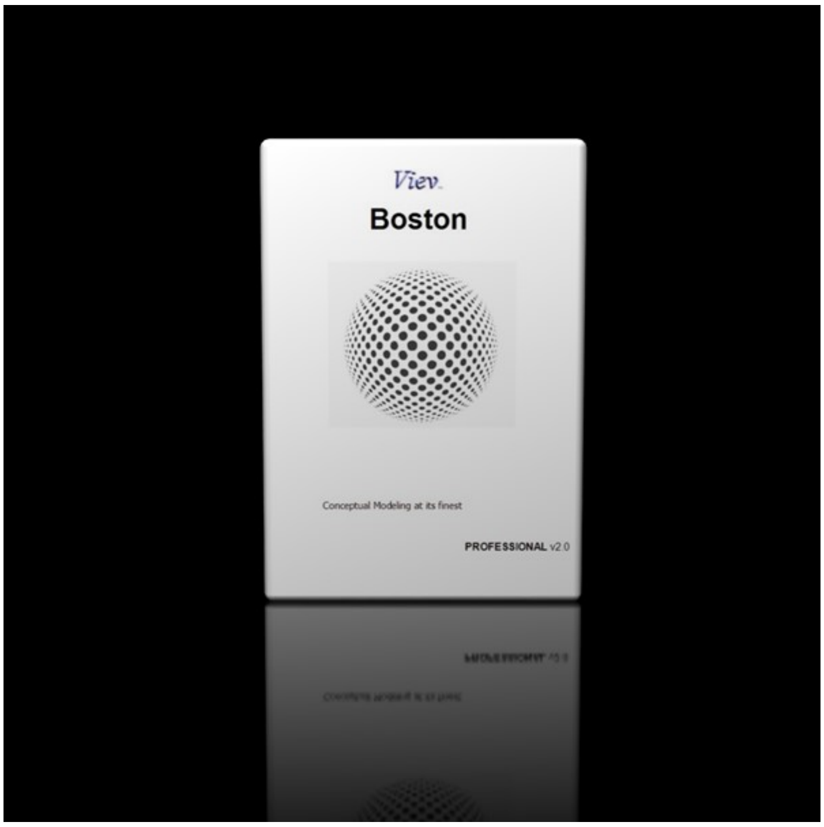
"FactEngine"

Boston Professional

Database Documentation



Author:	FactEngine
Database Version:	1.32
Release Date:	20th March 2022

Table of Contents

1. The Boston Database

The Boston database contains a state of the art MetaModel of Fact-Based Modeling (FBM). The following help topics describe the tables and fields of the Fact-Based Modeling MetaModel within Boston.

NB Tables that are prefixed, 'MetaModel', are at the MetaMetaModel level of a 4-Layer Architecture when considering languages interpreted from the Model (stored within the MetaModel) that are not ORM or FBM Models. Tables that are prefixed 'Model' are at the MetaModel level of a 4-Layer Architecture when considering languages interpreted from the Model (stored within the MetaModel) that are not ORM or FBM Models. When interpreting Object-Role Models from within the database, tables prefixed MetaMetaModel need only be considered MetaModel. Please contact FactEngine for a paper describing the architecture. Allowing for a 4-Layer Architecture supports Languages other than ORM to be stored within the same MetaModel.

The Boston metamodel/Metametamodel currently supports Object-Role Models, Entity Relationship Diagrams, Property Graph Schema and State Transition Diagrams.

⚠ **IMPORTANT** Never delete the Model called, "LanguageEnglish", and with ModelId, "English", from the Boston database backend.. This model is required by the Virtual Analyst functionality in Boston.

⚠ **IMPORTANT** Never delete the Model called, "Core", and with ModelId, "Core", from the database backend.. This model is required by Boston for future extensions including the creation of ER Diagrams within Boston.

Technical Details

The Boston database is a Microsoft Jet (MS Jet) database, and Boston uses the embeddable ADO.Net database engine to access the MS Jet database file. Download the ‘adodb.dll’ direct link library from Microsoft if your computer does not have that library and/or access to a Microsoft Jet database.

1.1. Boston Database Location

To find where the Boston database is stored within your implementation of Boston, select the [Boston]->[Configuration] menu options from within Boston and view the connection string within the [General] tab on the Boston Configuration form. If using Boston Online you will require administrator privileges to perform this task.

1.2. Backing-Up the Boston Database

If your installation of Boston uses the Microsoft Jet database engine you can make regular and ad hoc backups of your Boston database from within Boston.

To backup the Boston database, select the [Boston]->[Database]->[Backup Database] menu option from within Boston. If using Boston Online you will require administrator privileges to perform this task.

1. Boston will make a backup of the Boston database with a filename of the following format:

bostonYYYYMMDDHHNNSS.vdb

YYYY	- is the year of the database backup
MM	- is the month of the database backup
DD	- is the day of the month of the database backup
HH	- is the hour of the day of the database backup
NN	- is the minute of the hour of the day of the database backup
SS	- is the second of the minute of the hour of the day of the database backup

2. Concepts to Understand

The Boston database stores the model details of all the models available within the Boston software. Therefore all metamodel tables contain a reference to the model to which the model elemements belong.

This is because model elements (of any component of a model) can have the same Symbol (Concept) reference between models and the name/id of the model element is used to identify the model element along with the ModelId to which it belongs.

2.1. Model Elements and the ModelDictionary

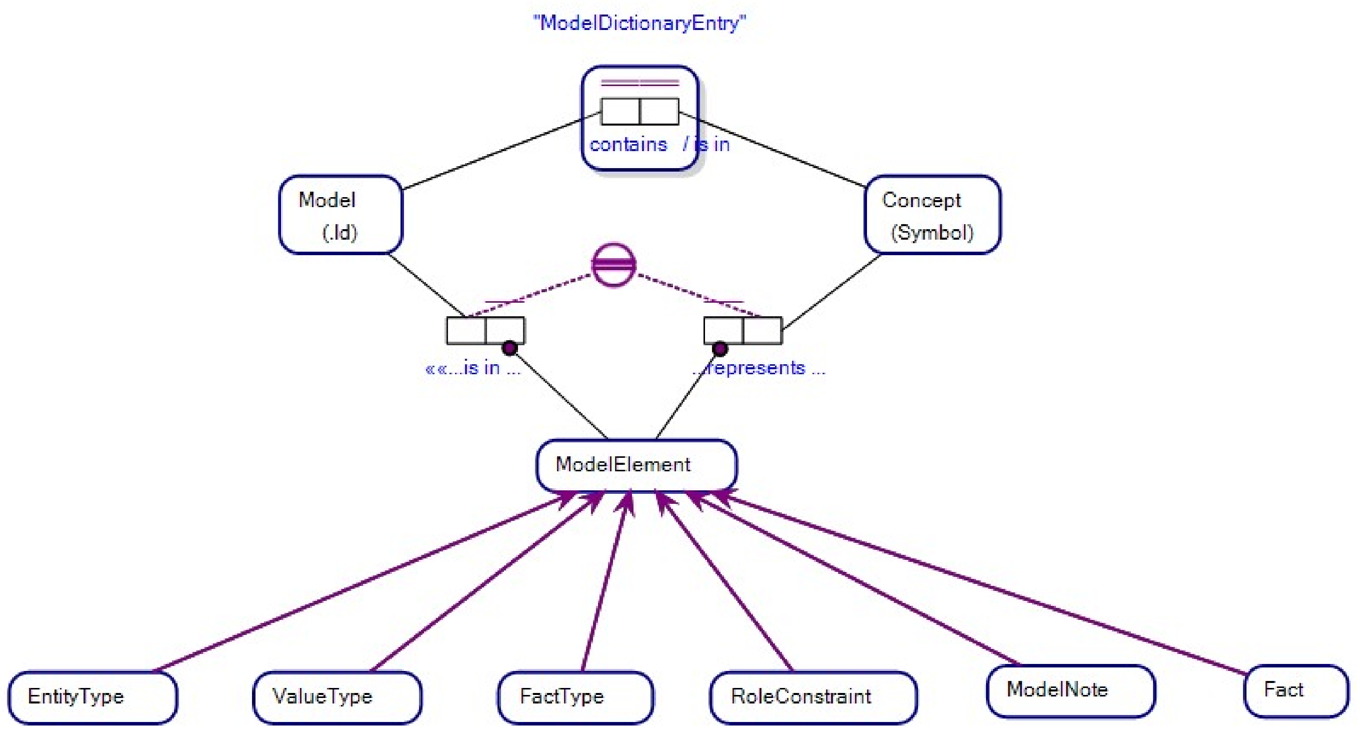
Anything within a model in Boston is considered a model element, but the primary model elements are Entity Types, Value Types, Fact Types, Role Constraints, Model Notes, Facts and Values.

Entity Types, Value Types, Fact Types, Role Constraints, Model Notes, Facts (Sample Populations) and even Values (of Sample Populations) are considered Model Elements in the Boston database and are represented by their Symbol (along with the Model to which they belong).

E.g. The Symbol for the Etity Type, Person, is "Person".

This can be achieved because ostensibly Object-Role Modeling is a predicate logic and the only time a "Person" would be both a Entity Type and a Fact Type in Object-Role Modeling is when there is an objectified Fact Type, "Person", with an Objectifying Entity Type of "Person". A Value Type, Person_Name, however would never be one of the other subtypes of ModelElement. The Symbol for a Fact or Model Note is an abitrary GUID, otherwise the actual name of the Model Element is used.

This makes it very easy to manage and debug a Boston database, as arbitrary 'Ids' for primary model elements is not used.

