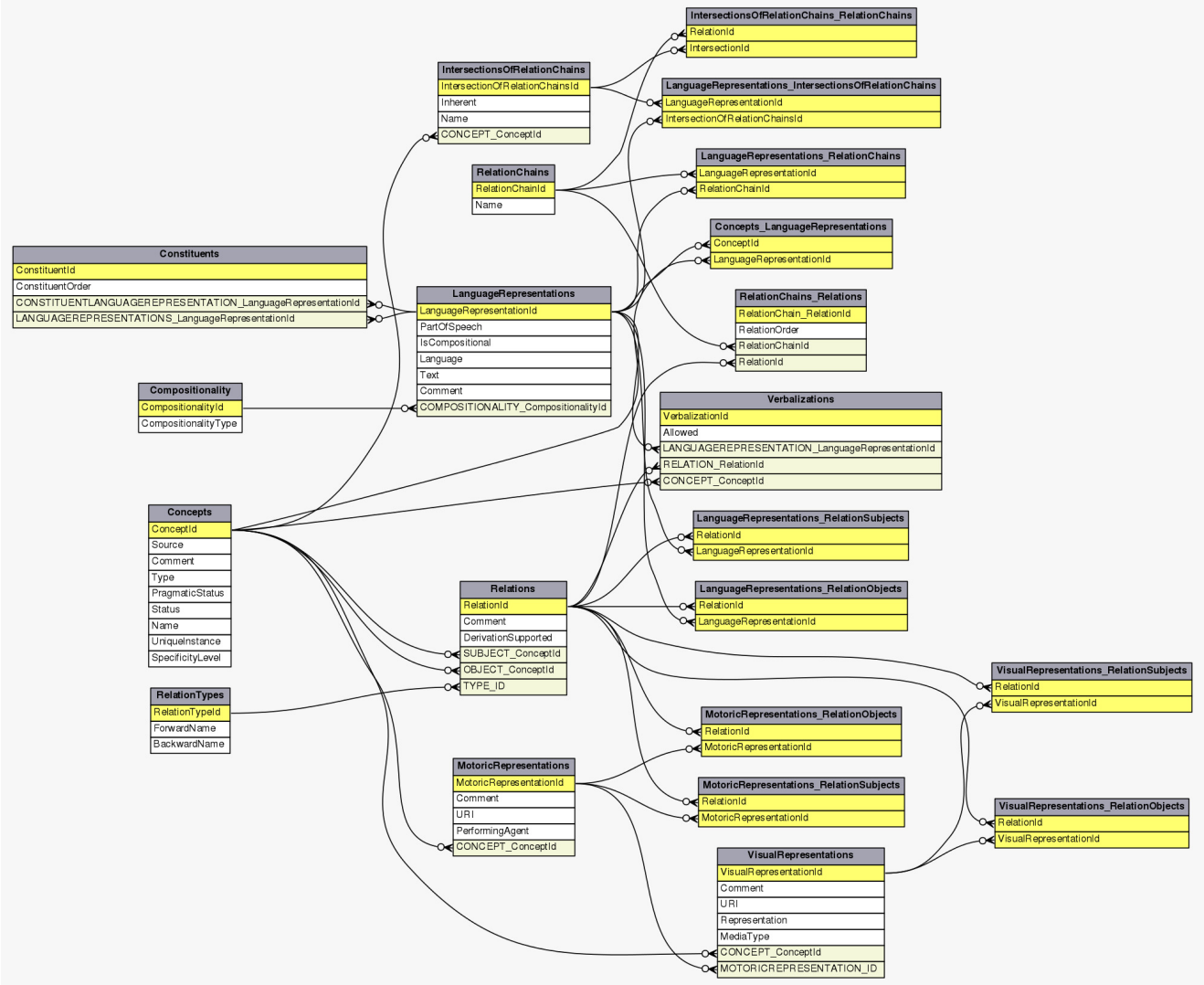


Table: Concepts

The concept is the main entity of the database. A concept can be a physical entity –such as an object or an animal-, a movement, an abstract notion (e.g. democracy, poverty) or a feature (something that characterizes another concept).

Columns

Name	Type	Description
ConceptId	bigint	Primary key (automatically generated).
Source	varchar(255)	Defines the source of the concept (if it has been added manually, from WordNet, or another way).
Comment	varchar(255)	Short description of the concept in plain text; it can be used to store extra information.
Type	varchar(255)	We recognize four types of concepts, i.e. movement, entity, feature, abstract. **Permitted values**: *ABSTRACT, ENTITY, FEATURE, MOVEMENT, UNKNOWN.*
PragmaticStatus	varchar(255)	Defines whether the concept is a figurative or a literal one. **Permitted values**: *FIGURATIVE, LITERAL, UNKNOWN.*
Status	varchar(255)	A Concept can be *constant, variable* or *template*. Concepts marked as Variables are concepts whose value has not been resolved or ones that are inherently related to a variable concept. These are concepts waiting for some reasoning to get appropriate values during application runtime. Only their concept type and their relation to other concepts are known. Example: The "*cut_dummyTool_bread'#movement*" concept is connected with a "tool" variable concept. A reasoner should search for an "entity" concept that could fill in this variable i.e. take up the role of 'tool' for the specific concept. Many entities could take up such role (e.g. knife, hands etc.); therefore the resolution takes place within the application, i.e. when the PRAXICON is used within an embodied cognition application such as a language-based human-robot interaction session. In such cases, the perceptual context along with a number of concepts and language-related parameters are taken into consideration for finding the optimal resolution. Templates are Concepts that define a pattern. For example: the "*cut_something_with_a_tool'#movement*" is a template and its relations define what we can use to cut something (in our example the "*cut_something_with_a_tool'#movement*" concept should be connected with the "knife" concept as a tool and bread as an object of interaction). **Permitted values**: *CONSTANT, VARIABLE, TEMPLATE*.
Name	varchar(255)	Human-friendly name for the concept.

Name	Type	Description
UniqueInstance	varchar(255)	Defines whether the concept is unique in the database. **Permitted values**: *YES, NO, UNKNOWN*.
SpecificityLevel	varchar(255)	Defines the specificity level of the concept, i.e. below-basic-level, basic-level or above-basic-level. **Permitted values**: *BASIC_LEVEL, SUPERORDINATE, SUBORDINATE, UNKNOWN*.

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	ConceptId	

Relationships

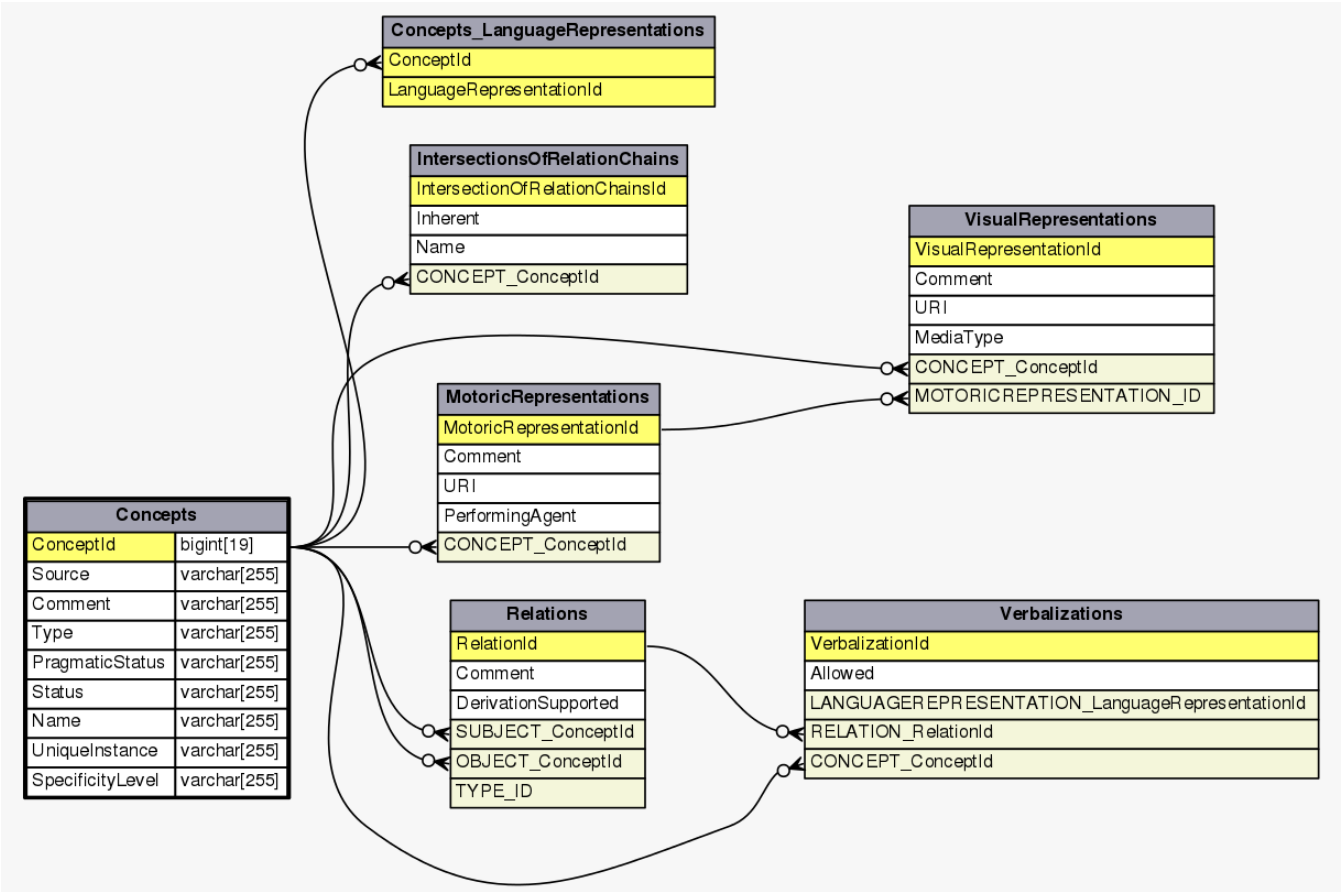


Table: LanguageRepresentations

A language representation of a concept is its linguistic manifestation (a word or expression). There is a many-to-many relationship between LanguageRepresentations and Concepts.

