CSRI TECHNICAL REPORTS

The PRAXICON database

Documentation

Dimitris Mavroeidis

Katerina Pastra

Athens, 2015

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.

To be cited as:

The PRAXICON database, version 1.0, CSRI Technical Reports, 2015.

This document is available:

On GitHub as part of the PRAXICON database software[1].

On the CSRI website[2].

[1] https://github.com/CSRI/PraxiconDB

[2] http://www.csri.gr/downloads/docs/Praxicon_database_documentation

Table of Contents

Abstract 4
Introduction 4
The PRAXICON 4
The PRAXICON database 5
Implementation 7
Tables 8
Relationships 9
Table: Concepts 11
Columns 11
Indexes 12
Relationships 13
Table: LanguageRepresentations 14
Columns 14
Indexes 14
Relationships 15
Table: Concepts_LanguageRepresentations 16
Columns 16
Indexes 16
Relationships 17
Table: VisualRepresentations 18
Columns 18
Indexes 18
Relationships 19
Table: MotoricRepresentations 20
Columns 20
Indexes 20
Relationships 21
Table: Relations 22
Columns 24
Indexes 24
Relationships 25
Table: RelationTypes 26
Columns 30
Indexes 30
Relationships 31
Table: RelationSets 32
Columns 32
Indexes 32
Relationships 33
Table: RelationSets_Relations 34
Columns 34
Indexes 34
Relationships 35
Table: LanguageRepresentations_RelationSets 36
Columns 36
Indexes 36
Relationships 37

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 spacio-temporal context and encodes abstract knowledge about the world, or –from the linguistic perspective– provides meaning. Thus far, robots have been equipped with 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 that 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 lexical, motoric and visual representations. A lexical representation is a lexical entry that humans use to describe the corresponding concept. A visual representation is an image (entities) or a video (movements) that depicts a visual instance of the concept. A motoric representation only exists for “movement” concepts and can be a computational representation (e.g. a Gait Energy Image - GEI) of a movement (e.g. running gait). Figure 1 shows lexical and visual representations of the concept “table-tennis_racquet”.

Two Concepts can form a Relation. Relations can be grouped and seen as an autonomous structure, the Relation Set. A Relation Set is a collection of Relations and can be either ordered or unordered. A Relation Set can have its own set of representations (much like a Concept). In this capacity, it can also be part of a Relation. Figure 2 shows an example of how these relationships can be realised.

****

TODO: Add some more explanatory figures of the Praxicon structures

****

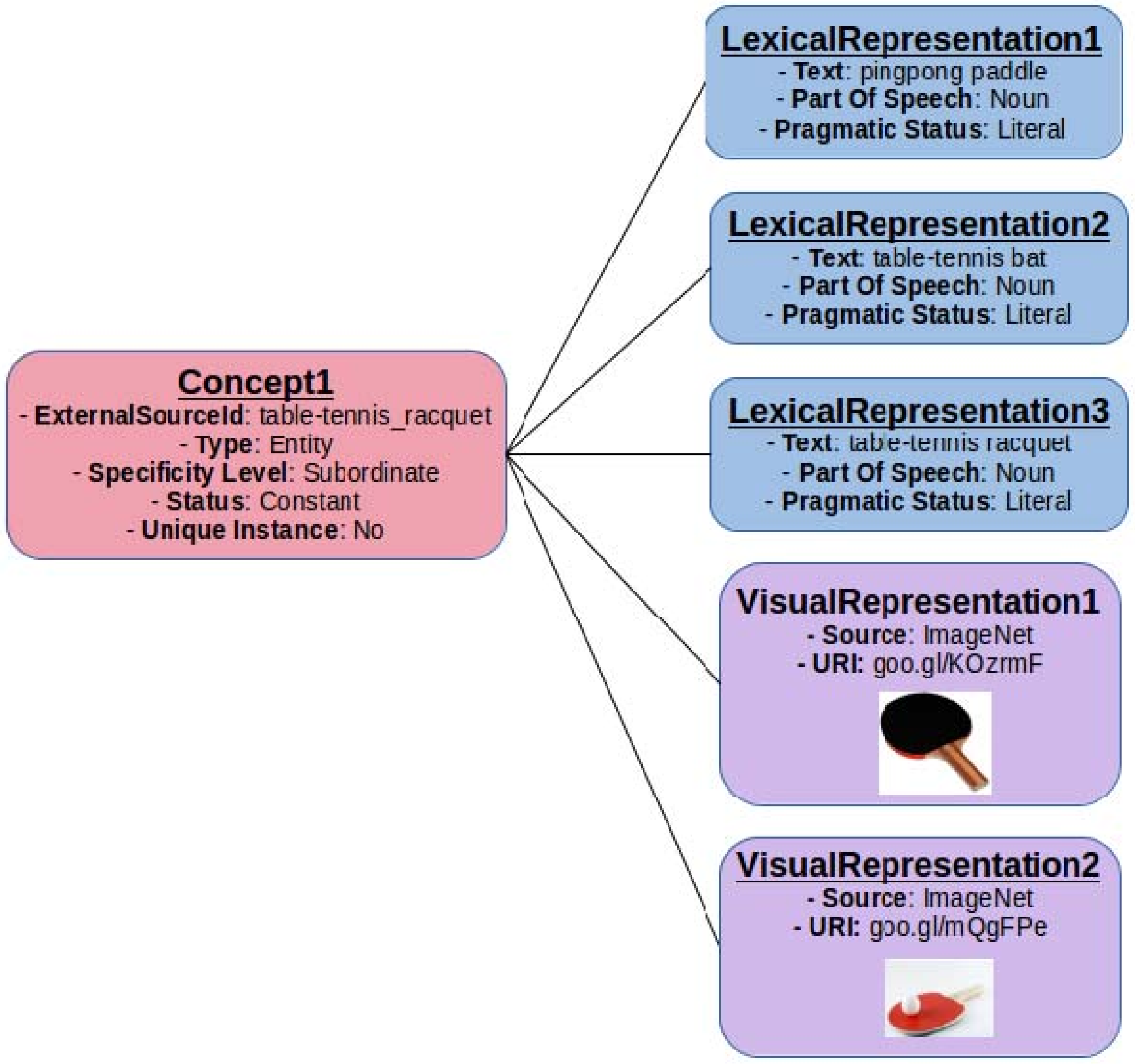


Figure 1 – Lexical and Visual representations of the "table-tennis_racquet" Concept.