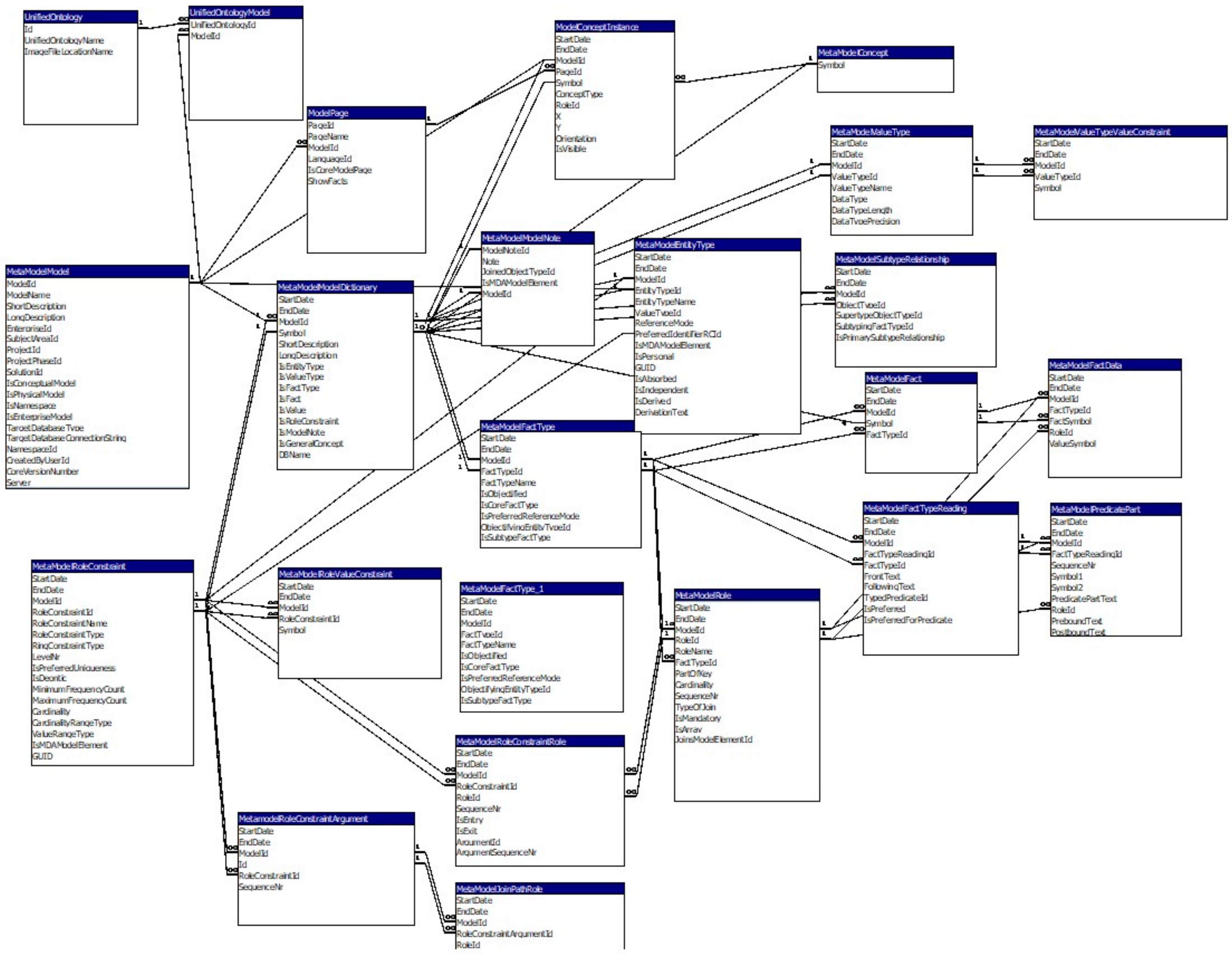


NB In Boston Objectifying Entity Types currently have their own unique Symbol/Name.

NB There is no table called ModelElement. See the MetaModelModelDictionary table.

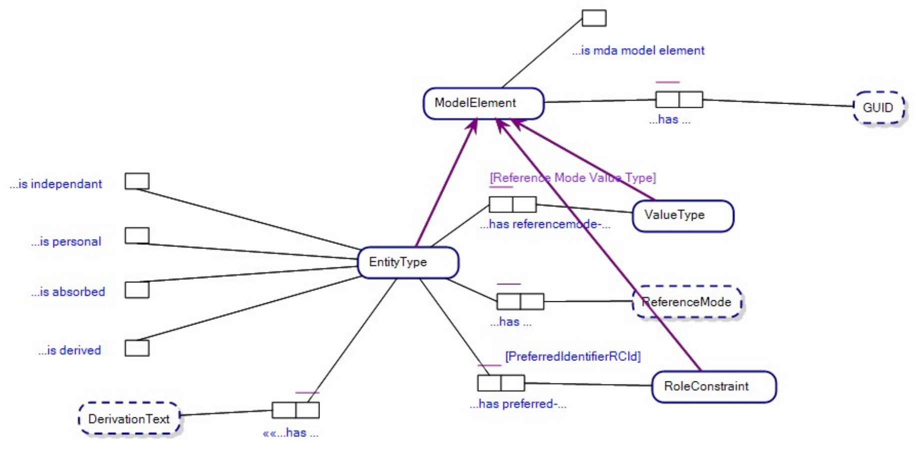
2.2. Schema

The pictorial layout of the Boston database schema is available from FactEngine.



2.3. Boston Model as an Object-Role Model

The Boston database conceptual model is available from FactEngine as a Boston Model.



3. Boston Database Tables

3.1. The MetaModel Database Tables

The Boston MetaModel consists of 19 tables that store Models and MetaModels. The following is a list of the Boston MetaModel tables in the database:

Table Name	Description
MetaModelConcept	Stores the set of Concepts as utilised in Models in Boston.
MetaModelEntityType	Stores the set of Entity Types by Model
MetaModelFact	Stores the set of Facts by Model
MetaModelFactData	Stores the set of Fact Data by Fact, by Model
MetaModelFactType	Stores the set of Fact Types by Model
MetaModelFactTypeReading	Stores the set of Fact Type Readings by Fact Type, by Model
MetaModelJoinPathRole	Stores Roles that are part of a Join Path for an Argument of a Role Constraint.
MetaModelModel	Stores the set of Models in Boston
MetaModelModelDictionary	Stores the set of Concepts utilised in a Model
MetaModelModelNote	Stores the set of Model Notes by Model
MetaModelPredicatePart	Stores the set of Predicate Parts of Fact Type Readings by Model
MetaModelRole	Stores the set of Roles by Fact Type, by Model
MetaModelRoleConstraint	Stores the set of Role Constraints by Model
MetaModelRoleConstraintArgument	Stores details of Role Constraint Arguments for those Role Constraints that have Arguments.
MetaModelRoleConstraintRole	Stores the set of Roles referenced by a Role Constraint, by Model
MetaModelRoleValueConstraint	Stores values for Role Value Constraints.
MetaModelSubtypeRelationship	Stores the set of Supertypes by Model Object/Object Type, by Model
MetaModelValueType	Stores the set of Value Types by Model
MetaModelValueTypeConstraint	Stores the set of Value Constraints for a Value Type, by Model
ModelConceptInstance	Stores the Instance data for Concepts by Page, by Model
ModelLanguage	Stores the set of Languages supported by Boston
ModelPage	Stores the set of Pages by Model

3.1.1. The MetaModelConcept table

The MetaModelConcept table stores the set of Concepts as utilised in Models in Boston.

Field	Description
Symbol	The Symbol that represents the Concept. e.g. A Concept of type Person is represented by the Symbol, 'Person'. The 'Symbol' of a Concept may be a String, Binary Bit Stream, Number, Chinese Character, Japanese Kanji Character etc. In the Boston MetaModel, the word 'runs' (for example) is thought of as a Symbol in its own right as if it were an idiomatic Symbol (which, of course, it conceptually is; only that in English that Concept is spelt using four characters rather than one). i.e. The Boston MetaModel takes a 'nominalistic' approach to conceptual modeling.

Concepts in the ORM Studio database

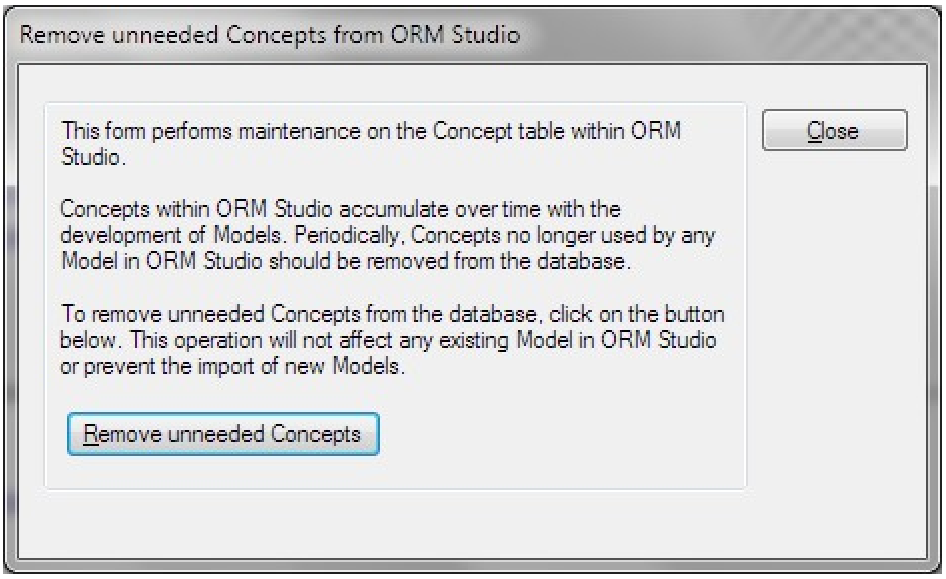
Concepts are stored within the Boston database within a table called, 'MetaModelConcept'.

The MetaModelConcept table has only one field called, 'Symbol', and stores a list of the Concepts/Symbols used in all Models within Boston.

Concepts instances within the Boston database accumulate over time with the development of Models. Periodically, Concepts no longer used by any Model in Boston should be removed from the database.

Table Maintenance - MetaModelConcept

To remove unneeded Concepts from the database, open the [Boston] menu on the main Boston form and select [Database]->[Remove unneeded Concepts].
You will be presented with the Concept maintenance form.



This form performs maintenance on the Concept table within ORM Studio.

Concepts within ORM Studio accumulate over time with the development of Models. Periodically, Concepts no longer used by any Model in ORM Studio should be removed from the database.

To remove unneeded Concepts from the database, click on the button below. This operation will not affect any existing Model in ORM Studio or prevent the import of new Models.

To remove unneeded Concepts from the database click on the button [Remove unneeded Concepts]. This operation will not affect any existing Model in Boston or prevent the import of new Models.

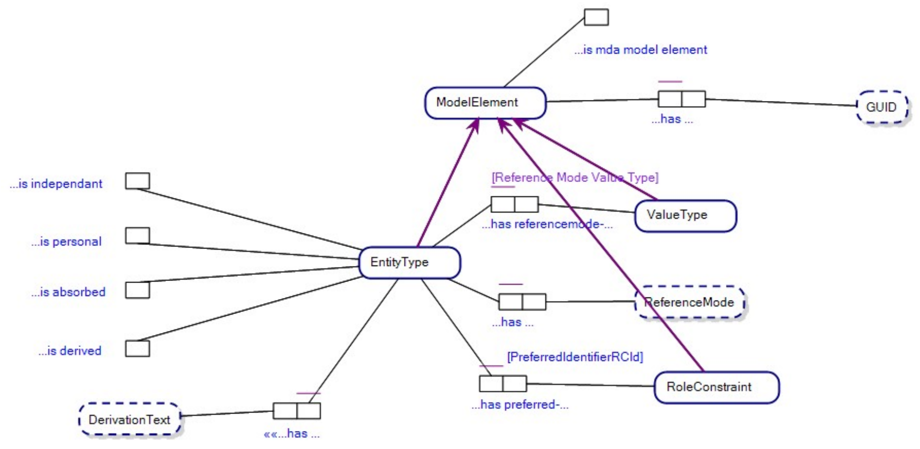
3.1.2. The MetaModelEntityType table

The MetaModelEntityType table stores the set of Entity Types utilised by a Model.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
EntityTypeId	The unique identifier of the Entity Type.
EntityTypeName	The name of the Entity Type. Must be the same as EntityTypeId in Boston.
ValueTypeId	If the Entity Type has a Reference Mode which is not a Compound Reference Scheme, set to the unique identifier (ValueTypeId) of the Value Type that forms part of the Reference Scheme for the Entity Type, else this field is NULL.
ReferenceMode	If the Entity Type has a Reference Mode which is not a Compound Reference Scheme, set the 'Reference Mode' of the Entity Type, else this field is NULL
PreferredIdentifierRCId	Set to the unique identifier (RoleConstraintId) of the Role Constraint which forms part of the Reference Scheme for the Entity Type.
IsMDAModelElement	True if the Entity Type is part of the Core Model in the 4-Layer Architecture used by Boston. MDA stands for 'Model Driven Architecture'.
IsPersonal	True if the Entity Type should result in ORM Verbalises that use "who" rather than "that" for instances of the Entity Type. Currently not used in Boston.
GUID	Unique universal identifier for the Entity Type. Not currently used by Boston. Can be used to store URIs etc in ontologies.
IsAbsorbed	True if the corresponding RDS (Relational Database Structure) Table for the Entity Type is absorbed into the table of the Primary Subtype Relationship (Identification Path) of the Entity Type. Is only relevant if the Entity Type is a subtype of another Model Element.
IsIndependent	True if the Entity Type is independent (in an ORM sense).
IsDerived	True if there is a derivation rule that defines instances of the Entity Type.
DerivationText	If the Entity Type is derived (see IsDerived above), the derivation rule that defines instances of the Entity Type.