Columns

Name	Type	Description
LanguageRepresentationId	bigint	Primary key (automatically generated).
PartOfSpeech	varchar(255)	Defines the part of speech for the language representation of a concept. **Permitted values**: *ADJECTIVE, ADVERB, NOUN, PARTICIPLE, PROPER_NOUN, VERB*.
IsCompositional	varchar(255)	Defines whether the language representation consists of one or more constituents. **Permitted values**: *YES, NO, UNKNOWN*.
Language	varchar(255)	Defines the language of the language representation. **Permitted values**: *AB, AA, AF, AK, SQ, AM, AR, AN, HY, AS, AV, AE, AY, BM, BA, EU, BE, BN, BH, BI, BS, BR, BG, MY, CA, CH, CE, NY, ZH, CV, KW, CO, CR, HR, CS, DA, DV, NL, DZ, EN, EO, ET, EE, FO, FJ, FI, FR, FF, GL, KA, DE, EL, GN, GU, HT, HA, HE, HZ, HI, HO, HU, IA, ID, IE, GA, IG, IK, IO, IS, IT, IU, JA, JV, KL, KN, KR, KS, KK, KM, KI, RW, KY, KV, KG, KO, KU, KJ, LA, LB, LG, LI, LN, LO, LT, LU, LV, GV, MK, MG, MS, ML, MT, MI, MR, MH, MN, NA, NV, NB, ND, NE, NG, NN, NO, II, NR, OC, OJ, CU, OM, OR, OS, PA, PI, FA, PL, PS, PT, QU, RM, RN, RO, RU, SA, SC, SD, SE, SM, SG, SR, GD, SN, SI, SK, SL, SO, ST, AZ, ES, SU, SW, SS, SV, TA, TE, TG, TH, TI, BO, TK, TL, TN, TO, TR, TS, TT, TW, TY, UG, UK, UR, UZ, VE, VI, VO, WA, CY, WO, FY, XH, YI, YO, ZA, ZU*.
Text	varchar(255)	The name of the language representation, usually a word.
Comment	varchar(255)	Short description of the language representation in plain text; it can be used to store extra information.

Indexes

Name	Type	Columns	Description
LanguageRepresentations_CONSTITUENTS_ConstituentId	Non-unique	CONSTITUENTS_ConstituentId	
LnggRepresentationsCMPSTIONALITYCompositionalityId	Non-unique	COMPOSITIONALITY_CompositionalityId	
LnggRprsnttnsLNGGRPRSNTTNCONSTITUENTSConstituentId	Non-unique	LANGUAGEREPRESENTATIONCONSTITUENTS_ConstituentId	

Name	Type	Columns	Description
PRIMARY	Unique	LanguageRepresentationId	

Relationships

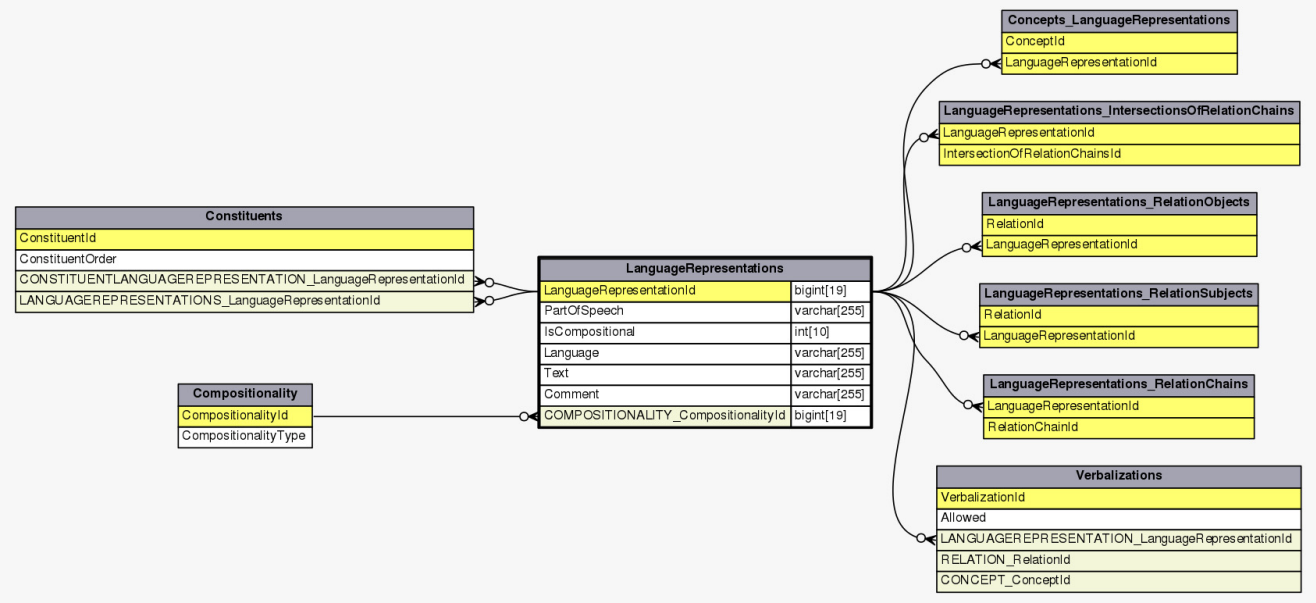


Table: Compositionality

When a language representation consists of 2 or more constituent elements (usually words), we imprint it in this table. We specify the type of compositionality (multi-word or composite) and the language representation it refers to.

This table is closely linked together with the "Constituents" table, which records the parts of the multi-word or composite word (See Table: Constituents).

This table is connected to the LanguageRepresentations table with a one-to-one relationship. This extra step of normalization was necessary since there are not many composite words or multi-words and recording them on the LanguageRepresentations table would create many "NULL" entries and thus create performance issues.

Columns

Name	Type	Description
CompositionalityId	bigint	Primary key (automatically generated).
CompositionalityType	varchar(255)	The type of compositionality. There are currently two types identified, multiword and composite. **Permitted values**: *MULTIWORD, COMPOSITE_WORD, UNKNOWN.*
LANGUAGEREPRESENTATION_LanguageRepresentationId	bigint	Foreign key to LanguageRepresentation table.

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	CompositionalityId	

Relationships

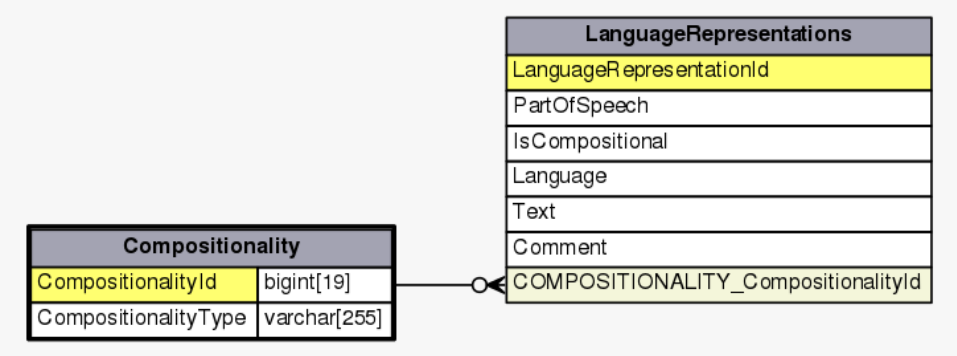


Table: Constituents

This table is filled when a language representation is either a multiword or a composite word. We record the constituents of this language representation. The "*CONSTITUENTSLANGUAGEREPRESENTATIONS_LanguageRepresentationId*" foreign key records one of the constituents of the multiword or composite word, which, in turn, is recorded by the "*LANGUAGEREPRESENTATIONS_LanguageRepresentationId*" foreign key. Both foreign keys point to the "LanguageRepresentations" table.

Columns

Name	Type	Description
ConstituentId	bigint	Primary key (automatically generated).
ConstituentOrder	smallint	The order of the constituent word inside the multiword or composite word.
CONSTITUENTSLANGUAGEREPRESENTATIONS_LanguageRepresentationId	bigint	Foreign key to the LanguageRepresentations table regarding the constituent of the multiword or composite word.
LANGUAGEREPRESENTATIONS_LanguageRepresentationId	bigint	Foreign key to the LanguageRepresenations table regarding the referring language representation (that consists of a multiword or composite word).

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	ConstituentId	

Relationships

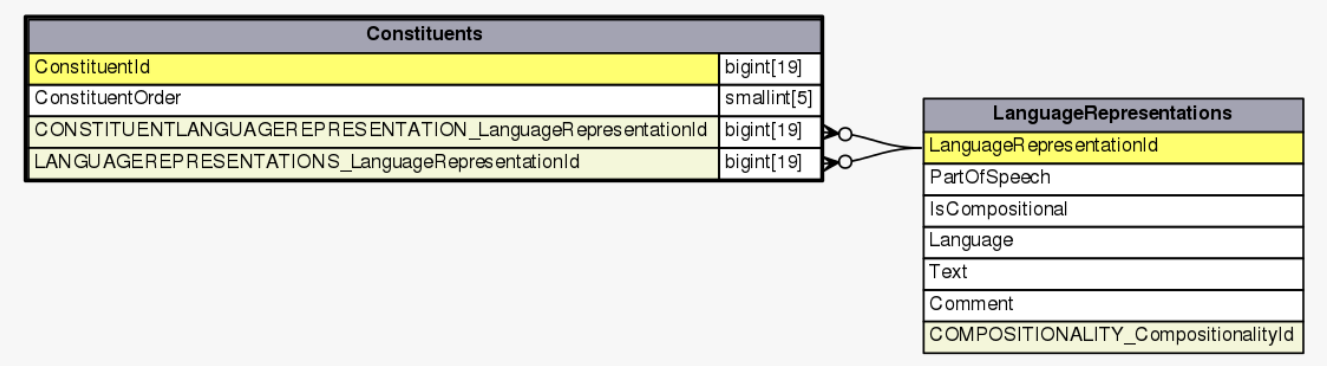


Table: Concepts_LanguageRepresentations

An intermediary table. It breaks the many-to-many relationship between Concepts and LanguageRepresentations table, thus adhering to the 3rd normal form.