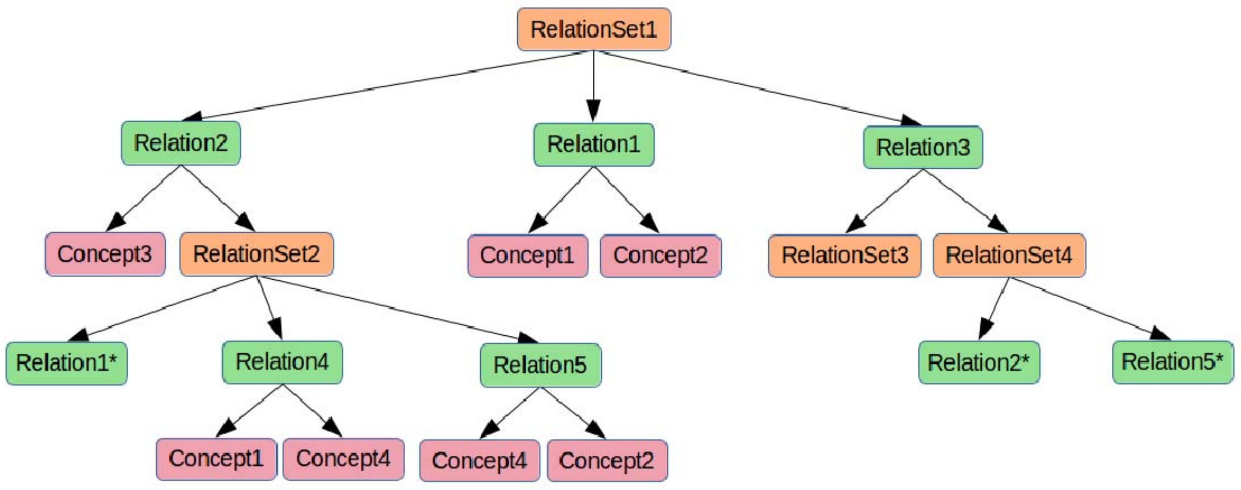


Figure 2 – A schematic representation of how concepts, relations and relation sets can be organized in the Praxicon Database.

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 presented 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. Intermediate tables created to depict Many-to-Many relationships are not presented in this document.

Implementation

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

Tables

Name
Concepts
LanguageRepresentations
Concepts_LanguageRepresentations
VisualRepresentations
MotoricRepresentations
Relations
RelationTypes
RelationSets
RelationSets_Relations
LanguageRepresentations_RelationSets

Relationships

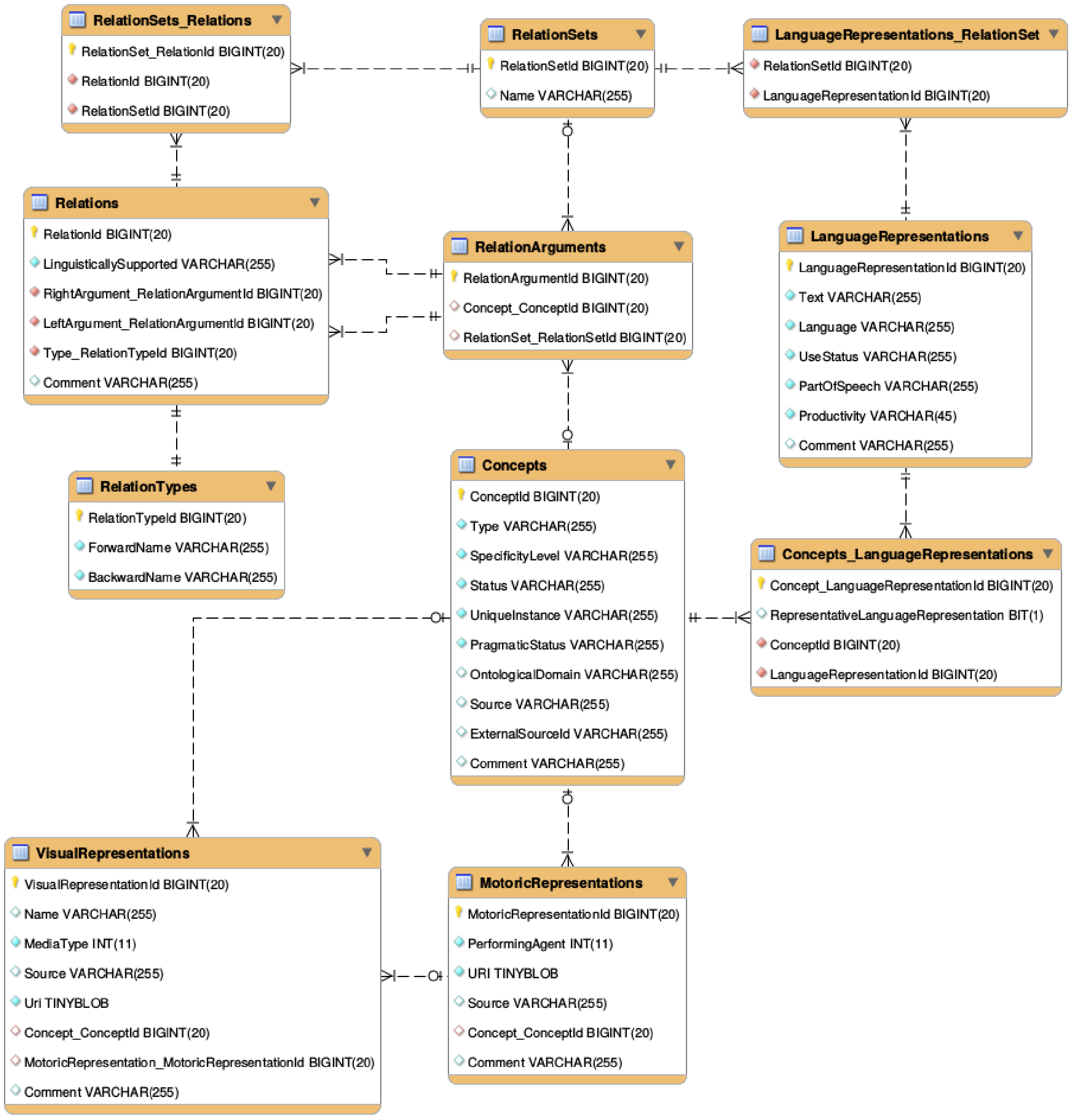

RelationSets_Relations
RelationSet_RelationId BIGINT(20)
RelationId BIGINT(20)
RelationSetId BIGINT(20)
RelationSets
RelationSetId BIGINT(20)
Name VARCHAR(255)
LanguageRepresentations_RelationSet
RelationSetId BIGINT(20)
LanguageRepresentationId BIGINT(20)
Relations
RelationId BIGINT(20)
LinguisticallySupported VARCHAR(255)
RightArgument_RelationArgumentId BIGINT(20)
LeftArgument_RelationArgumentId BIGINT(20)
Type_RelationTypeId BIGINT(20)
Comment VARCHAR(255)
RelationArguments
RelationArgumentId BIGINT(20)
Concept_ConceptId BIGINT(20)
RelationSet_RelationSetId BIGINT(20)
LanguageRepresentations
LanguageRepresentationId BIGINT(20)
Text VARCHAR(255)
Language VARCHAR(255)
UseStatus VARCHAR(255)
PartOfSpeech VARCHAR(255)
Productivity VARCHAR(45)
Comment VARCHAR(255)
RelationTypes
RelationTypeId BIGINT(20)
ForwardName VARCHAR(255)
BackwardName VARCHAR(255)
Concepts
ConceptId BIGINT(20)
Type VARCHAR(255)
SpecificityLevel VARCHAR(255)
Status VARCHAR(255)
UniqueInstance VARCHAR(255)
PragmaticStatus VARCHAR(255)
OntologicalDomain VARCHAR(255)
Source VARCHAR(255)
ExternalSourceId VARCHAR(255)
Comment VARCHAR(255)
Concepts_LanguageRepresentations
Concept_LanguageRepresentationId BIGINT(20)
RepresentativeLanguageRepresentation BIT(1)
ConceptId BIGINT(20)
LanguageRepresentationId BIGINT(20)
VisualRepresentations
VisualRepresentationId BIGINT(20)
Name VARCHAR(255)
MediaType INT(11)
Source VARCHAR(255)
Uri TINYBLOB
Concept_ConceptId BIGINT(20)
MotoricRepresentation_MotoricRepresentationId BIGINT(20)
Comment VARCHAR(255)
MotoricRepresentations
MotoricRepresentationId BIGINT(20)
PerformingAgent INT(11)
URI TINYBLOB
Source VARCHAR(255)
Concept_ConceptId BIGINT(20)
Comment VARCHAR(255)

Figure 3 – The full Entity-Relationship diagram of the Praxicon Database.