See also:

The MetaModelModelDictionary table



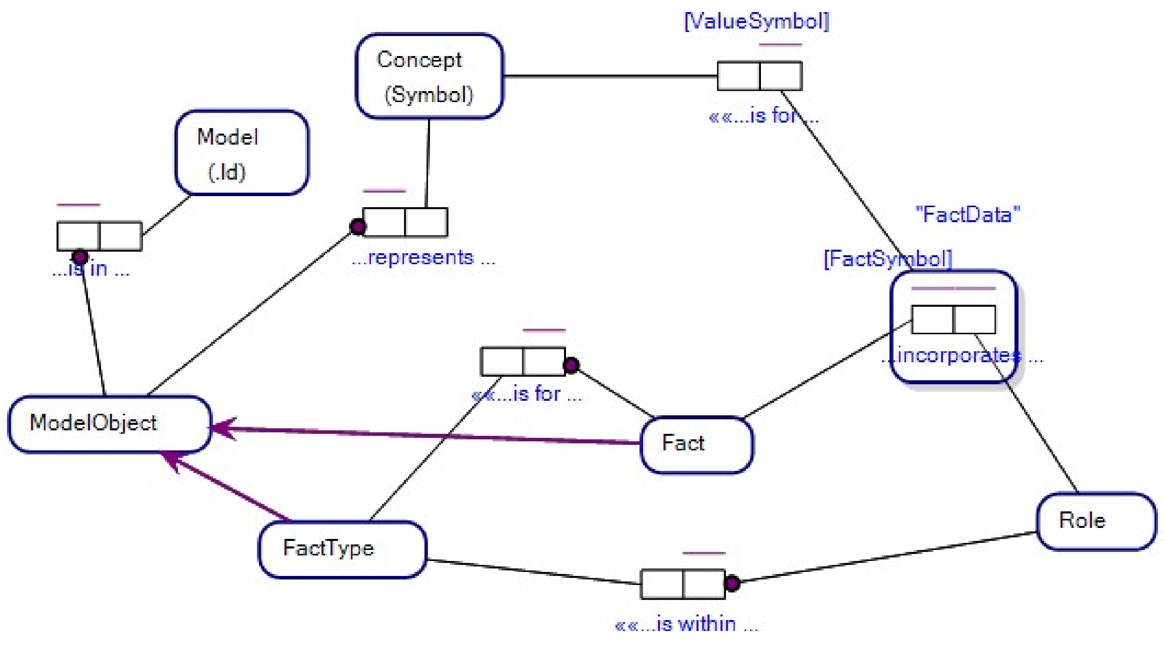
The MetaModelFact table

The MetaModelFact table stores the set of Facts in a Model.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
Symbol	The unique identifier of the Fact. Equivalent to 'RowId' in say an ORACLE database. In the Boston generic Object-Role Modeling MetaModel (gORMmm), a Lemma is the lowest level of identification of a Concept/Symbol, of which a Fact (Tuple) can be a Concept/Symbol in it's own right. I.e. Facts of Fact Types have identity in the Boston model.
FactTypeId	Foreign Key reference to the MetaModelFactType table.

See also:

The MetaModelFactData table
The MetaModelFactType table
The MetaModelModelDictionary table



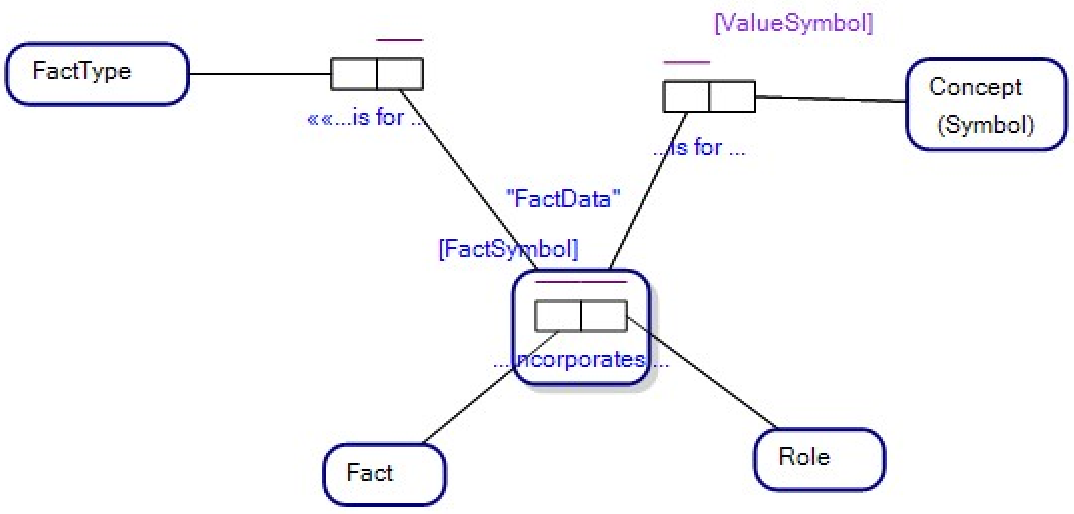
3.1.3. The MetaModelFactData table

The MetaModelFactData table stores the data of a Fact.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
FactTypeId	Foreign Key reference to the MetaModelFactType table. The Fact Type to which the Fact Data belongs.
FactSymbol	Foreign Key reference to the MetaModelFact table. The Fact to which the Fact Data belongs.
RoleId	Foreign Key reference to the MetaModelRole table. The Role within the Fact Type to which the Fact Data belongs.
ValueSymbol	The data value of the Fact Data.

See also:

The MetaModelFactData table
The MetaModelFactType table



3.1.4. The MetaModelFactType table

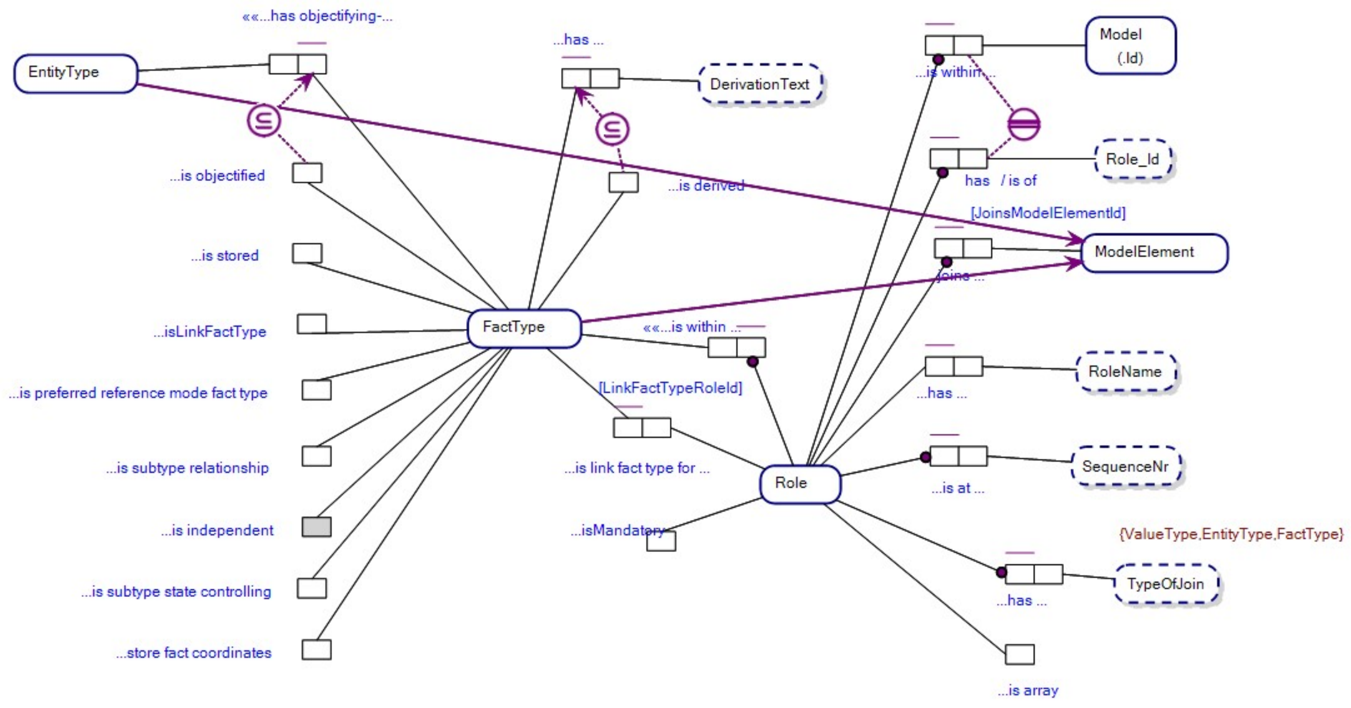
The MetaModelFactType table stores details of each Fact Type in a Model.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
FactTypeId	The unique identifier of the Fact Type.
FactTypeName	The name of the Fact Type. Must be the same as FactTypeId in Boston.
IsObjectified	Set to True if the Fact Type is objectified, else set to False.
IsCoreFactType	Not used in this version of Boston.
IsPreferredReferenceMode	If an Entity Type does not have a Compound Reference Scheme, set to True where the Fact Type is the Fact Type that contains the Role Constraint which defines the Reference Scheme for the Entity Type, else set to False. Used to hide Fact Types within the graphical user interface (GUI), where a Reference Scheme for an Entity Type (that does not have a Compound Reference Scheme) is not displayed to the user of the software.
ObjectifyingEntityTypeId	Set to the unique identifier of the Entity Type that is the Objectifying Entity Type of an Objectified Fact Type, else NULL
IsSubtypeFactType	Set to True if the Fact Type is an Identity Fact Type that defines a Subtype Relationship between Object Types.
IsMDAModelElement	True if the FactType is part of the Core MetaModel stored within the ORM MetaModel as MetaMetaModel, else False. MDA stands for 'Model Driven Architecture'.
IsDerived	True if instances of the Fact Type are defined by a derivation rule, else False.
DerivationText	If the Fact Type is derived, the derivation rule that ranges over instances of the Fact Type.
IsLinkFactType	True if the Fact Type is a Link Fact Type for an Objectified Fact Type, else False.
GUID	Unique universal identifier for the Fact Type. Not currently used by Boston. Can be used to store URIs etc in ontologies. Used predominantly for model sharing with other platforms.
LinkFactTypeRoleId	If the Fact Type is a Link Fact Type, is a foreign key reference to the Role to which the Link Fact Type is related.
IsIndependent.	True if the Fact Type is independent in an ORM sense.
IsSubtypeStateControlling	True if the FactType is a relation between a Supertype and a ValueType such that values of the ValueType determine the state of Subtypes of the Supertype, else False Used when State Transition Diagrams are created for a Value Type that is correlated with the type of Subtypes of a Supertype linked to that Value Type. E.g. Person has Gender, where PersonType is a Value Type with a Value Constraint ranging over 'Male', 'Female'.
StoreFactCoordinates	NB Only used for MDA FactTypes in the CMML. True if coordinates are stored for Facts allocated to a Page. Normally FactData coordinates are stored only. FactData/Values are not always unique on a Page. Facts are always unique on a Page.

