

## The EAV Model

### Intro

In this section, I will provide a step-by-step explanation of the entity attribute value (EAV) model approach as implemented in the MsAccess tool. To gain a thorough understanding of how the tool operates and the EAV model itself, I suggest working with the MsAccess tool directly, using the provided training materials. Once you've had hands-on experience with the tool, it proves beneficial to explore the actual EAV data model. This article provides you with the knowledge about the EAV data model created in the MsAccess tool.

I will now proceed to describe each table included in the tool, along with a description and sample data, to help you grasp the concept more effectively.

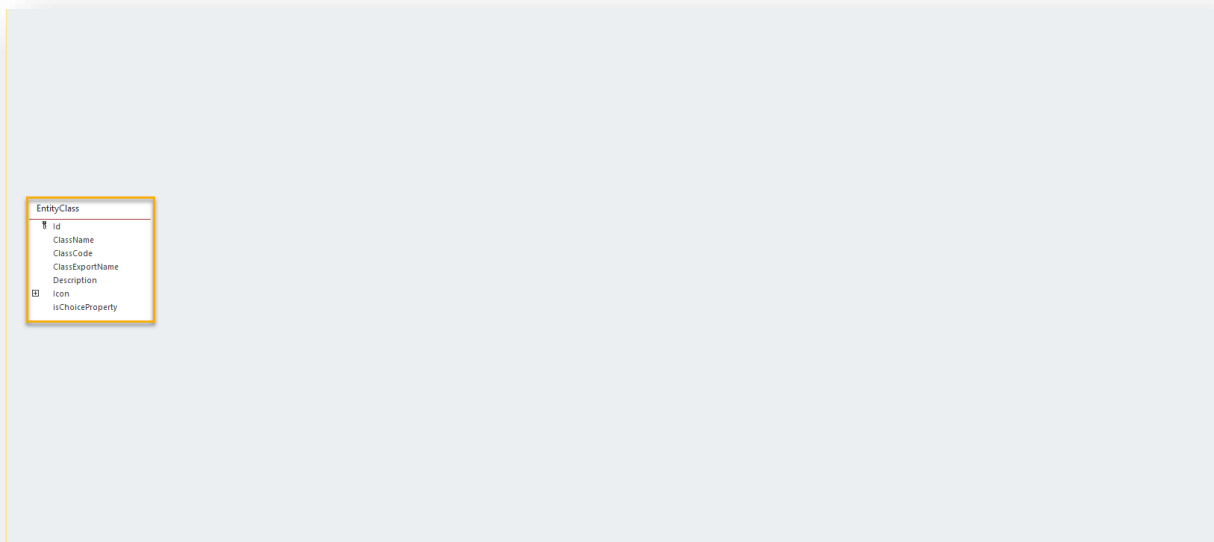
(Please check the disclaimer at the end of this document regarding the terminology used, if desired.)

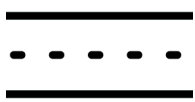


## The datamodel tables

In this chapter, The Data Model Tables, all tables are explained based on which we can configure the data model.

### Table EntityClass

The initial table in the EAV model is called ENTITYCLASS. This table serves as a listing of the subjects for which you wish to register records (instances). You can draw a parallel between the records in this table and the main tables in a traditional relational model. In the context of the electricity industry, examples of entitieclasses could include connector, grid area, business partner, ENTSO-e role, contract, and more.



Example records in table EntityClass			
Id	4	5	6
Classname	Road	Place	Material
ClassCode	Road	PLC	Mater
ClassExportName <sup>1</sup>	Not in use	Not in use	Not in use
Description	www.infoaboutentityclass.com	www.infoaboutentityclassroad.com	www.infoaboutmaterial.com
IsChoiceProperty	0 ( <input type="checkbox"/> )	0 ( <input type="checkbox"/> )	-1 ( <input checked="" type="checkbox"/> )
Icon			

Within the ENTITYCLASS table, we store information related to each subject (i.e. the entity / class). This includes the Classname, Classcode, and optionally the ClassExportName and a Description for the class. The "Id" field is a numeric value that is automatically generated, providing a unique identifier for each record within this table. While the ClassName and ClassCode are visible in the tool, ClassExportName is not visible. In the Description column, you can enter a web address for each entity class, where you refer to the page where you store metadata about the entity class. Additionally, there is an option to include an Icon<sup>2</sup>, which will be displayed in the graphical user interface (GUI) when accessing the data.

The IsChoiceProperty field is a boolean field that serves as an indicator for whether the class is associated with master data or reference data. In other words, it helps determine if the class is introduced to categorize an instance (record) of another class or not. By setting this field to true or false, you can indicate the nature of the class and its role in the data categorization process:

- For the "Road" entity-class, instances within this class typically represent self-contained instances. To reflect this, set the IsChoiceProperty field to "No"  for this class.
- In the "Material" entity-class, instances are registered to classify instances of another entity-class, making it a reference data class. For example, a "Road" instance (from the "Road" class) can be associated with a specific material. Therefore, for classes like "Material," it is recommended to set the IsChoiceProperty checkbox to "Yes"  unless storing materials is a core business requirement rather than a categorization feature. Other typical reference classes may include color, customer type, voltage level, and more.

<sup>1</sup> This column is used in TenneT's production concept for a specific reason (as a key field for data exchange), but for the purpose of explaining the concept, this field is not relevant.