Icon	Explanation
🔑	Primary Key
◆	Required Field
◇	Non-required Field
◆	Required Foreign Key
◇	Non-Required Foreign Key
+○— —|<	Non-mandatory OneToMany Relation
‖— —|<	Mandatory OneToMany Relation
‖— —‖	Mandatory OneToOne Relation

Table 1 – An explanatory table of the various database schema symbols.

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, e.g. red, hard).

Columns

Name	Type	Description
🔑ConceptId	bigint	Primary key (automatically generated).
Type	varchar(255)	We recognize four types of concepts, i.e. movement, entity, feature. **Permitted values**: *ENTITY, FEATURE, MOVEMENT, 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*.
Status	varchar(255)	A Concept can be *constant or variable*. 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. Variable concepts also include those that define a pattern. Example 1: "*cut_something_with_a_tool'#movement*" - a variable and its relations define what we can use to cut something (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). Example 2: 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. **Permitted values**: *CONSTANT, VARIABLE*.
PragmaticStatus	varchar(255)	A concept can be either abstract (e.g. morale, paranoia) or concrete (e.g. knife, rabbit). **Permitted values**: *ABSTRACT, CONCRETE*.
UniqueInstance	varchar(255)	Defines whether the concept is unique in the database. **Permitted values**: *YES, NO, UNKNOWN*.

Name	Type	Description
OntologicalDomain	varchar(255)	Defines the ontological domain the concept belongs to.
Source	varchar(255)	Defines the source of the concept (if it has been added manually, from WordNet, or another way).
ExternalSourceId	varchar(255)	If the concept was acquired from an external source (e.g. Wordnet), this field provides the identification number or text of the item in that source.
Comment	varchar(255)	Short description of the concept in plain text; it can be used to store extra information.

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	ConceptId	

Relationships

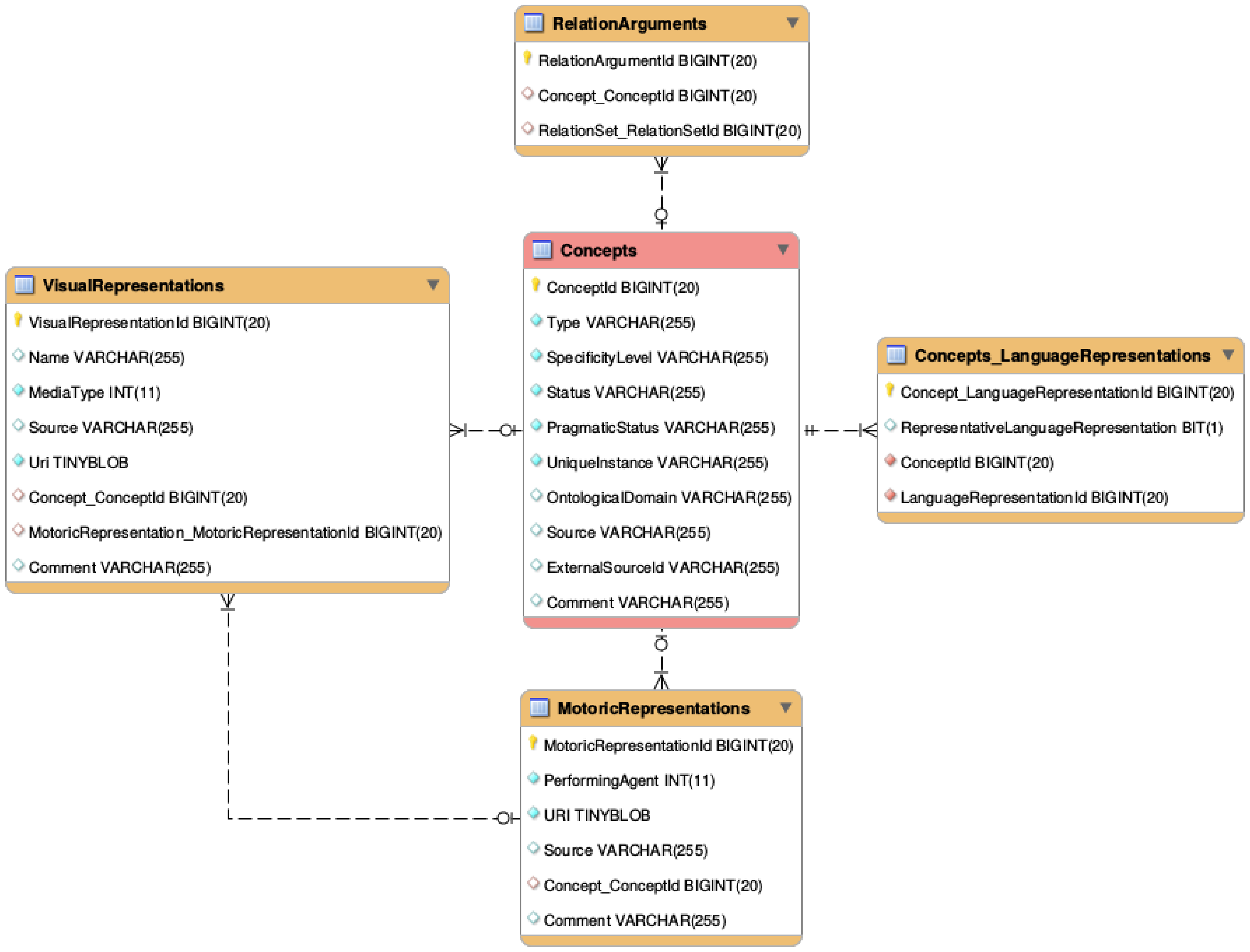

RelationArguments
RelationArgumentId BIGINT(20)
Concept_ConceptId BIGINT(20)
RelationSet_RelationSetId BIGINT(20)
Concepts
ConceptId BIGINT(20)
Type VARCHAR(255)
SpecificityLevel VARCHAR(255)
Status VARCHAR(255)
PragmaticStatus VARCHAR(255)
UniqueInstance VARCHAR(255)
OntologicalDomain VARCHAR(255)
Source VARCHAR(255)
ExternalSourceId VARCHAR(255)
Comment VARCHAR(255)
VisualRepresentations
VisualRepresentationId BIGINT(20)
Name VARCHAR(255)
MediaType INT(11)
Source VARCHAR(255)
Uri TINYBLOB
Concept_ConceptId BIGINT(20)
MotoricRepresentation_MotoricRepresentationId BIGINT(20)
Comment VARCHAR(255)
Concepts_LanguageRepresentations
Concept_LanguageRepresentationId BIGINT(20)
RepresentativeLanguageRepresentation BIT(1)
ConceptId BIGINT(20)
LanguageRepresentationId BIGINT(20)
MotoricRepresentations
MotoricRepresentationId BIGINT(20)
PerformingAgent INT(11)
URI TINYBLOB
Source VARCHAR(255)
Concept_ConceptId BIGINT(20)
Comment VARCHAR(255)

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.