See also:

The MetaModelFact table

The MetaModelFactData table
The MetaModelRole table
The MetaModelFactTypeReading table
The MetaModelPredicatePart table



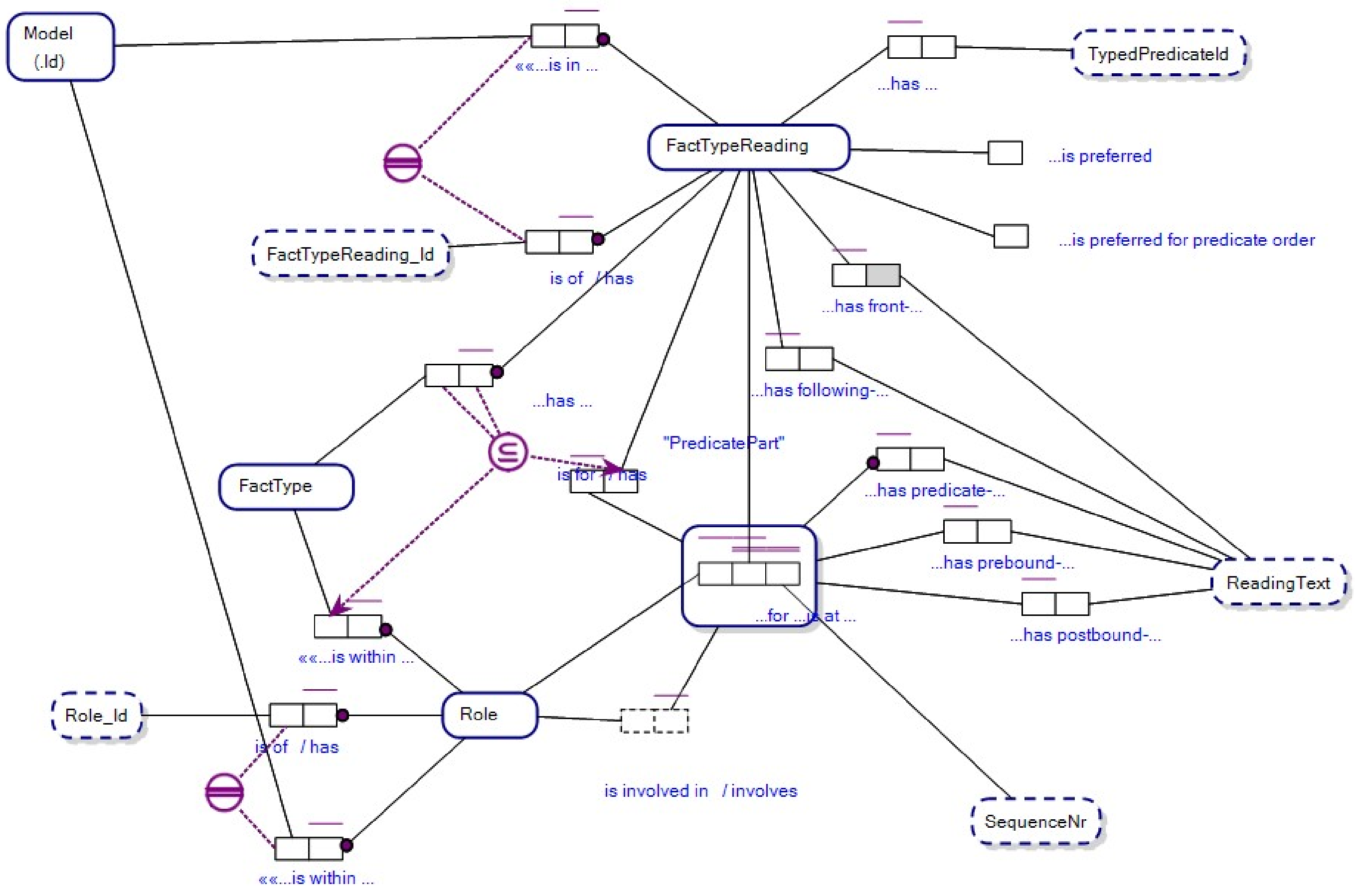
3.1.5. The MetaModelFactTypeReading table

The MetaModelFactTypeReading table stores details of each Fact Type Reading for a Fact Type.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
FactTypeReadingId	The unique identifier of the Fact Type Reading.
FactTypeId	Foreign Key reference to the MetaModelFactType table. The Fact Type to which the Fact Type Reading belongs.
FrontText	Stores the Front Text of a Fact Type Reading. E.g. In the Fact Type Reading "sometimes Person goes to Town", 'sometimes' is Front Text.
FollowingText	Stores the Following Text of a Fact Type Reading. E.g. In the Fact Type Reading "Person goes to Town sometimes", 'sometimes' is Front Text.
TypedPredicateId	The Id of the TypedPredicate to which the FactTypeReading belongs. A TypedPredicate is an ordered set of Roles of a FactType. The order of the Roles is always the same as the FactTypeReading. There may be more than one FactTypeReading with the same TypedPredicateId Not really used in Boston. See the Fact-Based Modelling Working Group's, Fact-Based Modelling MetaModel.
IsPreferred	True if the FactTypeReading is the preferred Fact Type Reading for the associated Fact Type, else False. See the Fact-Based Modelling Working Group's, Fact-Based Modelling MetaModel. Not really used in Boston for anything important.
IsPreferredForPredicate	If there is more than one FactTypeReading for a FactType with the same TypedPredicateId, one of them may be 'Preferred'. See the Fact-Based Modelling Working Group's, Fact-Based Modelling MetaModel. Not really used in Boston for anything important.

See also:

The MetaModelFactType table
The MetaModelPredicatePart table



Fact Type Reading – ORM Model

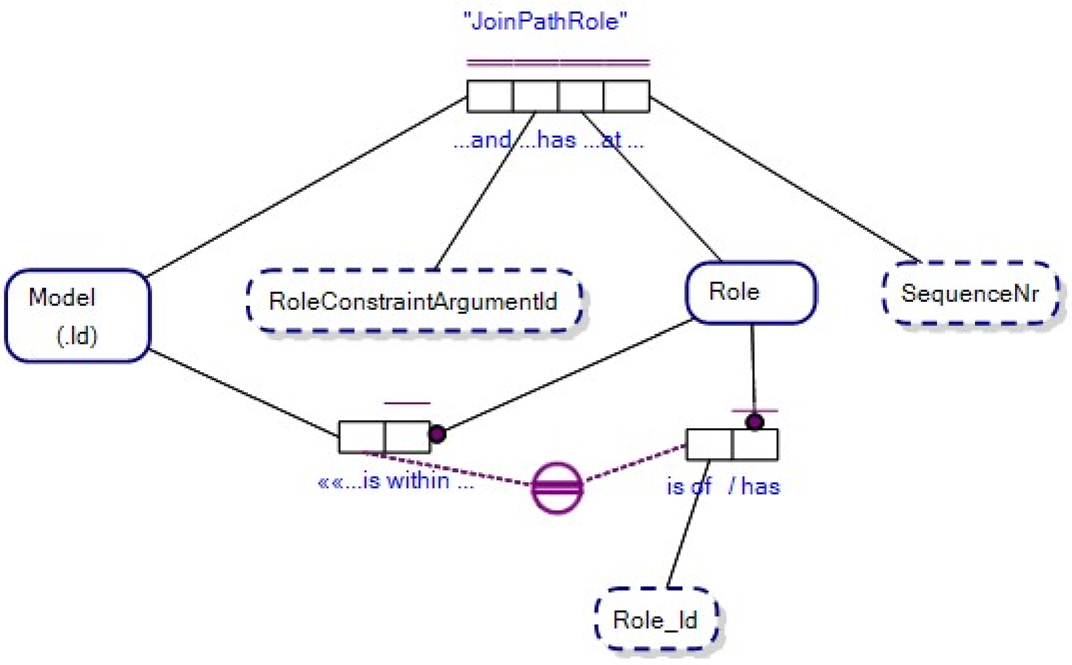
3.1.6. The MetaModelJoinPathRole table

The MetaModelJoinPathRole table stores Roles that are part of a Join Path for an Argument of a Role Constraint.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
RoleConstraintArgumentId	The Role Constraint Argument to which the Model Element belongs. Foreign Key reference to the MetaModelRoleConstraintArgument table.
RoleId	Represents the Role that is part of the Join Path defined (in part) by instances stored in this table. Foreign Key reference to the MetaModelRole table.
SequenceNr	The ordinal position within the Role Path that the Role identified by this tuple plays in a Join Path for a Role Constraint Argument.

See also:

The MetaModelRoleConstraint table
The MetaModelRoleConstraintArgument table



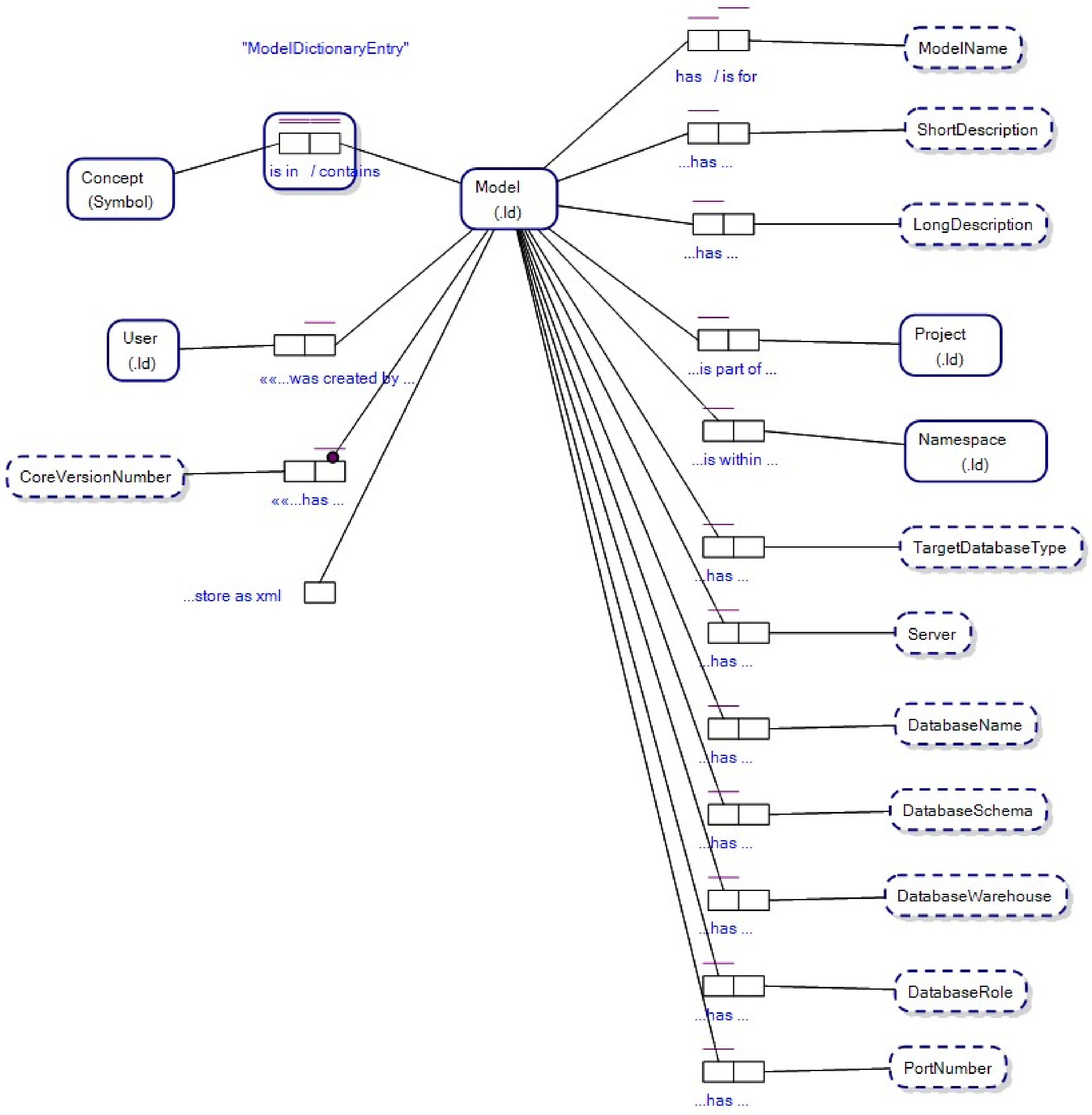
3.1.7. The MetaModelModel table

The MetaModelModel table stores details of a Model.