Columns

Name	Type	Description
ConceptId	bigint	Composite primary key combining two foreign keys to Concepts and LanguageRepresentations tables.
LanguageRepresentationId	bigint	

Indexes

Name	Type	Columns	Description
CncptLanguageRepresentationLnguageRepresentationId	Non-unique	LanguageRepresentationId	
PRIMARY	Unique	ConceptId, LanguageRepresentationId	

Relationships

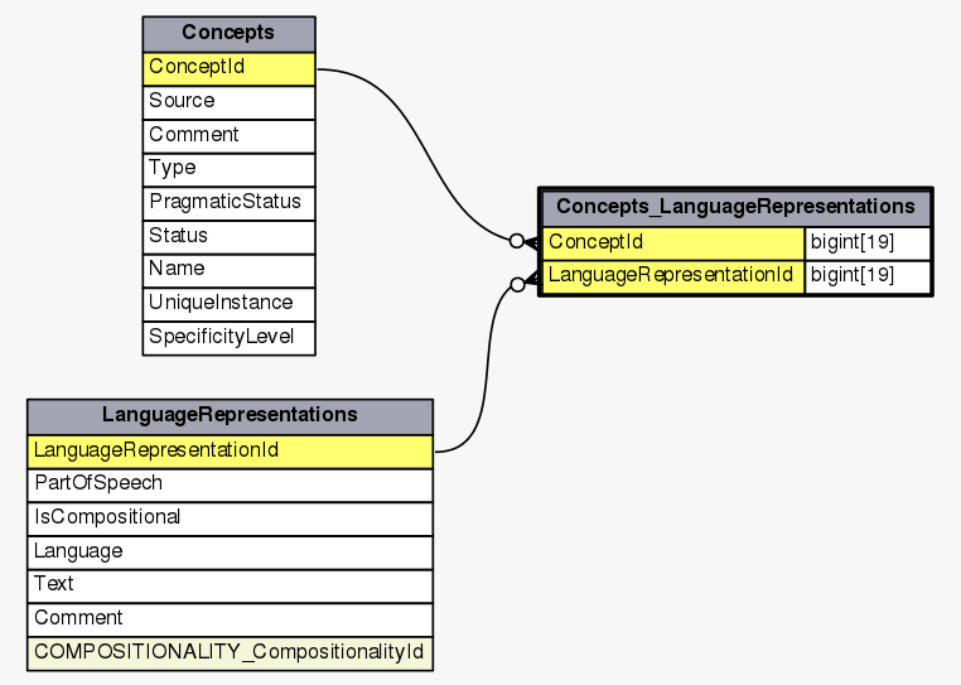


Table: VisualRepresentations

Records the visual manifestation(s) of a concept. Currently, this can be an image or a video. This table is connected to the "Concepts" table with a one-to-many relationship.

Columns

Name	Type	Description
VisualRepresentationId	bigint	Primary key (automatically generated).
Comment	varchar(255)	Short description of the visual representation in plain text; it can be used to store extra information.
URI	blob	The URL (or the path) to the actual file that describes the concept visually.
MediaType	varchar(255)	The type of media of the representation. **Permitted values**: *IMAGE, VIDEO*.
CONCEPT_ConceptId	bigint	Foreign key to Concepts table.
MOTORICREPRESENTATION_ID	bigint	Foreign key to MotoricRepresentations table.

Indexes

Name	Type	Columns	Description
FK_VisualRepresentations_CONCEPT_ConceptId	Non-unique	CONCEPT_ConceptId	
FK_VisualRepresentations_MOTORICREPRESENTATION_ID	Non-unique	MOTORICREPRESENTATION_ID	
PRIMARY	Unique	VisualRepresentationId	

Relationships

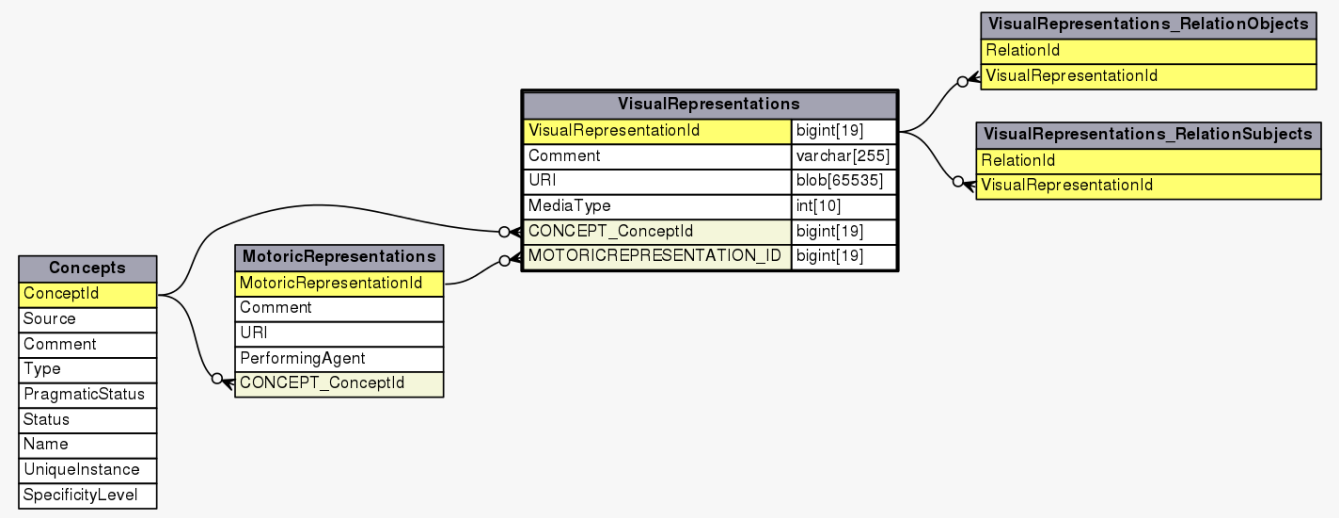


Table: MotoricRepresentations

Records the motoric manifestation(s) of a concept. This table is connected to the "Concepts" table with a one-to-many relationship.

Columns

Name	Type	Description
MotoricRepresentationId	bigint	Primary key (automatically generated).
Comment	varchar(255)	Short description of the motoric representation in plain text; it can be used to store extra information.
URI	blob	The URL (or the path) to the actual file that describes the concept motorically.
PerformingAgent	varchar(255)	The agent that performs the action. **Permitted values** (other values can be added): *ADULT, CHILD, ICUB, PR2, NAO*.
CONCEPT_ConceptId	bigint	Foreign key to Concepts table.

Indexes

Name	Type	Columns	Description
FK_MotoricRepresentations_CONCEPT_ConceptId	Non-unique	CONCEPT_ConceptId	
PRIMARY	Unique	MotoricRepresentationId	

Relationships

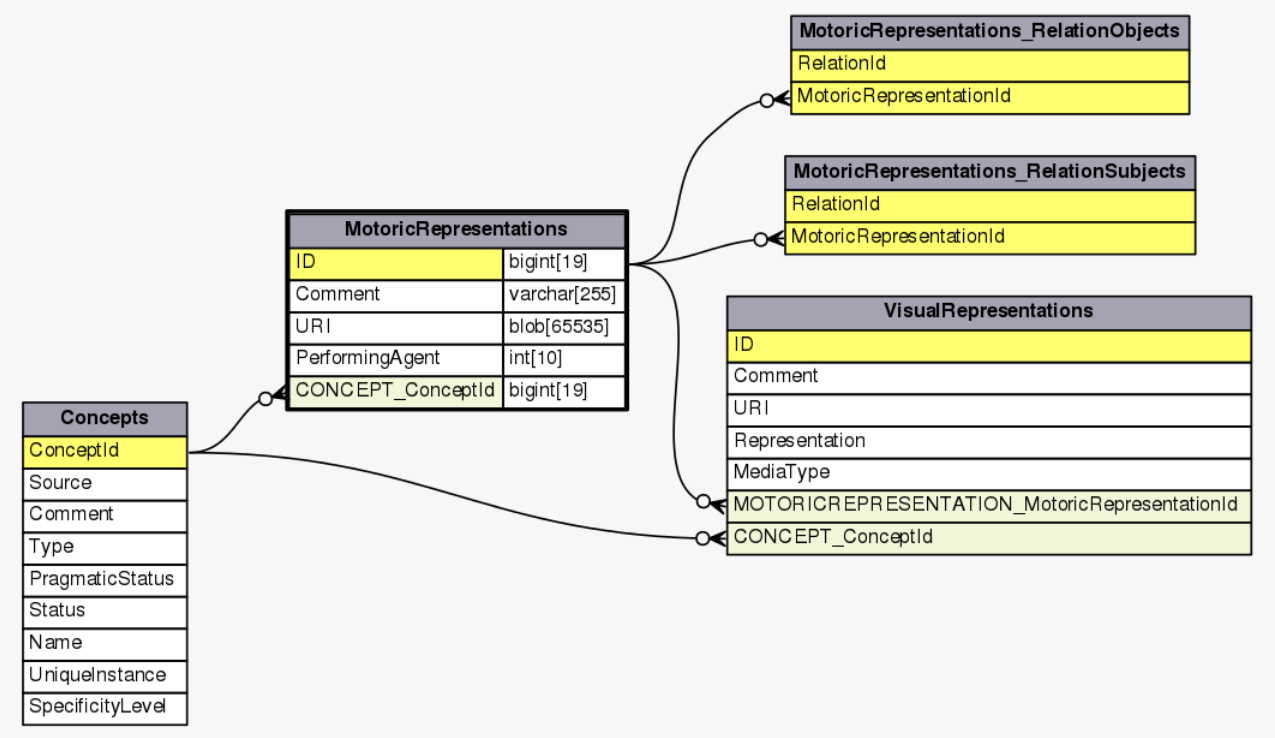


Table: Relations

The relation is at the core of the PRAXICON. A “relation” connects two concepts together semantically. It takes one concept as the “subject” and another one as the “object”. Relation types are explained in the description of the “RelationTypes” table.