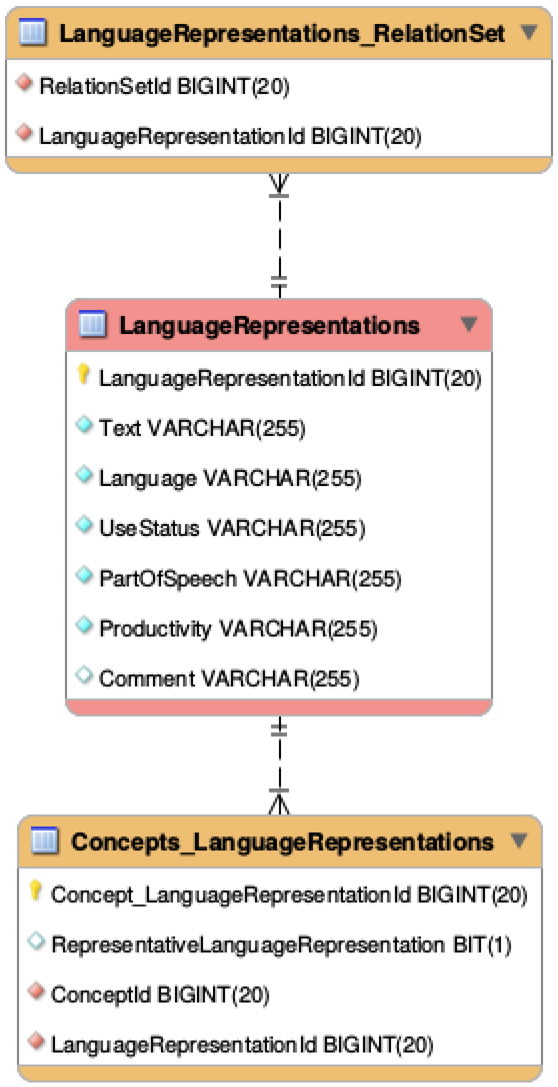
Columns

Name	Type	Description
LanguageRepresentationId	bigint	Primary key (automatically generated).
Text	varchar(255)	The name of the language representation. Can be a word, a word-phrase or a whole phrase.
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*.
UseStatus	varchar(255)	Defines whether the language representation is used in a figurative or a literal manner. **Permitted values**: *FIGURATIVE, LITERAL, UNKNOWN.*
PartOfSpeech	varchar(255)	Defines the part of speech for the language representation of a concept. **Permitted values**: *ADJECTIVE, ADVERB, NOUN, PARTICIPLE, PROPER_NOUN, VERB*.
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
PRIMARY	Unique	LanguageRepresentationId	

Relationships



LanguageRepresentations_RelationSet
RelationSetId BIGINT(20)
LanguageRepresentationId BIGINT(20)
LanguageRepresentations
LanguageRepresentationId BIGINT(20)
Text VARCHAR(255)
Language VARCHAR(255)
UseStatus VARCHAR(255)
PartOfSpeech VARCHAR(255)
Productivity VARCHAR(255)
Comment VARCHAR(255)
Concepts_LanguageRepresentations
Concept_LanguageRepresentationId BIGINT(20)
RepresentativeLanguageRepresentation BIT(1)
ConceptId BIGINT(20)
LanguageRepresentationId BIGINT(20)

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. It additionally contains the *RepresentativeLanguageRepresentation* field which denotes whether the Language Representation is representative of the corresponding concept.

Columns

Name	Type	Description
🔑Concept_LanguageRepresentationId	bigint	Primary key (automatically generated).
RepresentativeLanguageRepresentation	boolean	Whether the language representation is representative of the corresponding concept.
🔑ConceptId	bigint	Foreign key to the Concepts table.
🔑LanguageRepresentationId	bigint	Foreign key to the LanguageRepresenations.

Indexes

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

Relationships

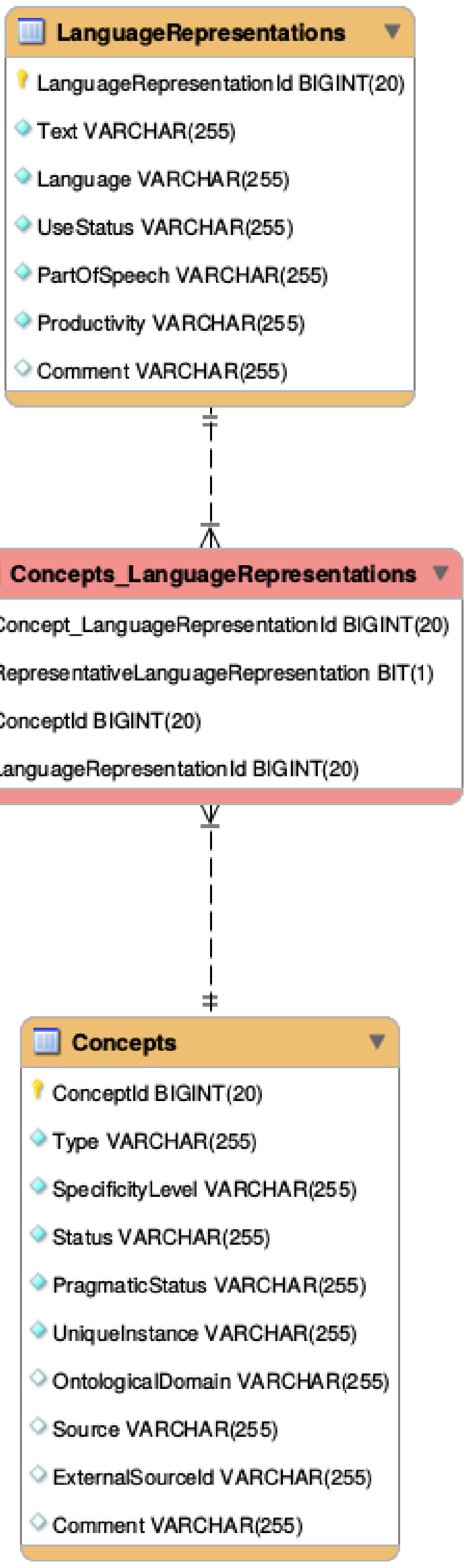

LanguageRepresentations
LanguageRepresentationId BIGINT(20)
Text VARCHAR(255)
Language VARCHAR(255)
UseStatus VARCHAR(255)
PartOfSpeech VARCHAR(255)
Productivity VARCHAR(255)
Comment VARCHAR(255)
Concepts_LanguageRepresentations
Concept_LanguageRepresentationId BIGINT(20)
RepresentativeLanguageRepresentation BIT(1)
ConceptId BIGINT(20)
LanguageRepresentationId BIGINT(20)
Concepts
ConceptId BIGINT(20)
Type VARCHAR(255)
SpecificityLevel VARCHAR(255)
Status VARCHAR(255)
PragmaticStatus VARCHAR(255)
UniqueInstance VARCHAR(255)
OntologicalDomain VARCHAR(255)
Source VARCHAR(255)
ExternalSourceId VARCHAR(255)
Comment VARCHAR(255)