⚠ **IMPORTANT** Never delete the Model called, "English". This model is required by the Virtual Analyst functionality in Boston.
⚠ **IMPORTANT** Never delete the Model called, "Core". This model is required to create ER Diagrams or Property Graph Schemas in Boston.

Field	Description
ModelId	The unique identifier of the Model to which the Model Element belongs.
ModelName	The name of the Model.
ShortDescription	Not used in this version of Boston. A short description of the Model.
LongDescription	Not used in this version of Boston. A long description of the Model.
EnterpriseId	Not used in this version of Boston. Provided for consistency with the Richmond product by FactEngine.
SubjectAreaId	Not used in this version of Boston. Provided for consistency with the Richmond product by FactEngine.
ProjectId	Used in the Boston Online product/configuration of Boston. I.e. When Boston is in Client/Server mode with Projects, Users, Groups, Functions, Roles, Permissions.
ProjectPhaseId	Not used in this version of Boston. Provided for consistency with the Richmond product by FactEngine.
SolutionId	Not used in this version of Boston. Provided for consistency with the Richmond product by FactEngine.
IsConceptualModel	Not used in this version of Boston. Provided for consistency with the Richmond product by FactEngine.
IsPhysicalModel	Not used in this version of Boston. Provided for consistency with the Richmond product by FactEngine.
IsNamespace	Not used in this version of Boston. Provided for consistency with the Richmond product by FactEngine.
IsEnterpriseModel	Not used in this version of Boston. Provided for consistency with the Richmond product by FactEngine.
TargetDatabaseType	Not used in this version of Boston. Provided for consistency with the Richmond product by FactEngine.
TargetDatabaseConnectionString	Not used in this version of Boston. Provided for consistency with the Richmond product by FactEngine.
NamespaceId	The Namespace the Model is included within; especially used when Boston is run under a Client/Server environment.
CreatedByUserId	The User in the Client/Server model that created the Model.
CoreVersionNumber	The version of the Core MetaModel injected within the Model as part of the 4-Layer Architecture model of Boston. Used such that the Core MetaModel injection can be kept up to date.
Server	E.g. As required for connecting to Snowflake or TypeDB databases.
DatabaseName	E.g. As required for connecting to Snowflake or TypeDB databases.
Schema	E.g. As required for connecting to Snowflake databases.
Warehouse	E.g. As required for connecting to Snowflake databases.
Role	E.g. As required for connecting to Snowflake databases.
Port	E.g. As required for connecting to Snowflake or TypeDB databases.
StoreAsXML	True if the Model is stored as XML external to the Boston database, in which case only the MetaModelModel table stores information about the

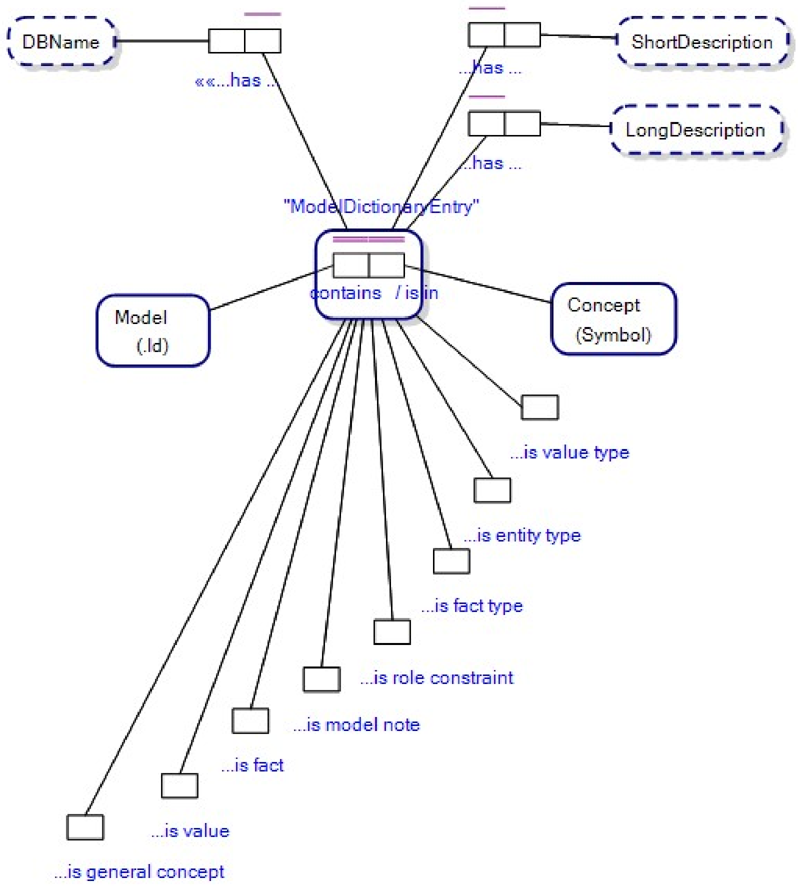
	Model. All other Model Element data is then stored as XML external to the database. NB A copy of the Model Element data (i.e. the rest of the Model) can be stored redundantly within the Boston database, but if stored as XML, the Model is loaded into memory (for use by Boston) by deserialising the Model stored in XML.



3.1.8. The Model Dictionary (MetaModelModelDictionary table)

Each Model in Boston has a Model Dictionary containing a set of the Concepts utilised by that Model. Each entry in the MetaModelModelDictionary table stores the Symbol of the Concept that the 'Dictionary Entry' represents and the Subtype of Concept that the Dictionary Entry represents (e.g. Entity Type, Value Type, Fact Type, Role Constraint, Model Note, Fact or Value).

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
Symbol	The unique Symbol within the Model Dictionary. Foreign Key reference to the MetaModelConcept table.
ShortDescription	A short description of the Model Element.
LongDescription	A long description of the Model Element.
IsEntityType	Set to True if the value of 'Symbol' is a unique identifier for an Entity Type within the Model.
IsValueType	Set to True if the value of 'Symbol' is a unique identifier for a Value Type within the Model.
IsFactType	Set to True if the value of 'Symbol' is a unique identifier for a Fact Type within the Model.
IsFact	Set to True if the value of 'Symbol' is a unique identifier for a Fact within the Model.
IsValue	Set to True if the value of 'Symbol' is a unique identifier for a Value within the Model.
IsRoleConstraint	Set to True if the value of 'Symbol' is a unique identifier for a Role Constraint within the Model.
IsModelNote	Set to True if the value of 'Symbol' is a unique identifier for a Model Note within the Model.
IsGeneralConcept	True if the Concept/Symbol is valid within the Model/Universe of Discourse but not yet assigned to one of Value Type, Entity Type, Fact Type, Role Constraint, Model Note, Fact or Value. **NB** Only used if the Model Elements of the Model are stored within the Boston database and not stored as XML outside of the Boston database. This is because the XML schema does not have an Element for Model Dictionary...only the Model Elements themselves, so a 'General' concept cannot be stored in the XML format.
DBName	For Entity Types or (Objectified) Fact Types, the name of the corresponding entity within the target database of the Model, especially if the database name of the entity is not the same as the corresponding Model Element in the Boston Model.

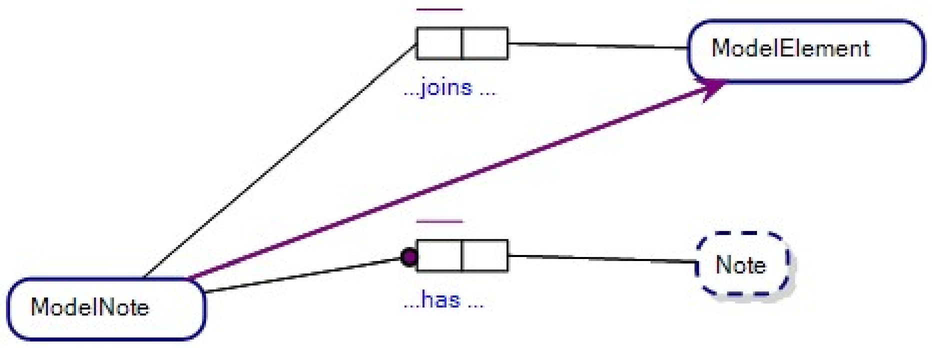


Model Dictionary – Object-Role Model

3.1.9. The MetaModelModelNote table

The MetaModelModelNote table stores details of Model Notes.

Field	Description
ModelNoteId	The unique identifier of the Model Note. **NB** To find which ModelNotes are within a Model, search the MetaModelModelDictionary table.
Note	The note text of the Model Note.
JoinedObjectTypeId	The unique identifier of the Object Type joined to the Model Note when displaying the Model Note within the GUI. Represents the Object Type to which the Model Note applies.
IsMDAModelElement	True if the ModelNote is part of the Core MetaModel injected within Models within the 4-Layer Architecture used by Boston, else False.
ModelId	The Model to which the ModelNote belongs. Foreign Key reference to the MetaModelModel table.



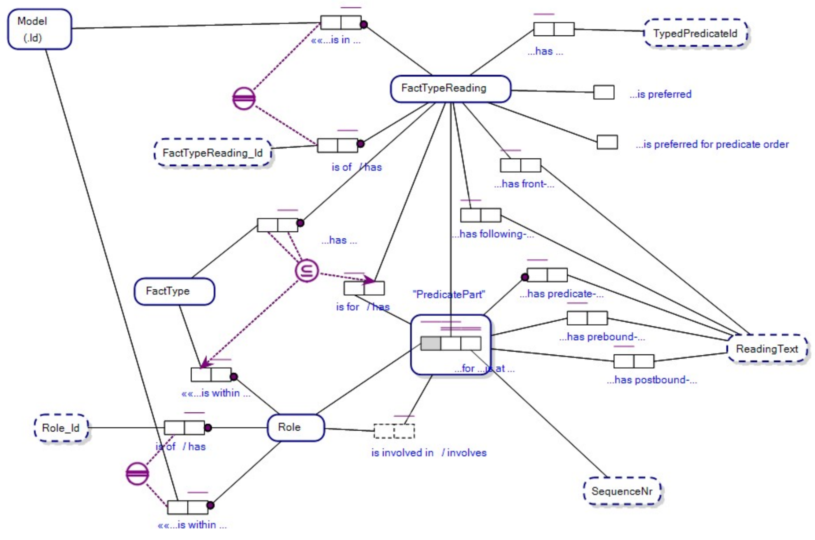
3.1.10. The MetaModelPredicatePart table

The MetaModelPredicatePart table stores details of the Predicate Parts of a Fact Type Reading.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
FactTypeReadingId	Foreign Key reference to the MetaModelFactTypeReading table. The Fact Type Reading to which the Predicate Part belongs.
SequenceNr	The ordinal position of the Predicate Part within the set of Predicate Parts for a Fact Type Reading. e.g. in the predicate, "Part is in Bin in Warehouse", the Predicate Part, 'is in', is the first in the sequence of Predicate Parts, and this field will have a value of '1'.
Symbol1	Not used
Symbol2	Not used
PredicatePartText	The part of the predicate represented by this Predicate Part. e.g. In the predicate, 'Part is in Bin in Warehouse', the first Predicate Part in the sequence of Predicate Parts for the predicate will have a value set to 'is in' for this field.
RoleId	The Role that joins to the Model Element for which this Predicate Part record provides Predicate Part Text. Foreign Key Reference to the MetaModelRole table. E.g. In the Fact Type with a reading, "Part is in Bin in Warehouse", for the Predicate Part Text, "is in", RoleId is a reference to the Role that joins to the 'Part' Model Element/Object Type.
PreboundText	In a Fact Type Reading, "Person has first-Name", "first-" is a Prebound Text.
PostboundText	In a Fact Type Reading, "Person has Account-number", "-number" is Postbound Text.