Columns

Name	Type	Description
RelationId	bigint	Primary key (automatically generated).
Comment	varchar(255)	Short description of the relation in plain text; it can be used to store extra information.
DerivationSupported	varchar(255)	Whether the relation supports derivation. There is no need to know which language representations are derivationally related. **Permitted values**: *YES, NO, UNKNOWN*.
SUBJECT_ConceptId	bigint	Foreign key to Concepts table which denotes which concept is the subject in this relation.
OBJECT_ConceptId	bigint	Foreign key to Concepts table which denotes which concept is the object in this relation.
TYPE_ID	bigint	Foreign key to the RelationTypes table.

Indexes

Name	Type	Columns	Description
FK_Relations_OBJECT_ConceptId	Non-unique	OBJECT_ConceptId	
FK_Relations_SUBJECT_ConceptId	Non-unique	SUBJECT_ConceptId	
FK_Relations_TYPE_ID	Non-unique	TYPE_ID	
PRIMARY	Unique	RelationId	

Relationships

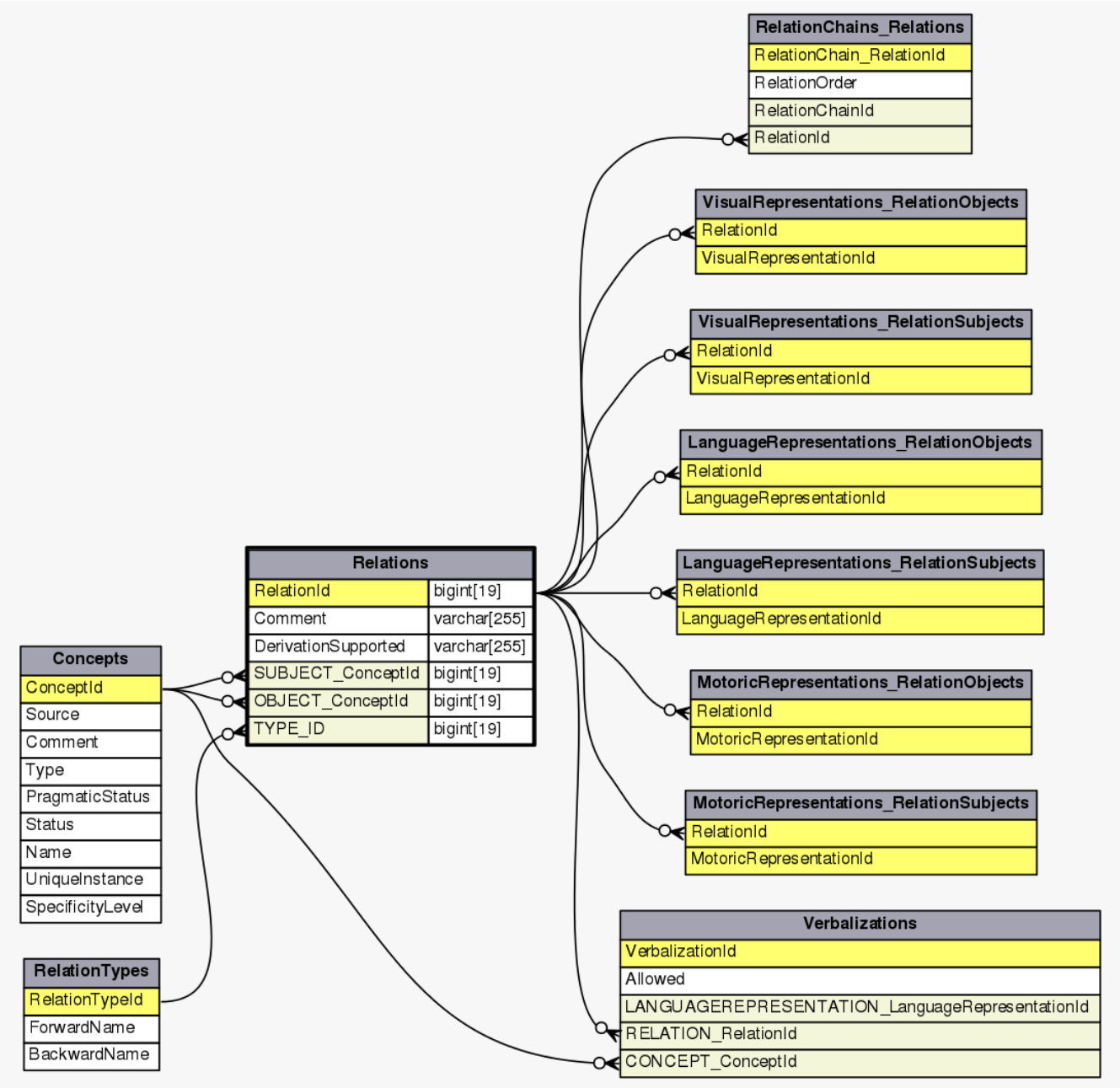


Table: RelationTypes

The relation type defines the nature of a relationship between two concepts. This table is connected to the "Relations" table with a one-to-one relationship. This further normalization step was taken in order to save storage space and for relevant queries to perform faster. Some of the relation types are organized hierarchically (see below):

- Colour
    - hue (e.g., red, red-like)
    - luminance (e.g., dark, light, opaque, transparent etc.)
    - intensity (e.g., vivid, pale, pastel)
    - combo (for cases when more than one subtypes mentioned in one word)
- Condition (e.g., robust, rigid)
- Material (e.g., out of iron)
- Shape
- Size
    - general (e.g., big)
    - length (e.g., long)
    - height (e.g., tall)
    - width (e.g., wide)
    - depth (e.g., deep)
    - combo (for cases when more than one subtypes mentioned in one word)
- Temperature (e.g., warm)
- Texture (e.g., soft)
- Visual pattern (e.g., lines,
- Volume (e.g., bulky, thin, thick etc.)
- Weight (e.g., heavy)
- Force (e.g., tightly).
- Speed Rate (e.g., fast, instantly).

Below, a comprehensive list of all relation types, an example for each type and an explanation where necessary.

Relation	Inverted Relation	Example
ACTION_GOAL	GOAL_ACTION	*'vacuum#with#vacuum_cleaner#the#dummy_object_vacuum_vacuum_cleaner', 'ACTION_GOAL', 'vacuum'*
ACTION_OBJECT	OBJECT_ACTION	*'wash#with#shampoo#the#dummy_object_wash_shampoo', 'ACTION_OBJECT', 'hair'*
ACTION_RESULT	RESULT_ACTION	*'fireball', 'RESULT_ACTION', 'nuclear_explosion'*
ACTION_TOOL	TOOL_ACTION	*'pluck#with#guitar_pick#the#guitar', 'ACTION_TOOL', 'guitar_pick'*
ASPECT_CONCEPT	CONCEPT_ASPECT	*'rhinorrhea', 'ASPECT_CONCEPT', 'common_cold'* This relation denotes an aspect of a concept. For instance, rhinorrhea is an aspect (symptom) of the common cold.
COMPARED_WITH	COMPARED_WITH	*'egg', 'COMPARED_WITH', 'car'* This relation can only be used inside an intersection of relation chains that also contains a relation that denotes the size of one of the concepts, e.g.: *'egg', 'HAS_SIZE', 'small'*.
ENABLES	ENABLED_BY	'incubation', 'ENABLES', 'infection' This example denotes that an infection can be caused after the incubation of a pathogenic organism.
HAS_ANTHROPOGENIC_EFFECT	ANTHROPOGENIC_EFFECT_OF	‘food’, ‘HAS_ANTHOPOGENIC_EFFECT’, ‘processed’
HAS_COLOUR	COLOUR_OF	*‘banana’, ‘HAS_COLOUR’, ‘yellow’*
HAS_CONDITION	CONDITION_OF	*'knock-knee', 'CONDITION_OF', 'leg'*
HAS_CONTENT	CONTENT_OF	*'record_sleeve', 'HAS_CONTENT', 'phonograph_record'*
HAS_DENSITY	DENSITY_OF	*‘air’, ‘HAS_DENSITY’, ‘thin’*
HAS_DEPTH	DEPTH_OF	*‘pool’, ‘HAS_DEPTH’, ‘deep’*
HAS_FORCE	FORCE_OF	*‘huricane’, ‘HAS_FORCE’, ‘strong’*
HAS_HEIGHT	HEIGHT_OF	*‘mountain’, ‘HAS_HEIGHT’, ‘high’*
HAS_HUE	HUE_OF	*‘basket’, ‘HAS_HUE’, ‘orange’*
HAS_INSTANCE	INSTANCE_OF	*'university', 'HAS_INSTANCE', 'harvard_university'* This example can be interpreted like this: *Harvard University belongs to the class of Universities.*
HAS_INTENSITY	INTENSITY_OF	‘wall’, HAS_INTENSITY, ‘pale’
HAS_LENGTH	LENGTH_OF	*‘bow’, ‘HAS_LENGTH’, ‘short’*
HAS_LOCATION	LOCATION_OF	*'loch_ness_monster', 'HAS_LOCATION', 'loch_ness'*
HAS_LUMINANCE	LUMINANCE_OF	‘film’, HAS_LUMINANCE, ‘transparent’
HAS_MATERIAL	MATERIAL_OF	*'cacao_bean', 'MATERIAL_OF', 'chocolate'* An “is-made-of” relation. In the example below, a