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

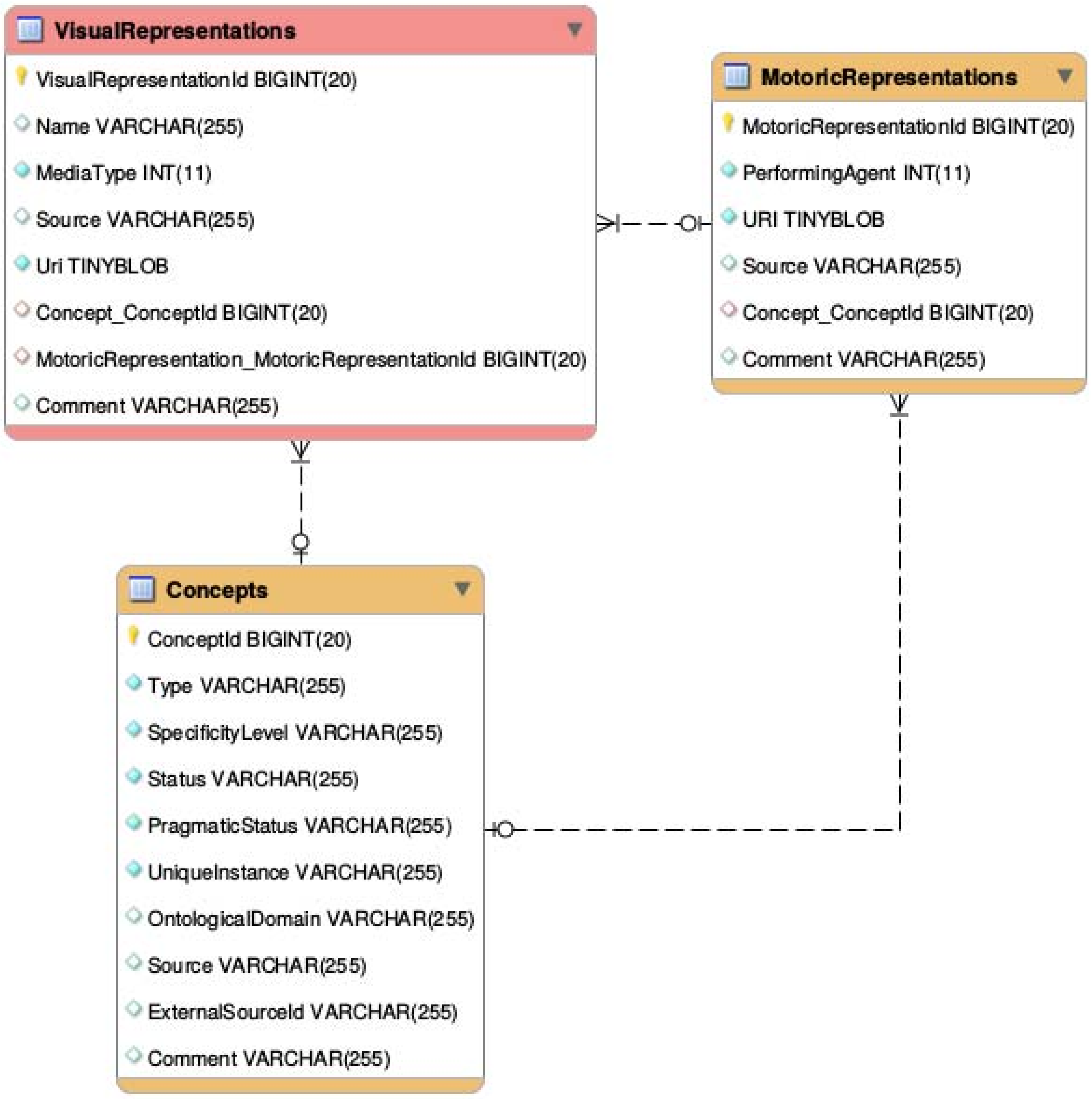

VisualRepresentations
VisualRepresentationId BIGINT(20)
Name VARCHAR(255)
MediaType INT(11)
Source VARCHAR(255)
Uri TINYBLOB
Concept_ConceptId BIGINT(20)
MotoricRepresentation_MotoricRepresentationId BIGINT(20)
Comment VARCHAR(255)
MotoricRepresentations
MotoricRepresentationId BIGINT(20)
PerformingAgent INT(11)
URI TINYBLOB
Source VARCHAR(255)
Concept_ConceptId BIGINT(20)
Comment VARCHAR(255)
Concepts
ConceptId BIGINT(20)
Type VARCHAR(255)
SpecificityLevel VARCHAR(255)
Status VARCHAR(255)
PragmaticStatus VARCHAR(255)
UniqueInstance VARCHAR(255)
OntologicalDomain VARCHAR(255)
Source VARCHAR(255)
ExternalSourceId VARCHAR(255)
Comment VARCHAR(255)