See also:

The MetaModelFactTypeReading table
The MetaModelFactType table

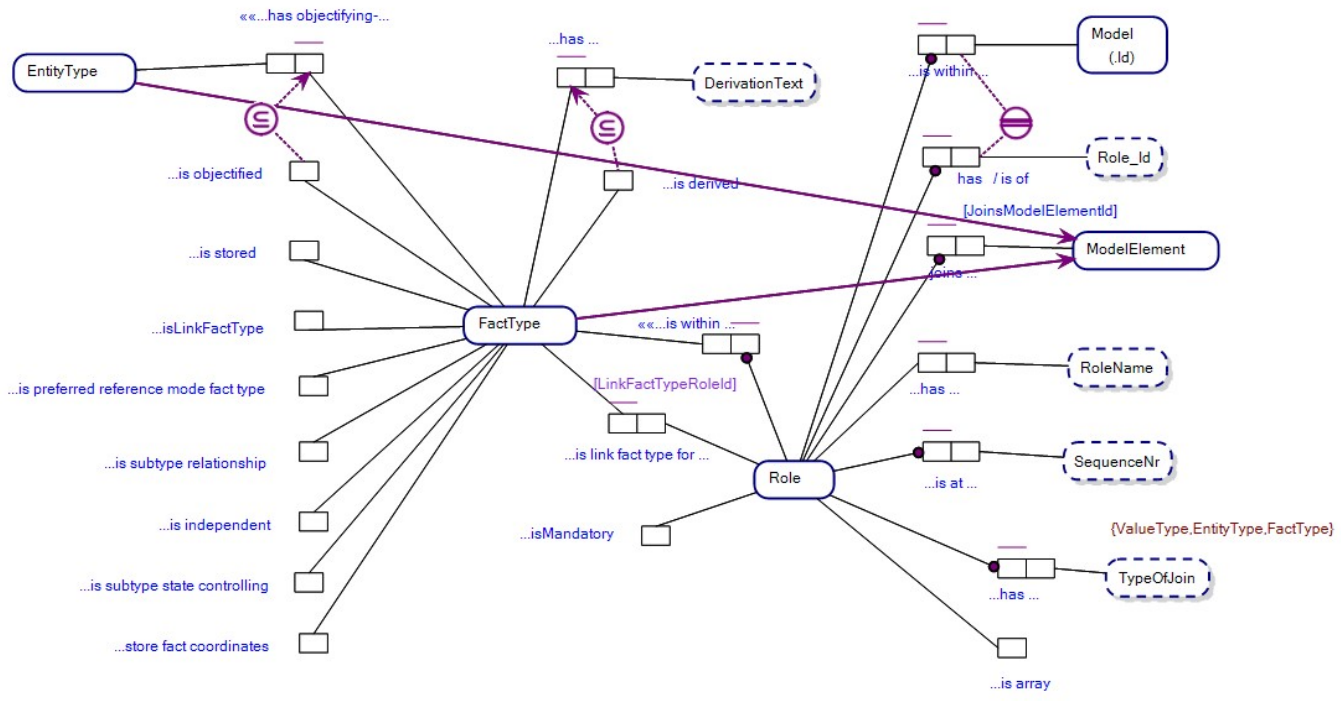


3.1.11. The MetaModelRole table

The MetaModelRole table stores details about the Roles of a Fact Type.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
RoleId	The unique identifier of the Role.
RoleName	If the Role has a Role Name, set to the 'Role Name' of the Role, else NULL.
FactTypeId	Foreign Key reference to the Fact Type table. The Fact Type to which the Role belongs.
PartOfKey	Not used in this version of Boston.
Cardinality	Not used in this version of Boston.
SequenceNr	The ordinal position of the Role within the set of Roles of a Fact Type. Especially used in Binary Fact Types that have Ring Constraints.
TypeOfJoin	'1' if the Role is hosted by an Entity Type; '2' if the Role is hosted by a Value Type; or '3' if the Role is hosted by a Fact Type.
IsMandatory	Set to True if the Role is Mandatory, else set to False.
IsArray	**NB** Only used for Models with NoSQL target databases. True if the Role (joined to a Model Element) represents an Array (stored in a NoSQL database such as MongoDB as a JSON array), rather than a separate Many-to-Many Table/Entity. I.e. The Role indicates that the array information forms part of the document for the joined Model Element. E.g. If Person drives Car is a Fact Type, and a person can drive many cars, and the information is stored as an array in the database, the Role joined to Person in the Fact Type would be flagged as 'IsArray'.
JoinsModelElementId	The Model Element joined by the Role. **NB** Ids of Entity Types, Value Types, Fact Types in Boston are the concept's Symbol itself. So if the Role joins to a Entity Type called 'Person', then this field stores 'Person'.

See also:
The MetaModelFactType table



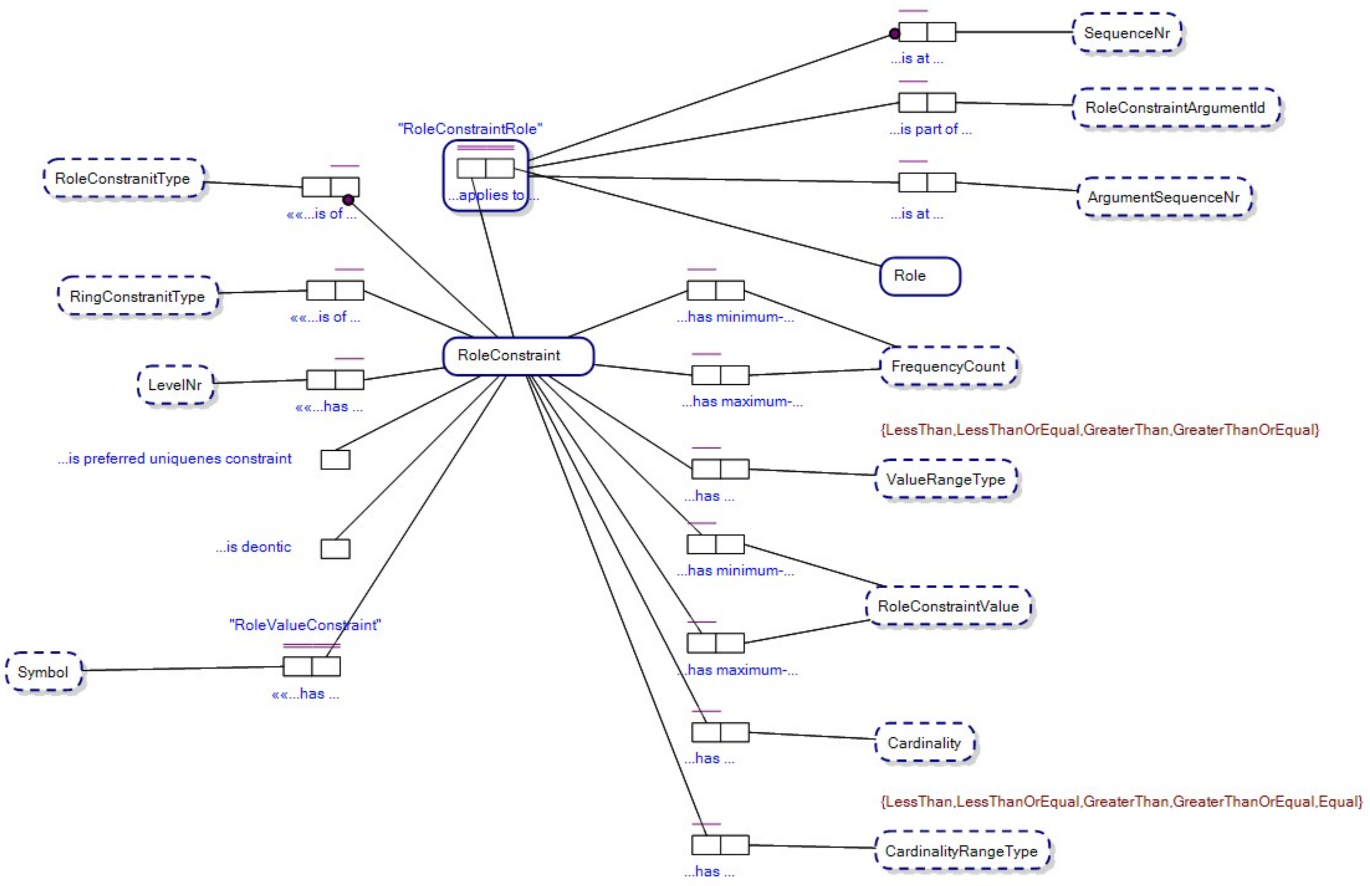
Role – Object-Role Model

3.1.12. The MetaModelRoleConstraint table

The MetaModelRoleConstraint table stores details of Role Constraints.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
RoleConstraintId	The unique identifier of the Role Constraint.
RoleConstraintName	The name of the Role Constraint. Must be the same as RoleConstraintId
RoleConstraintType	The type of Role Constraint represented by the element.
RingConstraintType	If the RoleConstraintType attribute of the Role Constraint element has a value of ‘RingConstraint’ then this attribute has a value of the type of Ring Constraint.
LevelNr	If the RoleConstraint is an, 'InternalUniquenessConstraint', then set to the displaying 'Level' in the hierarchy of Internal Uniqueness Constraints for the Fact Type to which the Role Constraint applies.
IsPreferredUniqueness	Has a value set to ‘true’ if the Role Constraint represented by the element represents the ‘Preferred Identification Scheme’ for an Entity Type.
IsDeontic	Has a value set to ‘true’ if the Role Constraint is ‘Deontic’, else has a value set to ‘false’.
MinimumFrequencyCount	If the RoleConsrtaintType attribute has a value of ‘FrequencyConstraint’ then this attribute has a value set to the minimum in the range of the Frequency Constraint.
MaximumFrequencyCount	If the RoleConsrtaintType attribute has a value of ‘FrequencyConstraint’ then this attribute has a value set to the maximum in the range of the Frequency Constraint.
Cardinality	If a Cardinality Constraint, the number of times instances appear within Facts that apply to the Role for this Role Constraint. See “CardinalityRangeType” below.
CardinalityRangeType	‘Equal’, 'LessThan', 'LessThanOrEqual', 'GreaterThan', 'GreaterThanOrEqual'
ValueRangeType	If the Role Constraint is a Value-Comparison Constraint, stores values of 'LessThan', 'LessThanOrEqual', 'GreaterThan', 'GreaterThanOrEqual'
IsMDAModelElement	True if the Role Constraint is part of the Core Model in the 4-Layer Architecture used by Boston. MDA stands for 'Model Driven Architecture'.
GUID	Unique universal identifier for the Role Constraint. Not currently used by Boston. Can be used to store URIs etc in ontologies.
MinimumValue	Used in Value Constraints. E.g. Role Value Constraints.
MaximumValue	Used in Value Constraints. E.g. Role Value Constraints.