		chocolate is made of cacao beans:
HAS_MEASUREMENT_UNIT	MEASUREMENT_UNIT_OF	*'grad', 'MEASUREMENT_UNIT_OF', 'right_angle'*
HAS_MEASUREMENT_VALUE	MEASUREMENT_VALUE_OF	*'light_breeze', 'MEASUREMENT_VALUE_OF', 'wind_scale'*
HAS_MEMBER	MEMBER_OF	*'homeboy', 'MEMBER_OF', 'youth_gang'*
HAS_NATURAL_EFFECT	NATURAL_EFFECT_OF	'under_water_earthquake', 'HAS_NATURAL_EFFECT', 'tsunami'
HAS_PART	PART_OF	*'body_hair', 'PART_OF', 'human'*
HAS_PARTIAL_INSTANCE	PARTIAL_INSTANCE_OF	*'play_back#with#dummy_tool_play_back#the#dummy_object_play_back','HAS_PARTIAL_INSTANCE','play_back#with#tape_player#the#dummy_object_play_back_tape_player'*
HAS_SHAPE	SHAPE_OF	*'tower', 'HAS_SHAPE', 'column'*
HAS_SIZE	SIZE_OF	*'box', 'HAS_SIZE', 'large'*
HAS_SPEED_RATE	SPEED_RATE_OF	'drive#with#dummy_tool_drive#the#dummy_object_drive', HAS_SPEED_RATE, 'fast'
HAS_STEP	STEP_OF	*'deal', 'STEP_OF', 'HAS_STEP', 'card_game'*
HAS_TEMPERATURE	TEMPERATURE_OF	*'water', 'HAS_TEMPERATURE', 'warm'*
HAS_TEXTURE	TEXTURE_OF	*'road', 'HAS_TEXTURE', 'rough'*
HAS_TIME_PERIOD	TIME_PERIOD_OF	*'time-out', 'TIME_PERIOD_OF', 'athletic_game'*
HAS_VISUAL_PATTERN	VISUAL_PATTERN_OF	*'sand', 'HAS_VISUAL_PATTERN', 'handprint'* The idea here is that the sand takes the shape of a hand. Another characteristic example of this relation, is when clouds take the shape of various objects such as dragons, elephants, etc.
HAS_VOLUME	VOLUME_OF	*'dummy_content_of_bowl', 'HAS_VOLUME', 'bowlful'*
HAS_WEIGHT	WEIGHT_OF	*'bag_of_oranges', 'HAS_WEIGHT', 'heavy'*
HAS_WIDTH	WIDTH_OF	'pool', 'HAS_WIDTH', 'narrow'
MORE	no inversion allowed, use NONE in backward naming	apple#entity HAS_Texture VARIABLE_1#feature avocado#entity HAS_TEXTURE VARIABLE_2#feature [VARIABLE_1#feature MORE hard#feature VARIABLE_1#feature COMPARED_WITH VARIABLE_1] <-- these two relations in inherent intersection
LESS	no inversion allowed, use NONE in backward naming	
METAPHOR_OF	METAPHOR_AS	*'knife', 'METAPHOR_OF', 'lithic_blade'* This example denotes that a lithic blade was likened to a knife.
PRODUCER_OF	PRODUCT_OF	*'Microsoft', 'PRODUCER_OF', 'software'*
TYPE_TOKEN	TOKEN_TYPE	*'ant', 'TYPE_TOKEN', 'wood_ant'* In this example, a wood ant is a kind of ant.

Columns

Name	Type	Description
RelationTypeId	bigint	Primary key (automatically generated).
ForwardName	varchar(255)	The name of the relation from left to right.
BackwardName	varchar(255)	The name of the relation if we reverse the position of the two concepts, i.e. we exchange the subject for the object and vice-versa.

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	RelationTypeId	

Relationships

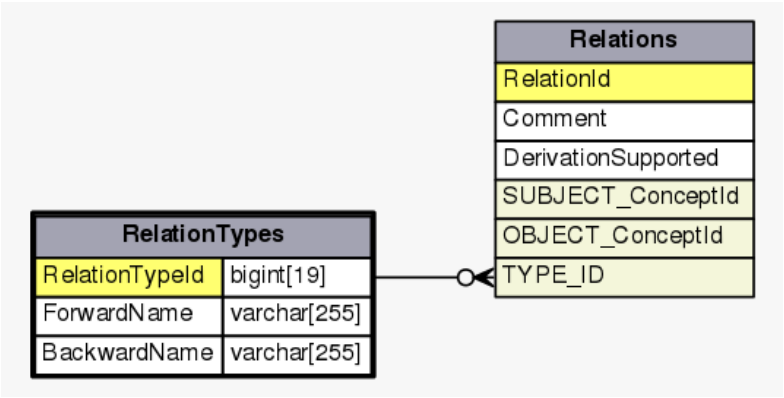


Table: RelationChains

Some relations may be more complex, allowing for the second part/argument to be a whole graph rather than a single concept. We call these “Relation Chains”. For example, the concept ‘butter knife’ is related to two other concepts as follows:

‘butter_knife’, ‘TOOL_ACTION’, ‘spread_withTool_Object’, ‘ACTION_OBJECT’, ‘butter’

In other words, the ‘butter knife’ concept is a tool used for spreading butter with; its direct relation to the ‘spread with tool an object’ concept is inherent and so is its indirect relation to the ‘butter’ concept. No part of this relation chain stands as an inherent relation independently of the whole graph. Constituent relations of a chain follow a fixed order. A relation chain can consist of at least one relation. There is a many-to-many relationship with the “Relations” table.

Columns

Name	Type	Description
RelationChainId	Bigint	Primary key (automatically generated).
Comment	varchar(255)	Short description of the motoric representation in plain text; it can be used to store extra information.

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	RelationChainId	

Relationships

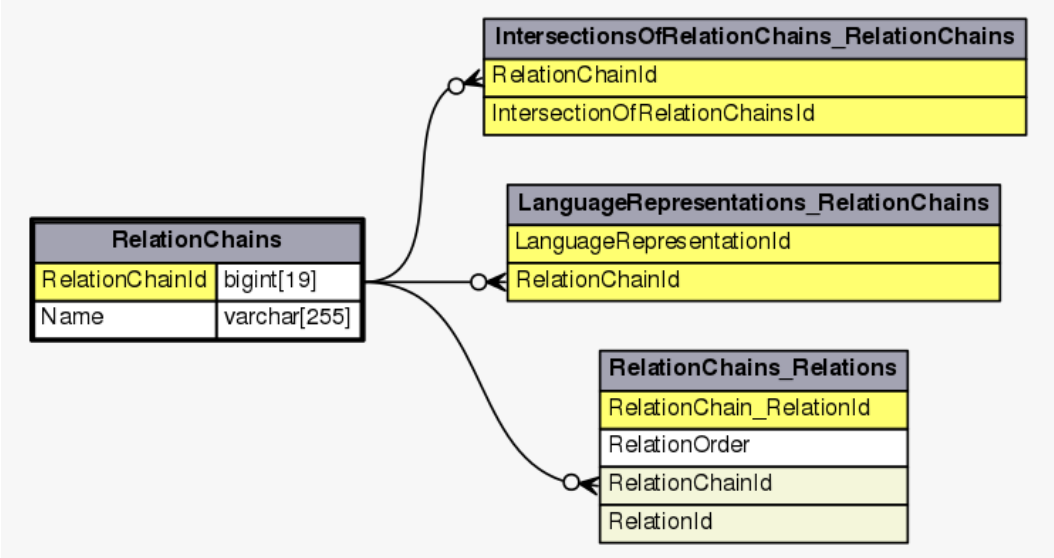


Table: RelationChains_Relations

This table is an intermediary one that breaks the many-to-many relationship between the RelationChains and Relations tables. It additionally contains the order of each relation in the relation chain.

Columns

Name	Type	Description
RelationChain_RelationId	bigint	Primary key (automatically generated).
RelationOrder	bigint	The order of the relation in the relation chain. It starts from 0.
RelationChainId	bigint	Foreign key to the RelationChains table.
RelationId	bigint	Foreign key to the Relations table.

Indexes

Name	Type	Columns	Description
FK_RelationChains_Relations_RelationChainId	Non-unique	RelationChainId	
FK_RelationChains_Relations_RelationId	Non-unique	RelationId	
PRIMARY	Unique	RelationChain_RelationId	

Relationships

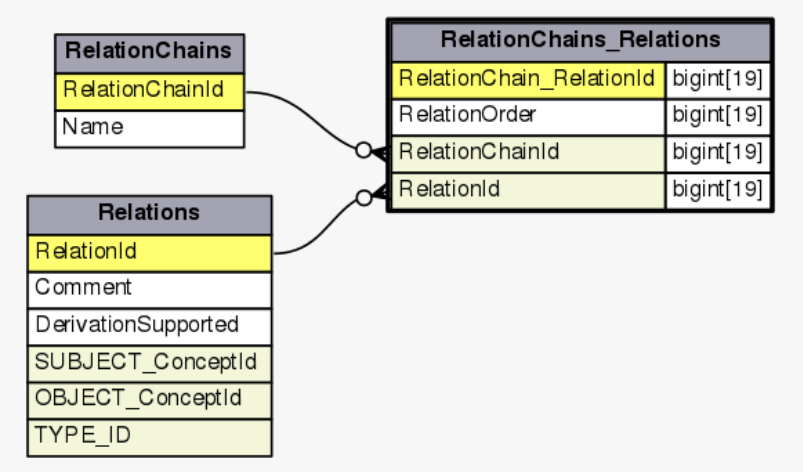


Table: IntersectionsOfRelationChains

Another form of complex relations are ones whose second part/argument is formulated by the union of two or more concepts. In such cases, a Concept A is defined through its relation with two (or more) concepts. For example, the concept "*black and white#feature*" has two type-token relations: one with "black" and another one with "white". But in order for something to be "black and white" both of these relations should be satisfied. We call this an "Intersection of Relation Chains". In other words, Concept A is the intersection of its same-type-relation to a number of other concepts; this is an inherent relation of Concept A to Concept B. There is a many-to-many relationship with the "RelationChains" table.