Table: MotoricRepresentations

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

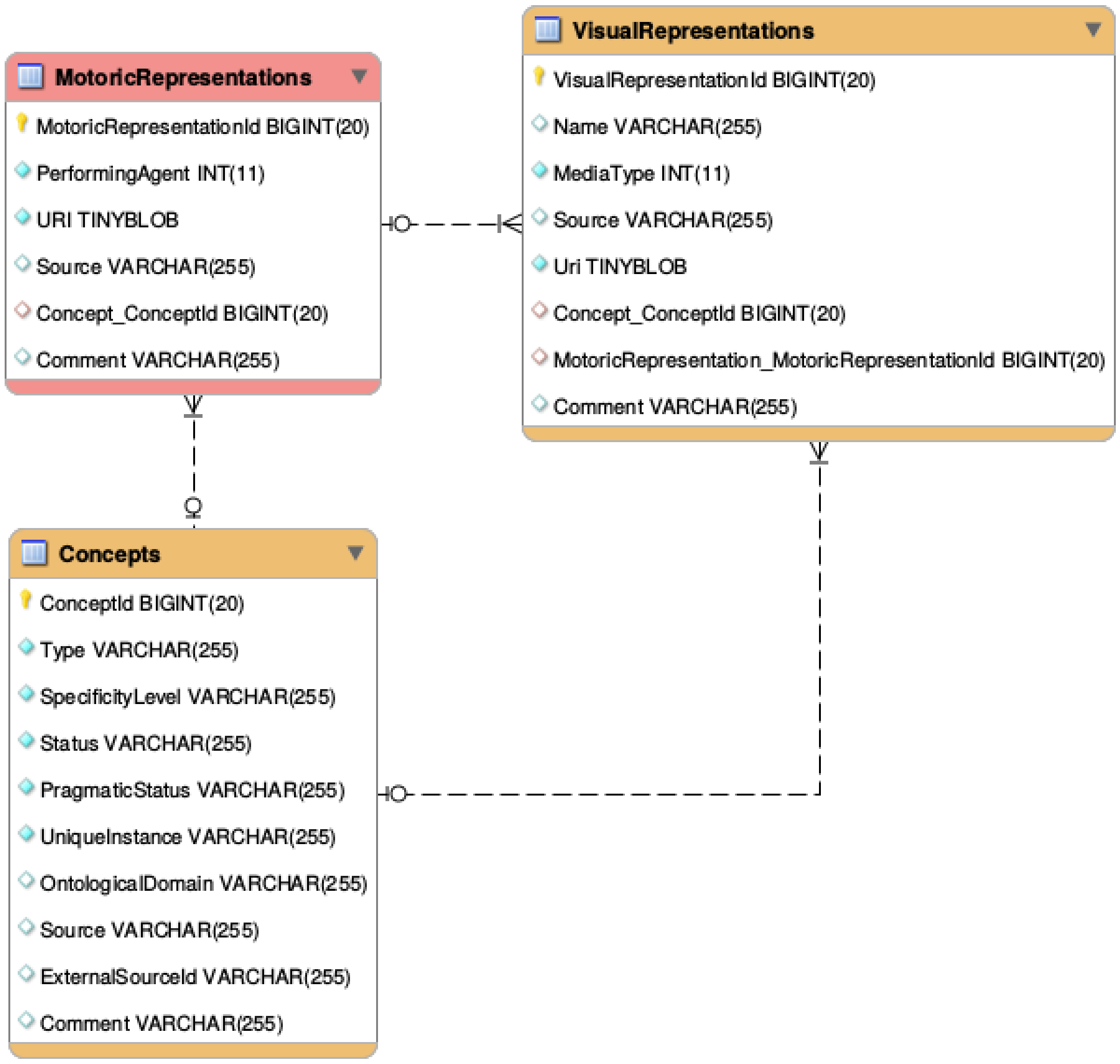
Columns

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



MotoricRepresentations
MotoricRepresentationId BIGINT(20)
PerformingAgent INT(11)
URI TINYBLOB
Source VARCHAR(255)
Concept_ConceptId BIGINT(20)
Comment VARCHAR(255)
VisualRepresentations
VisualRepresentationId BIGINT(20)
Name VARCHAR(255)
MediaType INT(11)
Source VARCHAR(255)
Uri TINYBLOB
Concept_ConceptId BIGINT(20)
MotoricRepresentation_MotoricRepresentationId BIGINT(20)
Comment VARCHAR(255)
Concepts
ConceptId BIGINT(20)
Type VARCHAR(255)
SpecificityLevel VARCHAR(255)
Status VARCHAR(255)
PragmaticStatus VARCHAR(255)
UniqueInstance VARCHAR(255)
OntologicalDomain VARCHAR(255)
Source VARCHAR(255)
ExternalSourceId VARCHAR(255)
Comment VARCHAR(255)

Table: RelationArguments

The relation argument was engineered as a way of determining the type of a right or left argument of a relation. It can be either a “relation set” or a “concept”.

Columns

Name	Type	Description
🔑RelationArgumentId	bigint	Primary key (automatically generated).
ConceptId	bigint	Foreign key to the Concepts table.
RelationSetId	bigint	Foreign key to the RelationSets table.

Relationships

Table: Relations

The relation is at the core of the PRAXICON. A “relation” connects two relation arguments concepts together semantically. It takes one concept as the “left argument” and another one as the “right argument”. 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.
LinguisticallySupported	varchar(255)	Whether the relation supports derivation. There is no need to know which language representations are derivationally related. **Permitted values**: *YES, NO, UNKNOWN*.
Inferred	varchar(255)	Whether a Relation is the result of an inference. If the relation has been inferred by a reasoning mechanism, then its “Inferred” value would be “YES”. **Permitted values**: *YES, NO, UNKNOWN*.
LEFTARGUMENT_RelationArgumentId	bigint	Foreign key to RelationArguments table which denotes which concept is the left argument in this relation.
RIGHTARGUMENT_RelationArgumentId	bigint	Foreign key to RelationArguments table which denotes which concept is the right argument in this relation.
TYPE_ID	bigint	Foreign key to the RelationTypes table.

Indexes

Name	Type	Columns	Description
FK_Relations_RIGHTARGUMENT_ConceptId	Non-unique	RIGHTARGUMENT_ConceptId	
FK_Relations_LEFTARGUMENTSUBJECT_ConceptId	Non-unique	LEFTARGUMENT_ConceptId	
FK_Relations_TYPE_ID	Non-unique	TYPE_ID	
PRIMARY	Unique	RelationId	

Relationships

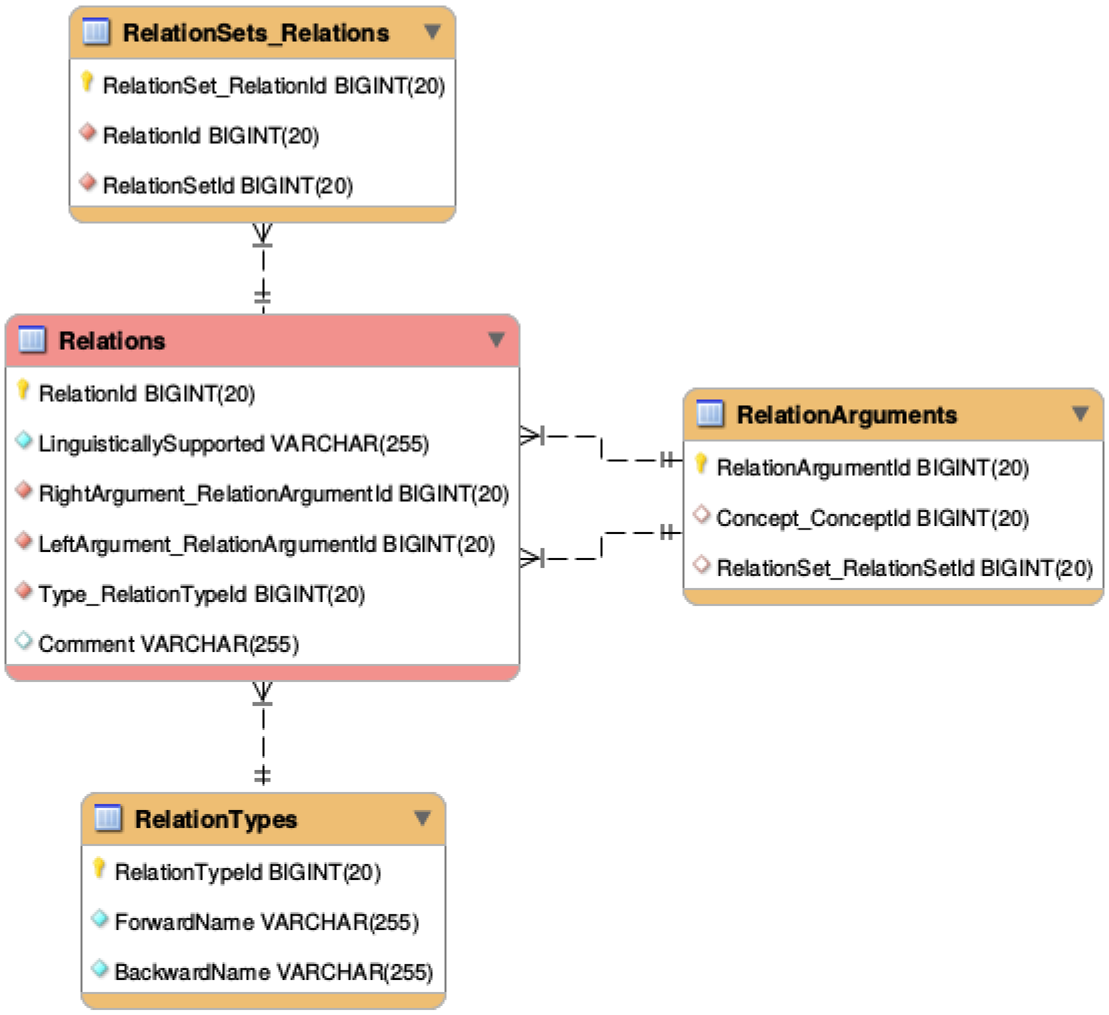

RelationSets_Relations
RelationSet_RelationId BIGINT(20)
RelationId BIGINT(20)
RelationSetId BIGINT(20)
Relations
RelationId BIGINT(20)
LinguisticallySupported VARCHAR(255)
RightArgument_RelationArgumentId BIGINT(20)
LeftArgument_RelationArgumentId BIGINT(20)
Type_RelationTypeId BIGINT(20)
Comment VARCHAR(255)
RelationArguments
RelationArgumentId BIGINT(20)
Concept_ConceptId BIGINT(20)
RelationSet_RelationSetId BIGINT(20)
RelationTypes
RelationTypeId BIGINT(20)
ForwardName VARCHAR(255)
BackwardName VARCHAR(255)