See also:
The MetaModelRoleConstraintRole table



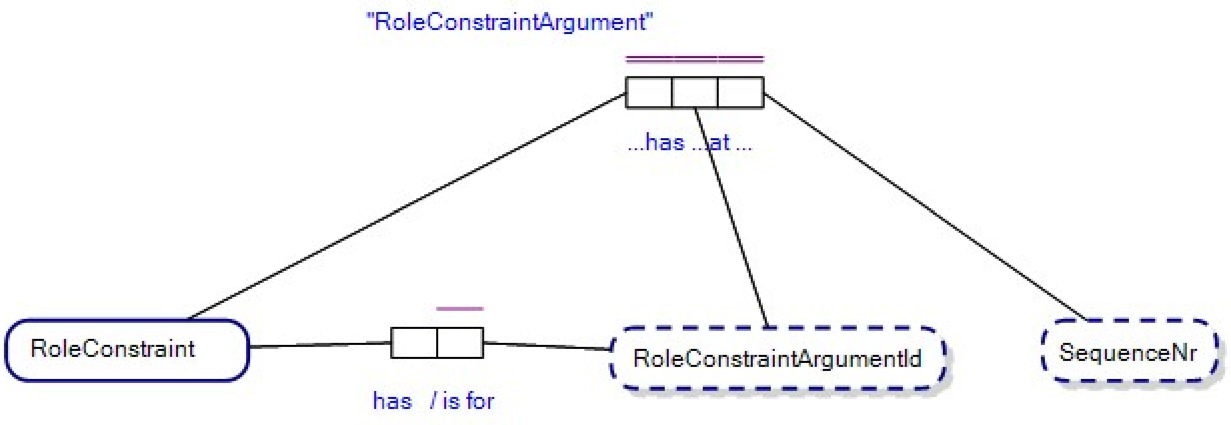
Role Constraint – Object-Role Model

3.1.13. The MetaModelRoleConstraintArgument table

The MetaModelRoleConstraintRole table stores details of the Roles linked by a Role Constraint.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
Id	The unique Id of the Argument (within the Model)
RoleConstraintId	The unique identifier of the Role Constraint to which the Argument belongs. Foreign Key Reference to the MetaModelRoleConstraint table.
SequenceNr	The ordinal position (as a Sequence Number) of the Argument within the set of Arguments for the Role Constraint. Starts at 1.

See also:
The MetaModelRoleConstraint table

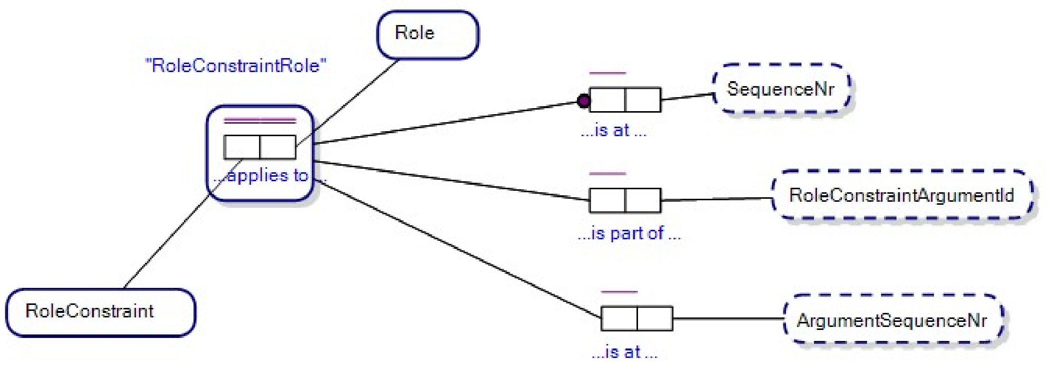


3.1.14. The MetaModelRoleConstraintRole table

The MetaModelRoleConstraintRole table stores details of the Roles linked by a Role Constraint.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
RoleConstraintId	Foreign Key reference to the MetaModelRoleConstraint table. The Role Constraint to which the reference to a 'Role' of that Role Constraint applies.
RoleId	Foreign Key reference to the MetaModelRole table. The Role referenced by the Role Constraint identified in the field, 'RoleConstraintId'.
SequenceNr	The ordinal position of the RoleConstraintRole within the sequence of RoleConstraintRole elements for a RoleConstraint.
IsEntry	**Not used.** Used only for SubsetConstraint, RoleConstraints. Has a value set to ‘true’ is the RoleConstraintRole element represents a RoleConstaint/Role link that is the ‘Entry’ link of a ‘Entry/Exit’ combination of RoleConstraint/Role links, else has a value set to ‘false’. e.g. If a link to a Role from a RoleConstraint is an ‘Entry’ link of a Subset Constraint, the values of Instances referenced by that Role are the subset of Instances referenced by the Role of a corresponding RoleConstraint/Role link that is an ‘Exit’ link.
IsExit	**Not used.** Used only for SubsetConstraint, RoleConstraints. Has a value set to ‘True’ is the RoleConstraintRole element represents a RoleConstaint/Role link that is the ‘Exit’ link of a ‘Entry/Exit’ combination of RoleConstraint/Role links, else has a value set to ‘False’.
ArgumentId	The Id of the Argument to which the RoleConstraintRole belongs, if the corresponding Role Constraint requires Arguments.
ArgumentSequenceNr	The SequenceNr of the RoleConstraintRole within the set of RoleConstraintRoles for the Argument to which this RoleConstraintRole belongs. Starts at 1.

See also:
The MetaModelRoleConstraint table

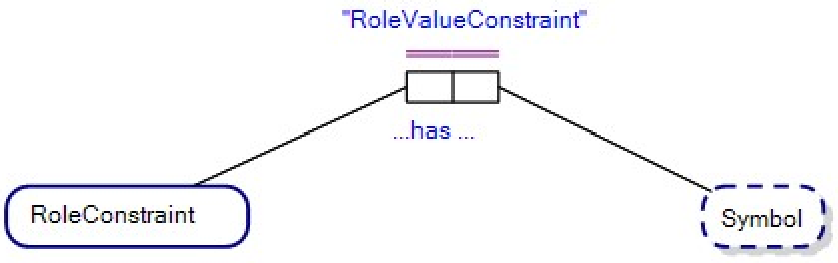


3.1.15. The MetaModelRoleValueConstraint table

The MetaModelRoleValueConstraint table stores Role Value Constraint values that relate to a Role.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs. Foreign Key reference to the MetaModelModel table.
RoleConstraintId	The Role Constraint to which the Role Value Constraint value relates. Foreign Key reference to the MetaModelRoleConstraint table.
Symbol	The value for the Role Value Constraint.

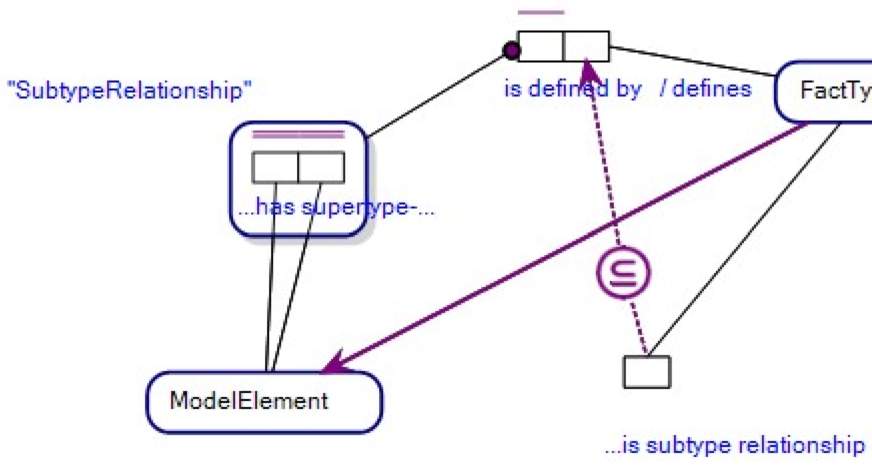
See also:
The MetaModelRoleConstraint table



3.1.16. The MetaModelSubtypeRelationship table

The MetaModelParentEntityType table stores details of the supertypes of subtypes within the Model.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
ObjectTypeId	Foreign Key reference to the MetaModelEntityType table. The Entity Type that is a Subtype of another Object Type.
SupertypeObjectTypeId	Foreign Key reference to the MetaModelEntityType table. The Entity Type that is the Supertype of the Entity Type identified by the value in the 'EntityTypeId' field of this table. **NB** May be the unique identifier of the Objectifying Entity Type of a Fact Type.
SubtypingFactTypeId	Foreign Key reference to the MetaModelFactType table. Represents the Fact Type that provides the Identity Relationship between the Subtype and the Supertype.
IsPrimarySubtypeRelationship	True if provides the Identification Path for a the Subtype.