Columns

Name	Type	Description
IntersectionOfRelationChainsId	bigint	Primary key (automatically generated).
Inherent	varchar(255)	Whether the Intersection is inherent or not. **Permitted values**: *YES, NO, UNKNOWN*.
Comment	varchar(255)	Short description of the Intersection in plain text; it can be used to store extra information.
CONCEPT_ConceptId	bigint	Foreign key to the Concepts table. The concept for which this Intersection is inherent.

Indexes

Name	Type	Columns	Description
FK_IntersectionsOfRelationChains_CONCEPT_ConceptId	Non-unique	CONCEPT_ConceptId	
PRIMARY	Unique	IntersectionOfRelationChainsId	

Relationships

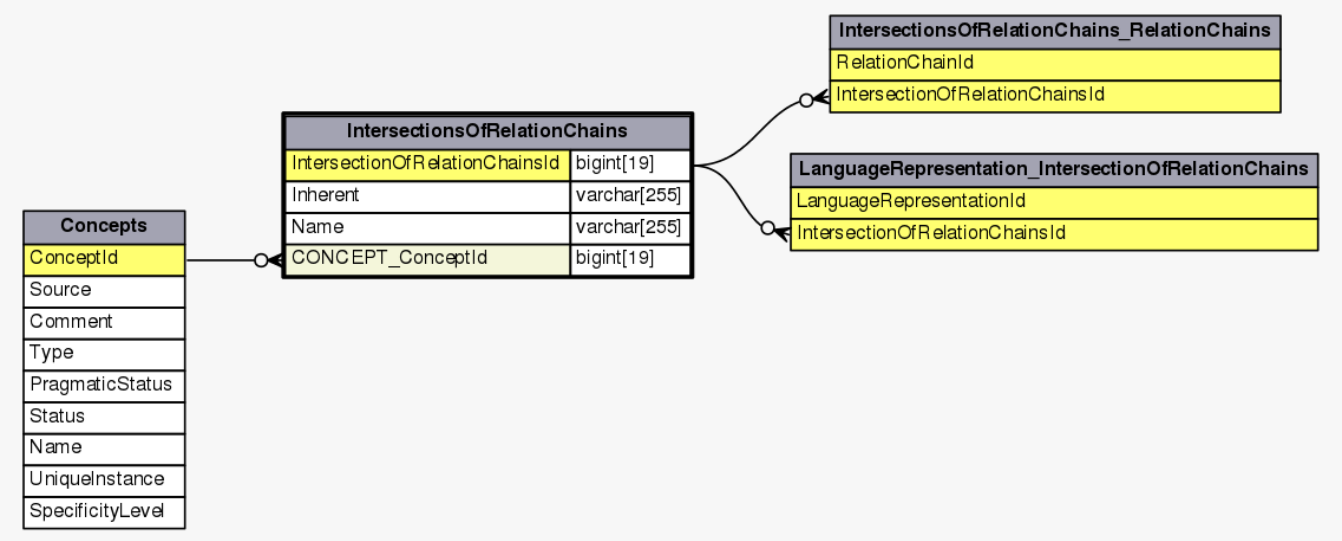


Table: IntersectionsOfRelationChains_RelationChains

An intermediary table. It breaks the many-to-many relationship between “Relations” and “IntersectionsOfRelationChains” table, thus adhering to the 3rd normal form.

Columns

Name	Type	Description
RelationChainId	bigint	Composite primary key combining two foreign keys to Intersections and Relations tables.
IntersectionOfRelationChainsId	bigint	

Indexes

Name	Type	Columns	Description
FK_Intersection_Relation_IntersectionId	Non-unique	IntersectionId	
PRIMARY	Unique	RelationId, IntersectionId	

Relationships

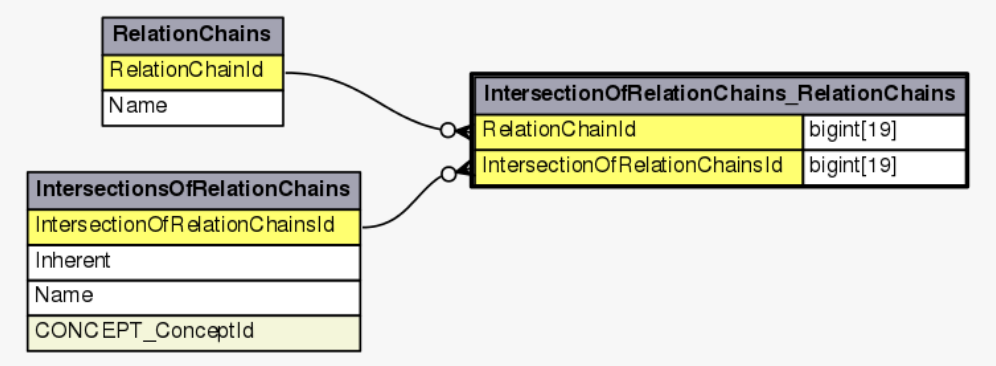


Table: Verbalizations

Sometimes, when two concepts are connected with a relation, one of the language representations of one (or both) of the concepts cannot be used. Thus, a table listing the language representation/relation/concept combinations that are allowed is needed.

Columns

Name	Type	Description
VerbalizationId	bigint	Primary key (automatically generated).
Allowed	varchar(255)	Whether the specific language representation of the concept is allowed in the relation.
LANGUAGEREPRESENTATION_LanguageRepresentationId	bigint	Foreign key to the LanguageRepresentations table.
RELATION_RelationId	bigint	Foreign key to the Relations table.
CONCEPT_ConceptId	bigint	Foreign key to the Concepts table.

Indexes

Name	Type	Columns	Description
FK_Verbalizations_CONCEPT_ConceptId	Non-unique	CONCEPT_ConceptId	
FK_Verbalizations_RELATION_RelationId	Non-unique	RELATION_RelationId	
PRIMARY	Unique	VerbalizationId	
VrblztonsLNGGRPRESENTATIONLanguageRepresentationId	Non-unique	LANGUAGEREPRESENTATION_LanguageRepresentationId	

Relationships

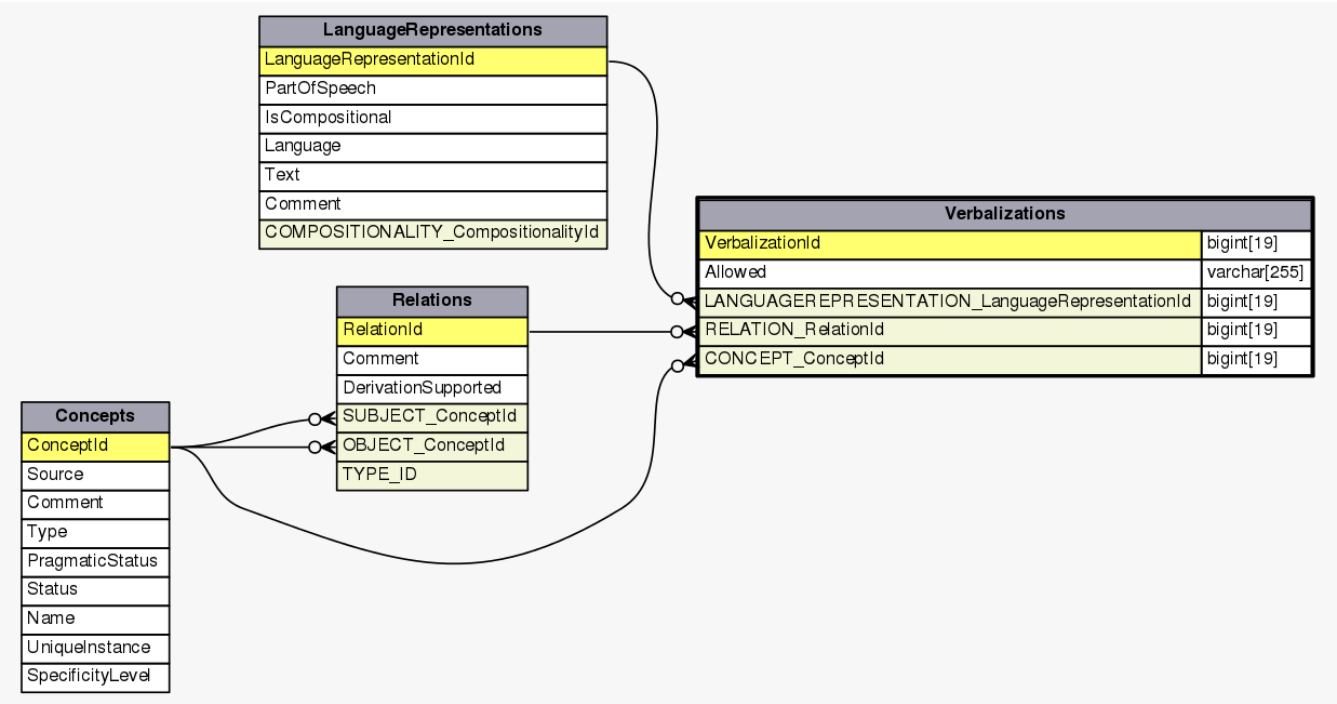


Table: LanguageRepresentations_IntersectionsOfRelationChains

An intermediary table. It breaks the many-to-many relationship between LanguageRepresentations and IntersectionsOfRelationChains table, thus adhering to the 3rd normal form.