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_AGENT	AGENT_ACTION	*'fly~with~dummy~tool~fly', 'ACTION_AGENT', 'seagul'*
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_ABSTRACT_ENTITY	ABSTRACT_ENTITY_ASPECT	*'rhinorrhea', 'ASPECT_ABSTRACT_ENTITY', 'common_cold'* This relation denotes an aspect of a concept. For instance, rhinorrhea is an aspect (symptom) of the common cold.
ASPECT_ABSTRACT_FEATURE	ABSTRACT_FEATURE_ASPECT	*'butterfly_wing', 'ASPECT_ABSTRACT_FEATURE', 'beautiful'*
COMPARED_WITH	COMPARED_WITH	*'egg', 'COMPARED_WITH', 'car'* This relation can only be used inside a Relation Set 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.
EVENT_STEP	STEP_EVENT	*'deal', 'STEP_EVENT', 'card_game'*
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_DIRECTION	DIRECTION_OF	*'lift_with_hand_the_lithic_blade#movement', 'HAS_DIRECTION', 'upwards'*
HAS_FORCE	FORCE_OF	*'huricane', 'HAS_FORCE', 'strong'*
HAS_FREQUENCY	FREQUENCY_OF	*'rub_with_finger_the_lithic_borer', 'HAS_FREQUENCY', 'continuous'*
HAS_HEIGHT	HEIGHT_OF	*'mountain', 'HAS_HEIGHT', 'high'*
HAS_HUE	HUE_OF	*'basket', 'HAS_HUE', 'orange'*
HAS_MOTOR_PROGRAM	MOTOR_PROGRAM_OF	*'university', 'HAS_INSTANCE', 'harvard_university'* This example can be interpreted like this: *Harvard*

		University is an instance of the University class.
HAS_INSTANCE	INSTANCE_OF	‘*bowling*’, HAS_INSTANCE, ‘*roll*’
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 above, a chocolate is made of cacao beans.
HAS_MATERIAL_RESISTANCE	RESISTANCE_MATERIAL_OF	*'soft_object', 'HAS_MATERIAL_RESISTANCE', 'soft'* An “is-made-of” relation. In the example above, 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_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_0F	*‘treat#with#dummy_tool_treat#the#dummy_object_treat’, 'HAS_PARTIAL_INSTANCE', 'treat#with#dummy_tool_treat_high_blood_pressure#the#high_blood_pressure'*
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_TEMPERATURE	TEMPERATURE_OF	*‘water’, ‘HAS_TEMPERATURE’, ‘warm’*
HAS_TEXTURE	TEXTURE_OF	*‘road’, ‘HAS_TEXTURE’, ‘rough’*
HAS_TIME	TIME_OF	*'time-out', 'TIME_OF', 'athletic_game'*
HAS_VALUE	VALUE_OF	
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 Relation Set.
LESS	no inversion allowed,	See above

	use NONE in backward naming	
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 right argument for the left and vice-versa.

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	RelationTypeId	

Relationships

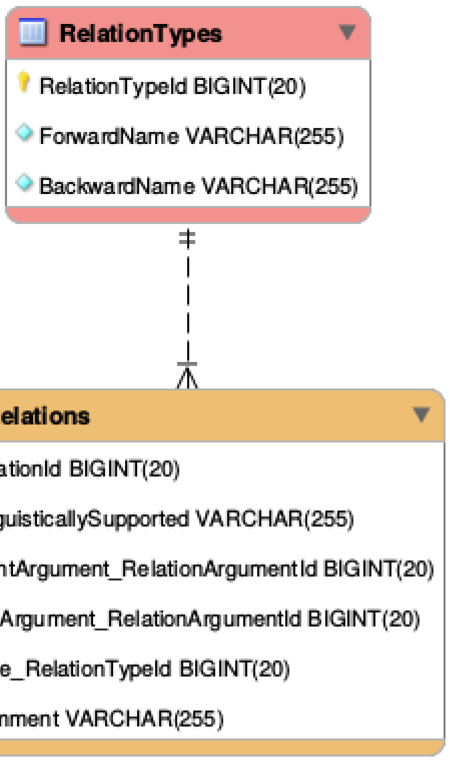

RelationTypes
RelationTypeId BIGINT(20)
ForwardName VARCHAR(255)
BackwardName VARCHAR(255)
Relations
RelationId BIGINT(20)
LinguisticallySupported VARCHAR(255)
RightArgument_RelationArgumentId BIGINT(20)
LeftArgument_RelationArgumentId BIGINT(20)
Type_RelationTypeId BIGINT(20)
Comment VARCHAR(255)

Table: RelationSets

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 Sets”. 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 Set can stand alone as a relation independently of the whole graph.

A Relation Set can consist of *at least* one relation, so there is a many-to-many relationship with the Relations table. Examples of Relation Sets containing only one relation are passive participles, e.g. snowy. “Snowy” can be analysed as a Relation Set with the Relation “dummy#HAS_CONDITION#snowy”.

In addition, a Relation Set can take its own Language Representation. Thus, there is a many-to-many relationship with the Language Representations table as well.

Columns

Name	**Type**	**Description**
RelationSetId	Bigint	Primary key (automatically generated).
Name	varchar(255)	A human-friendly name for the relation set.