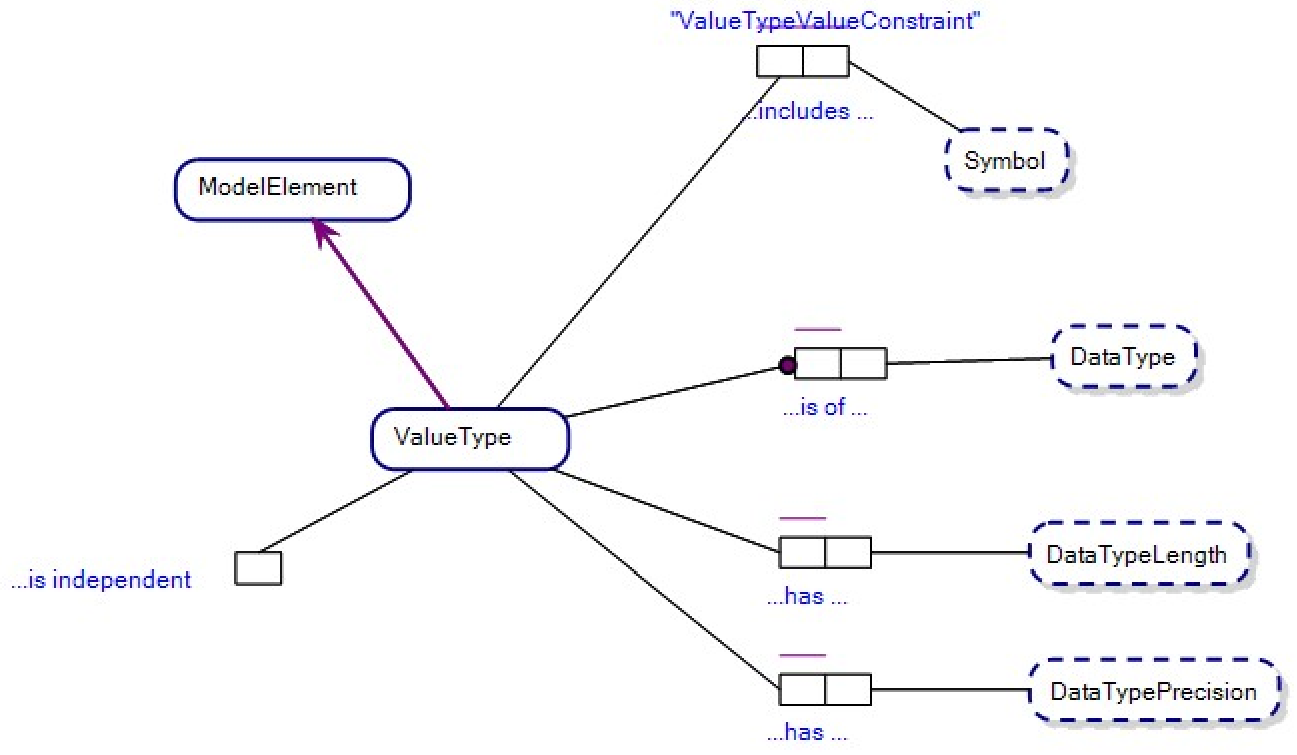


3.1.17. The MetaModelValueType table

The MetaModelValueType table stores details of Value Types.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
ValueTypeId	The unique identifier of the Value Type.
ValueTypeName	The name of the Value Type. Must be the same as ValueTypeId.
DataType	The Data Type of the Value Type.
DataTypeLength	The Length of the DataType. E.g. 100 where the Data Type is TextFixedLength limited to 100 characters. Default is 0.
DataTypePrecision	The Precision when for Numeric Data Types with a Precision Value. E.g. Double Precision Numerical data type.
IsMDAModelElement	True if the Value Type is part of the Core Model in the 4-Layer Architecture used by Boston. MDA stands for 'Model Driven Architecture'.
GUID	Unique universal identifier for the Value Type. Not currently used by Boston. Can be used to store URIs etc in ontologies.
IsIndependent	True if the Entity Type is independent (in an ORM sense).

See also:
The MetaModelValueTypeConstraint table

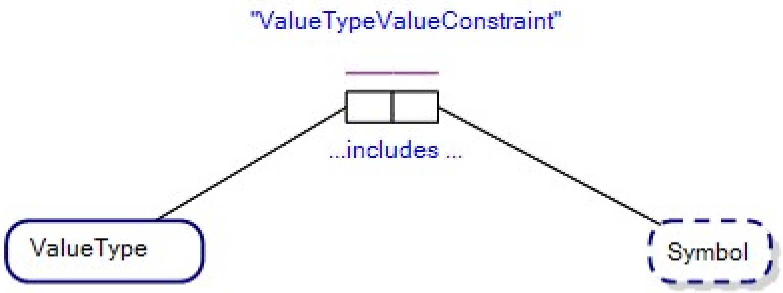


3.1.18. The MetaModelValueTypeConstraint table

The MetaModelValueTypeConstraint table stores details of Value Type Constraints.

Field	Description
StartDate	Unused at present
EndDate	Unused at present
ModelId	The unique identifier of the Model to which the Model Element belongs.
ValueTypeId	Foreign Key reference to the MetaModelValueType table. The Value Type for which the Value Type Constraint applies.
Symbol	If the Value Constraint is not a range of values, the unique data value within the set of values that constrain the values of the referenced Value Type. **NB** Value ranges are stored as text in the following format, "{1..n}"

See also:
The MetaModelValueType table

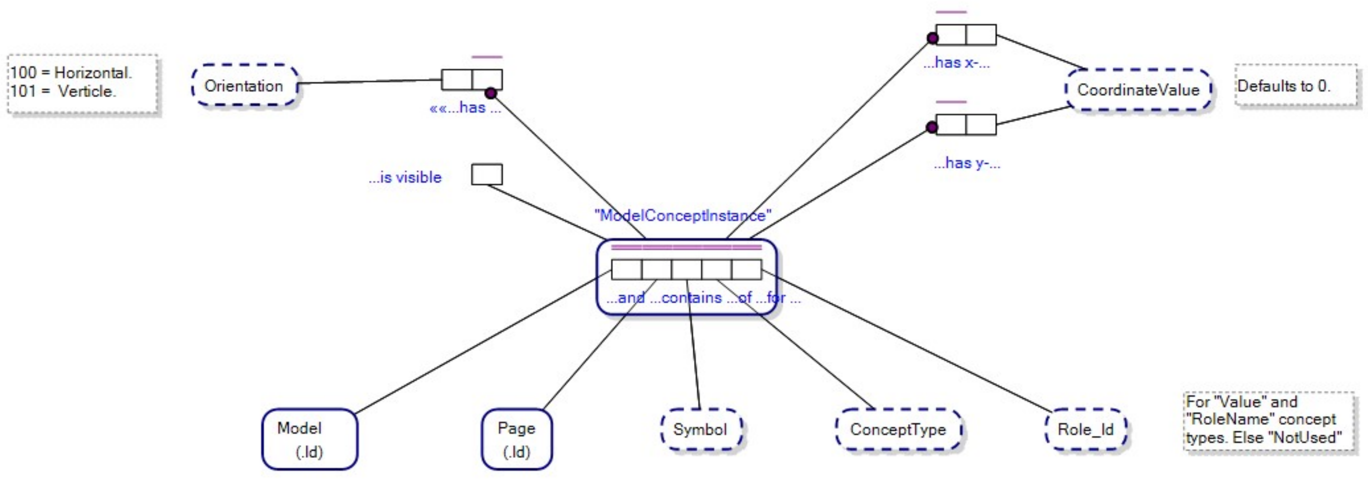


3.1.19. The ModelConceptInstance table

The ModelConceptInstance table stores the set of Concepts (identified by their 'Symbol') that appear on a Page within a Model.

Field	Data Type	Description
StartDate		Unused at present
EndDate		Unused at present
ModelId		The unique identifier of the Model to which the Model Element belongs.
PageId		Foreign Key reference to the Page table. The Page for which the Model Element is included on that Page within the Model.
Symbol		The unique identifier of the Model Element included on the Page within the Model.
ConceptType		The Concept Type of the Model Element included on the Page within the Model.
RoleId	Text	Where the 'ConceptType' of the Model Element is 'RoleName' or 'Value', set to the RoleId of the corresponding 'Role Name' or the 'Fact Data' to which the Model Element belongs. **NB** Else set to “NotUsed”
X	Number	The X-Coordinate of the Model Object as it appears on the Page within the Model. i.e. Within the GUI interface displaying the Page, the Y-Coordinate of the Model Element. **NB** Where the X-Coordinate of the Model Element is not required to display the Model Element within the Page, set to '0'.
Y	Number	The Y-Coordinate of the Model Object as it appears on the Page within the Model. i.e. Within the GUI interface displaying the Page, the Y-Coordinate of the Model Element. **NB** Where the X-Coordinate of the Model Element is not required to display the Model Element within the Page, set to '0'.
Orientation	Number	Not used by this version of Boston. Always set to '100' (Horizontal) in this version of Boston.
IsVisible	Boolean	Not used by this version of Boston. Always set to False in this version of Boston.

See also:
The ModelPage table



Concept Instance – Object-Role Model

3.1.20. The ModelLanguage table

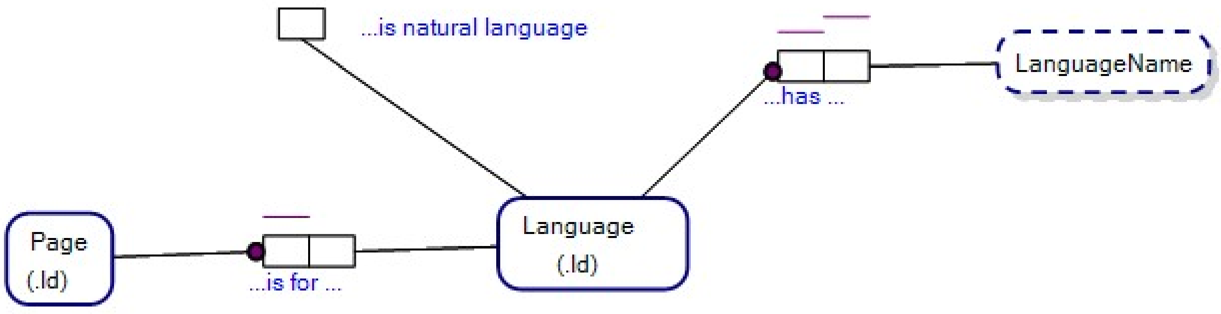
The ModelLanguage table stores details about Languages.

"Pages" in Boston are interpreted by their language. ORM Models/Diagrams are displayed as ORM Diagrams within Boston because they are interpreted as ORM Diagrams. Each 'Page' is flagged as to the language of interpretation.

In this version of Boston, all Pages are interpreted as ORM Models/Diagrams.

Field	Description
LanguageId	The unique identifier of the Language
LanguageName	The name of the Language. E.g. "ORM", "Entity Relationship Diagram", "English", “PropertyGraphSchema”, “StateTransition Diagram”
IsNaturalLanguage	Set to True if the Language is a naturally spoken Language by people.

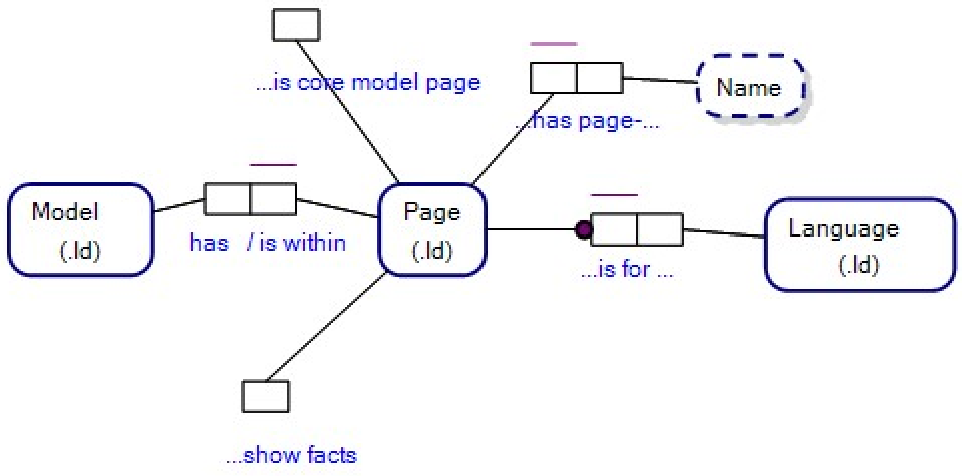
See also:
The ModelPage table



3.1.21. The ModelPage table

The ModelPage table stores details about each 'Page' within a Model.

Field	Description
PageId	The unique identifier of the Page.
PageName	The name of the Page.
ModelId	The unique identifier of the Model to which the Page belongs.
LanguageId	Foreign Key Reference to the ModelLanguage table. Represents the Language that should be used to interpret Models stored within the MetaModel. **NB** Boston currently supports the interpretation of ORM Models, Entity Relationship Diagrams, Property Graph Schema and State Transition Diagrams from within the ORM MetaModel/MetaMetaModel. Contact FactEngine for a paper on how this works. Boston incorporates a 4-Layer Architecture with different languages as logical injections within the ORM metamodel.
IsCoreModelPage	Not used in this version of Boston. Provided for consistency with the Richmond product by Viev Pty Ltd.
ShowFacts	Set to True if the Fact Table of Facts is to be displayed to the user on loading of the Page within the graphical user interface (GUI).



End of Document