Columns

Name	Type	Description
LanguageRepresentationId	bigint	Composite primary key combining two foreign keys to LanguageRepresentations and IntersectionOfRelationChains tables.
IntersectionOfRelationChainsId	bigint	

Indexes

Name	Type	Columns	Description
LnggRprsnttnntrsctnfRlationChainsntrsctnfRltnChnsd	Non-unique	IntersectionOfRelationChainsId	
PRIMARY	Unique	LanguageRepresentationId, IntersectionOfRelationChainsId	

Relationships

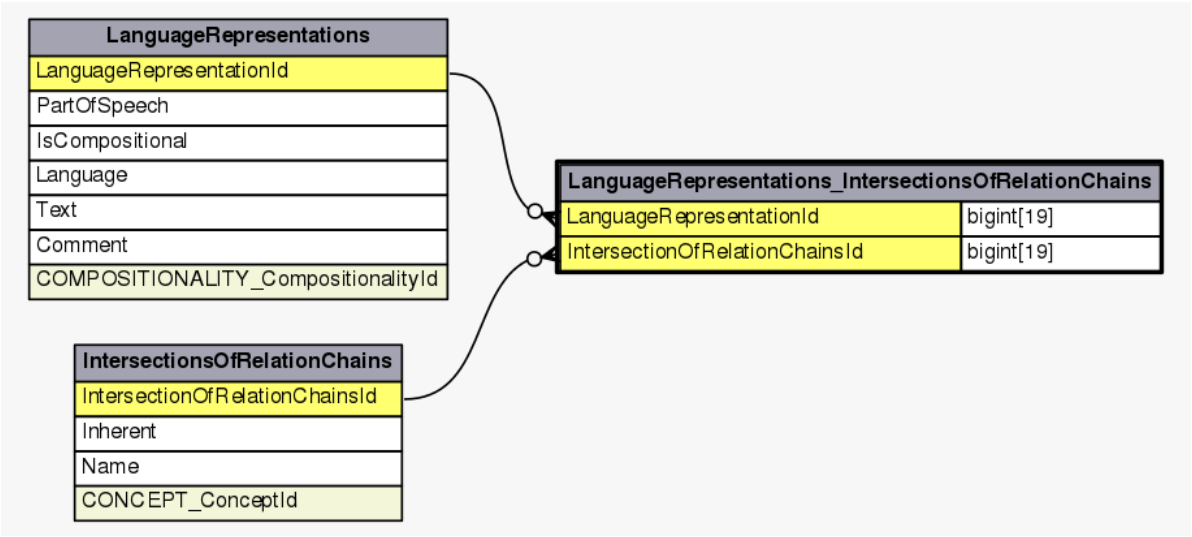


Table: LanguageRepresentations_RelationObjects

An intermediary table. It breaks the many-to-many relationship between LanguageRepresentations and Relations table with respect to the relation's object, thus adhering to the 3rd normal form.

Columns

Name	Type	Description
RelationId	bigint	Composite primary key combining two foreign keys to Relations and LanguageRepresentations tables.
LanguageRepresentationId	bigint	

Indexes

Name	Type	Columns	Description
LnggRpresentationRelationObjectLnggRpresentationId	Non-unique	LanguageRepresentationId	
PRIMARY	Unique	RelationId, LanguageRepresentationId	

Relationships

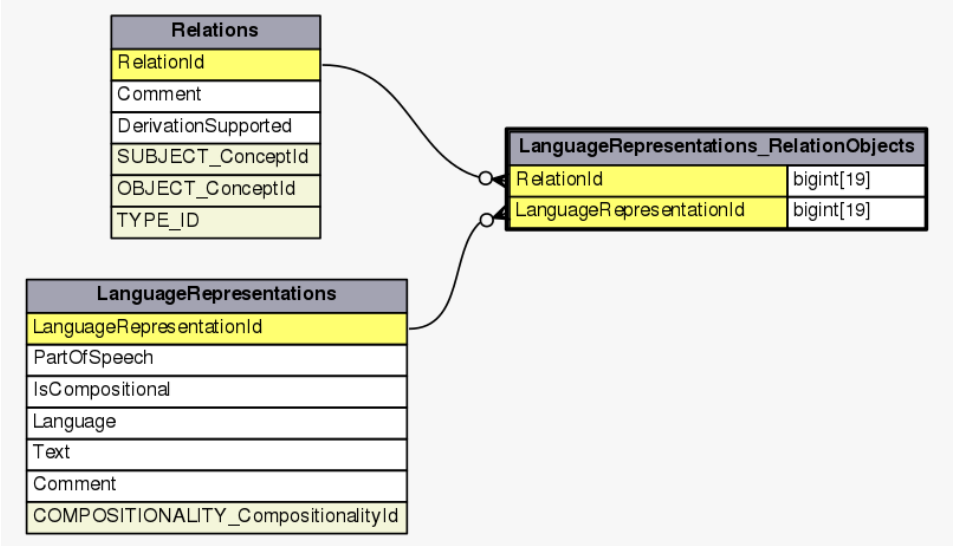


Table: LanguageRepresentations_RelationSubjects

An intermediary table. It breaks the many-to-many relationship between LanguageRepresentations and Relations table with respect to the relation's subject, thus adhering to the 3rd normal form.

Columns

Name	Type	Description
RelationId	bigint	Composite primary key combining two foreign keys to Relations and LanguageRepresentations tables.
LanguageRepresentationId	bigint	

Indexes

Name	Type	Columns	Description
LnggRprsentationRelationSubjectLnggRpresentationId	Non-unique	LanguageRepresentationId	
PRIMARY	Unique	RelationId, LanguageRepresentationId	

Relationships

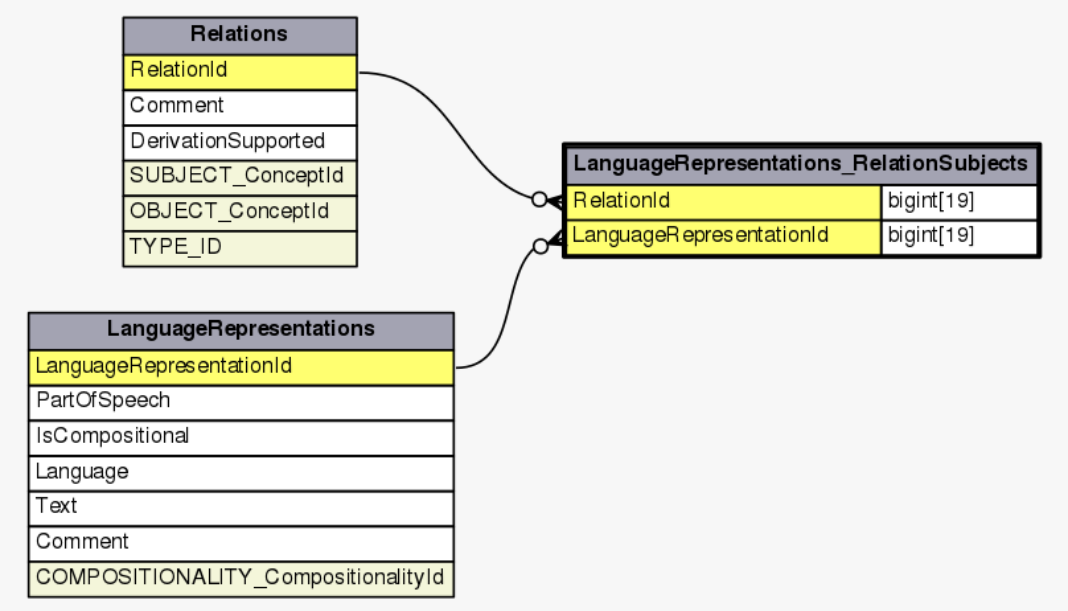


Table: LanguageRepresentations_RelationChains

An intermediary table. It breaks the many-to-many relationship between LanguageRepresentations and RelationChains table, thus adhering to the 3rd normal form.

Columns

Name	Type	Description
LanguageRepresentationId	bigint	Composite primary key combining two foreign keys to LanguageRepresentations and RelationChains tables.
RelationChainId	bigint	

Indexes

Name	Type	Columns	Description
LnguageRepresentationsRelationChainsRlationChainId	Non-unique	RelationChainId	
PRIMARY	Unique	LanguageRepresentationId, RelationChainId	

Relationships

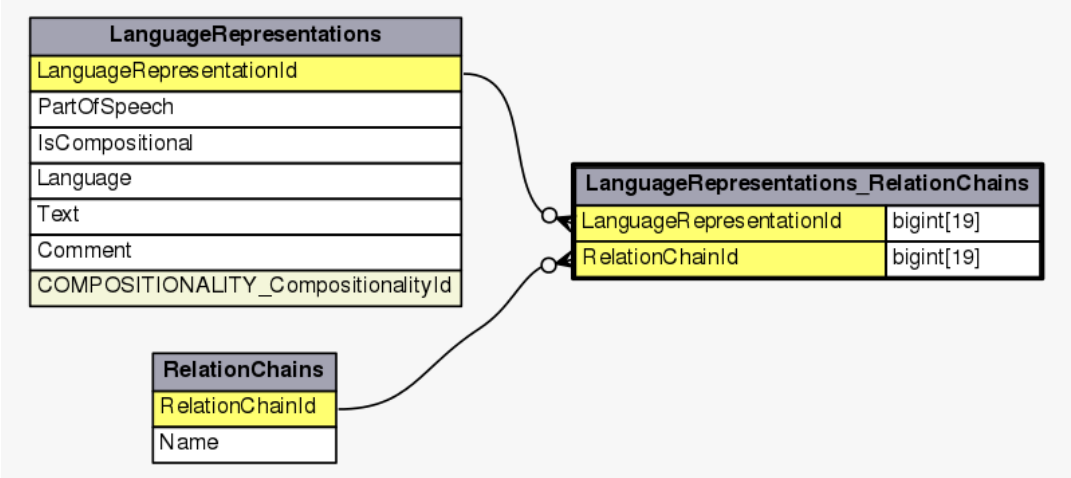


Table: MotoricRepresentations_RelationObjects

An intermediary table. It breaks the many-to-many relationship between MotoricRepresentations and Relations table with respect to the relation's object, thus adhering to the 3rd normal form.