Indexes

Name	**Type**	**Columns**	**Description**
PRIMARY	Unique	RelationSetId	

Relationships

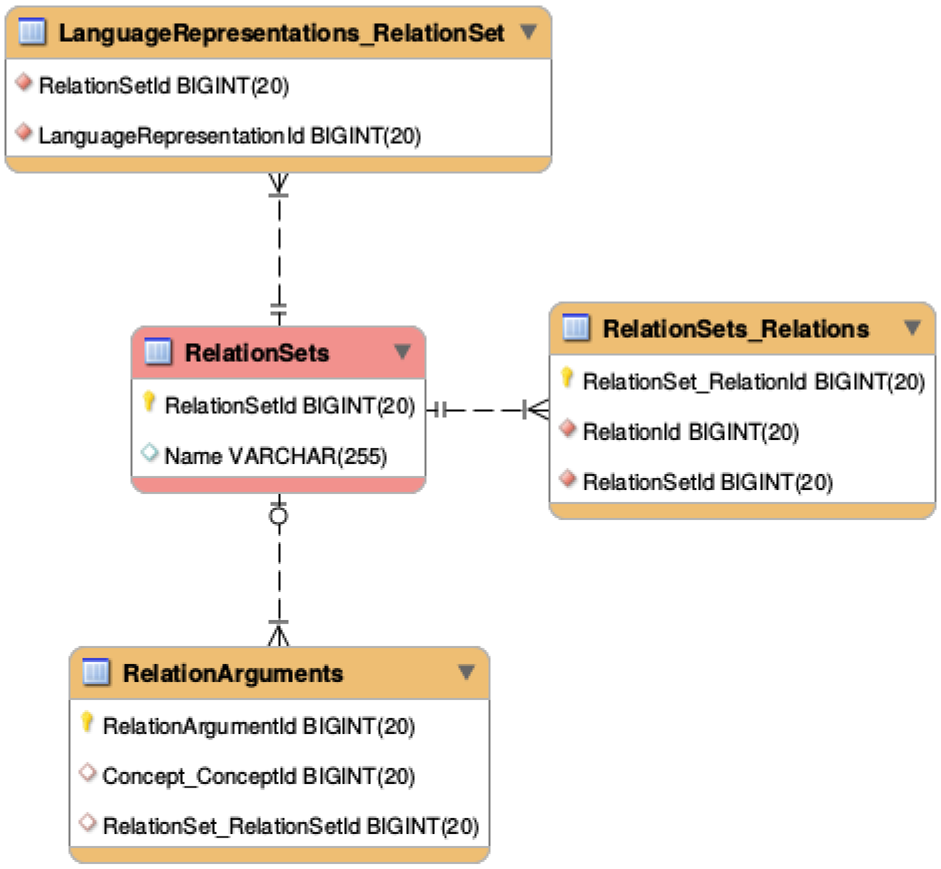

LanguageRepresentations_RelationSet
RelationSetId BIGINT(20)
LanguageRepresentationId BIGINT(20)
RelationSets
RelationSetId BIGINT(20)
Name VARCHAR(255)
RelationSets_Relations
RelationSet_RelationId BIGINT(20)
RelationId BIGINT(20)
RelationSetId BIGINT(20)
RelationArguments
RelationArgumentId BIGINT(20)
Concept_ConceptId BIGINT(20)
RelationSet_RelationSetId BIGINT(20)

Table: RelationSets_Relations

This table is an intermediary one that breaks the many-to-many relationship between the RelationSets and Relations tables.

Columns

Name	Type	Description
🔑RelationSet_RelationId	bigint	Primary key (automatically generated).
RelationSetId	bigint	Foreign key to the RelationSets table.
RelationId	bigint	Foreign key to the Relations table.

Indexes

Name	Type	Columns	Description
FK_RelationSets_Relations_RelationSetId	Non-unique	RelationSetId	
FK_RelationSets_Relations_RelationId	Non-unique	RelationId	
PRIMARY	Unique	RelationSet_RelationId	

Relationships

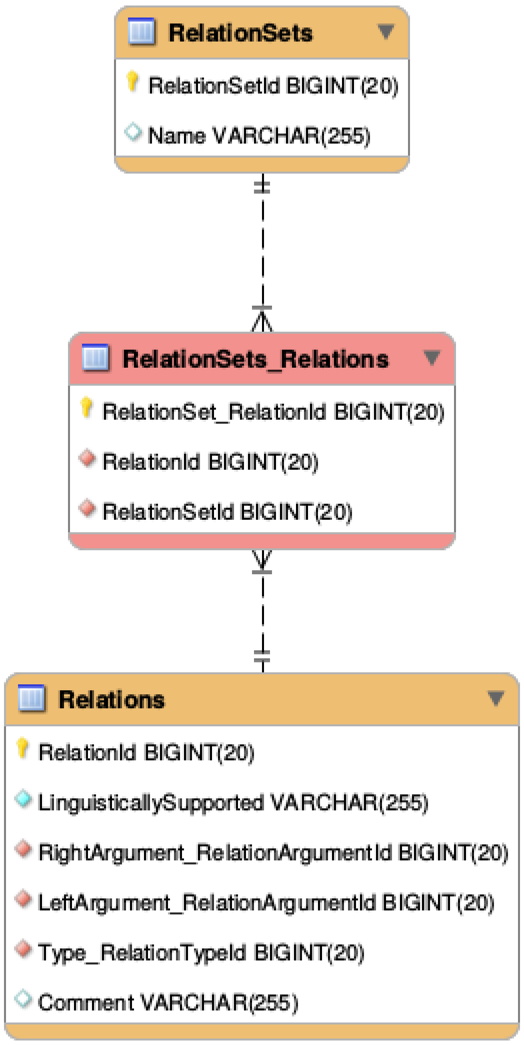

RelationSets
RelationSetId BIGINT(20)
Name VARCHAR(255)
RelationSets_Relations
RelationSet_RelationId BIGINT(20)
RelationId BIGINT(20)
RelationSetId BIGINT(20)
Relations
RelationId BIGINT(20)
LinguisticallySupported VARCHAR(255)
RightArgument_RelationArgumentId BIGINT(20)
LeftArgument_RelationArgumentId BIGINT(20)
Type_RelationTypeId BIGINT(20)
Comment VARCHAR(255)

Table: LanguageRepresentations_RelationSets

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

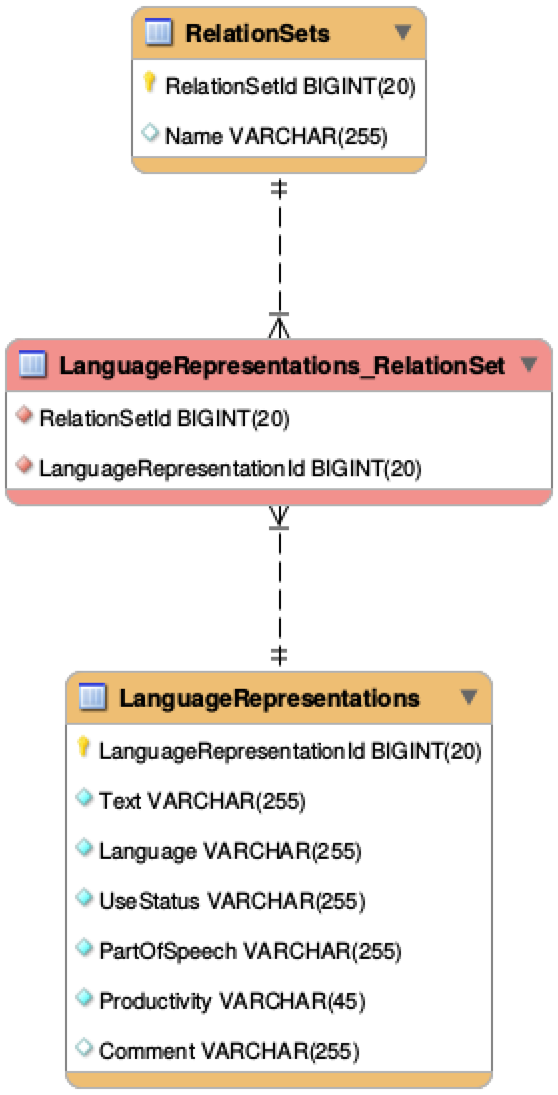
Columns

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

Indexes

Name	Type	Columns	Description
LnggRprsnttnRelationSetsRltnSetId	Non-unique	RelationSetId	
PRIMARY	Unique	LanguageRepresentationId, RelationSetsId	

Relationships


RelationSets
RelationSetId BIGINT(20)
Name VARCHAR(255)
LanguageRepresentations_RelationSet
RelationSetId BIGINT(20)
LanguageRepresentationId BIGINT(20)
LanguageRepresentations
LanguageRepresentationId BIGINT(20)
Text VARCHAR(255)
Language VARCHAR(255)
UseStatus VARCHAR(255)
PartOfSpeech VARCHAR(255)
Productivity VARCHAR(45)
Comment VARCHAR(255)