Columns

Name	Type	Description
RelationId	bigint	Composite primary key combining two foreign keys to Relations and MotoricRepresentations tables.
MotoricRepresentationId	bigint	

Indexes

Name	Type	Columns	Description
MtrcRpresentationRelationObjectMtrcRpresentationId	Non-unique	MotoricRepresentationId	
PRIMARY	Unique	RelationId, MotoricRepresentationId	

Relationships

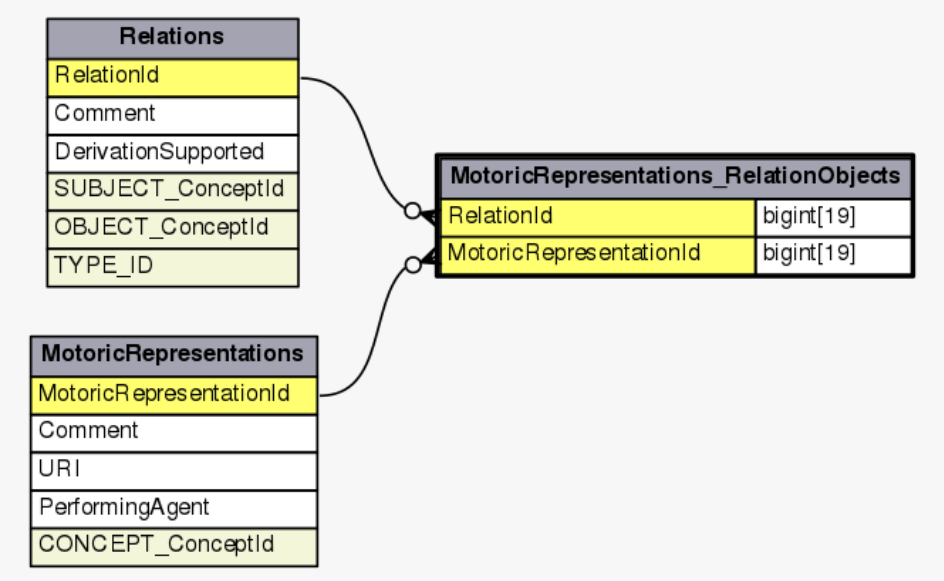


Table: MotoricRepresentations_RelationSubjects

An intermediary table. It breaks the many-to-many relationship between MotoricRepresentations and Relations table with respect to the relation's subject, thus adhering to the 3rd normal form.

Columns

Name	Type	Description
RelationId	bigint	Composite primary key combining two foreign keys to Relations and MotoricRepresentations tables.
MotoricRepresentationId	bigint	

Indexes

Name	Type	Columns	Description
MtrcRprsentationRelationSubjectMtrcRpresentationId	Non-unique	MotoricRepresentationId	
PRIMARY	Unique	RelationId, MotoricRepresentationId	

Relationships

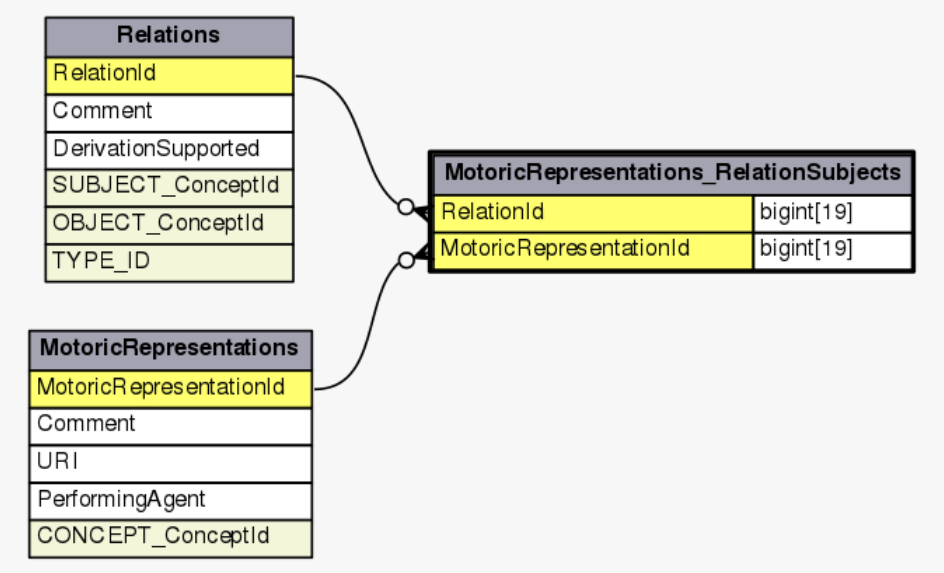


Table: VisualRepresentations_RelationObjects

An intermediary table. It breaks the many-to-many relationship between VisualRepresentations and Relations table with respect to the relation's subject, thus adhering to the 3rd normal form.

Columns

Name	Type	Description
RelationId	bigint	Composite primary key combining two foreign keys to Relations and VisualRepresentations tables.
VisualRepresentationId	bigint	

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	RelationId, VisualRepresentationId	
VslRepresentationRelationObjectVslRepresentationId	Non-unique	VisualRepresentationId	

Relationships

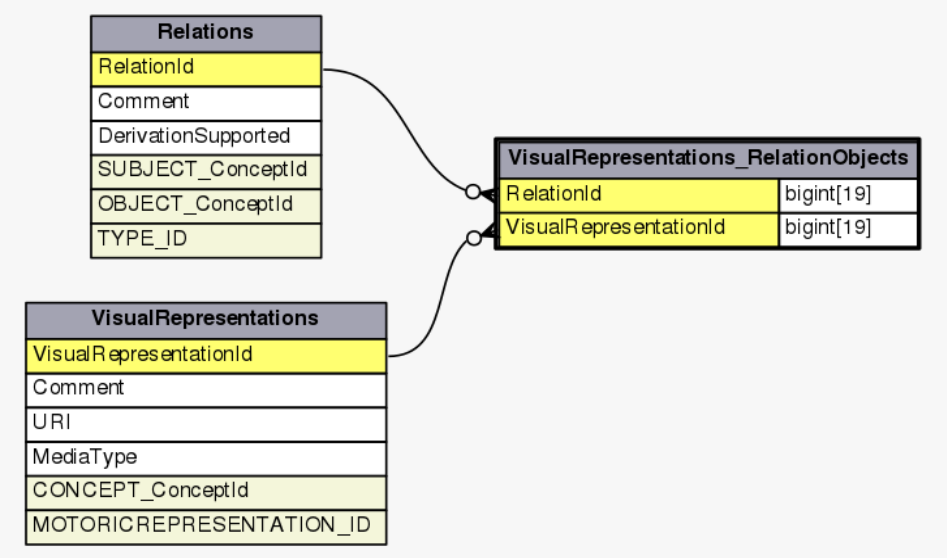


Table: VisualRepresentations_RelationSubjects

An intermediary table. It breaks the many-to-many relationship between LanguageRepresentations and Relations table with respect to the relation's subject, thus adhering to the 3rd normal form.

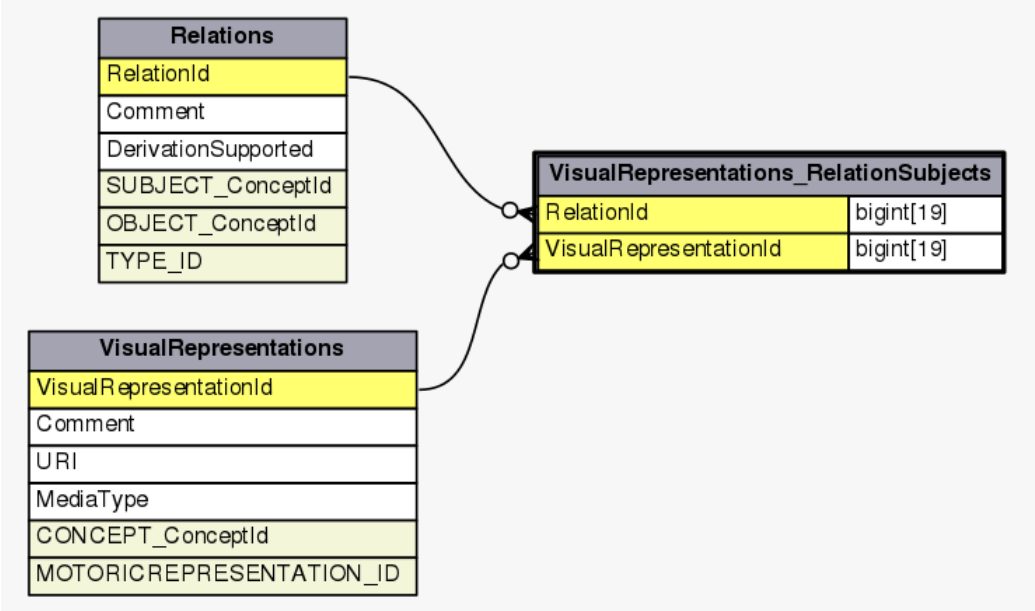
Columns

Name	Type	Description
RelationId	bigint	Composite primary key combining two foreign keys to Relations and VisualRepresentations tables.
VisualRepresentationId	bigint	

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	RelationId, VisualRepresentationId	
VslRpresentationRelationSubjectVslRepresentationId	Non-unique	VisualRepresentationId	

Relationships



Restrictions

The PRAXICON database contains two kinds of restrictions. The first ones are implied by the E-R schema. The second ones cannot be incorporated in the relationships between tables and are imposed externally to cater for special needs of the theoretical background of PRAXICON.

Restrictions have been implemented in the JPA context.

Implied (Foreign keys)

Imposed