<sup>2</sup> I recommend visiting [iconarchive.com](http://iconarchive.com) for a wide selection of great icons. You can find the icons used in this explanation and documentation on that website. It provides a vast collection of icons that you can explore and choose from. Icons used: [iconsmind.com](http://iconsmind.com)

## Table Attribute

The second table introduced is `ATTRIBUTE`. Whereas the records in `ENTITYCLASS` could be seen as main tables in a relational model, the records in the `ATTRIBUTE` table can be compared to the relational equivalent column headers or referential tables.

Basically, they are the "questions" you want to ask about an instance related to an entity class. Think of a name, a BSN number or a license plate number. But also think of number of seats or doors (in a car), number of m<sup>2</sup> or m<sup>3</sup> (in a house) or number of litres (capacity of a swimming pool or your favourite bottle of wine).

The `Id` is a numeric unique value, that is automatically generated. We record the `Attribute` itself (i.e. the title), a `Description` and a `ValidationRule` (You can optionally enter an Excel formula as a validation rule based on which an attribute value is validated during registration).

If a `SortOrder` is registered, the attributes in list view will be presented in the sorting order as defined (1: on top, followed by 2, followed by 3, ...). With the `OverlapAllowed` attribute, you record whether an instance is allowed to have more than one value for the same attribute at the same time. If set to 'On' , overlap is allowed.

You can register a `Hyperlink` for each attribute. This `Hyperlink` can then refer to a page where you store metadata about the respective attribute. This webpage can be accessed immediately through the tool, if you have assigned an attribute value to the applicable attribute (and a hyperlink was registered for this attribute).

Creation date/time (`Created`) is generated automatically, for information purposes only.

The last item saved in the `ATTRIBUTE` table is the `DataType`. This is where the key to the `ATTRIBUTE_DATATYPE` table is stored. You select the applicable `DataType` for the `Attribute`. (The content of) the table `ATTRIBUTE_DATATYPE` is explained in the next paragraph.



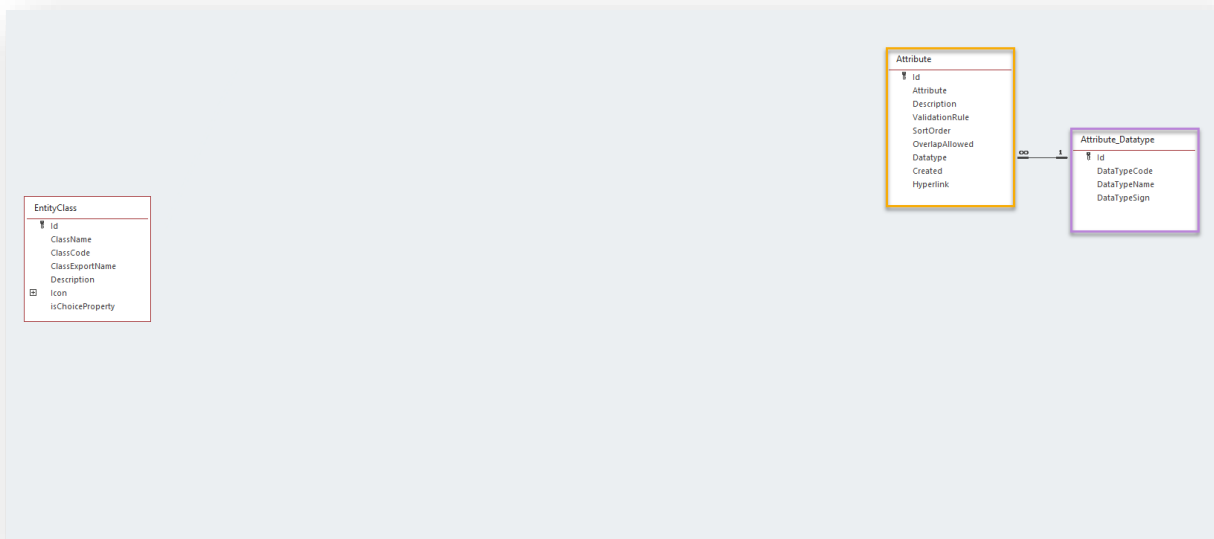
## Table Attribute\_Datatype

A datatype refers to the specific unit or category associated with an attribute.

- Names, BSN (social security) numbers, and registration numbers are classified as the *text* datatype.
- The capacity of a swimming pool is expressed in *litres*.
- The surface area of a house is expressed in square meters ( $m^2$ ).
- The maximum allowable power of an electricity connection is expressed in kilowatts (*kW*).

It is crucial to accurately register the appropriate datatype. In the application, a "text attribute" and a "hyperlink attribute" are handled differently compared to a "numeric attribute" like kWh,  $m^2$ , or litre.

Datatypes are defined within the ATTRIBUTE\_DATATYPE table.



Example records in table Attribute_Datatype			
Id	85	86	87
DataTypeCode	Text	KM	Ltr
DataTypeName	Text	Kilometer	Liter
DataTypeSign		Km	Ltr

The Id is generated automatically. DataTypeCode and DataTypeName represent the unit you need to register, serving to describe and identify that unit. DataTypeSign is an optional field where you can enter a symbol or letters. If used, the registered sign will be appended to a value when it's displayed in a list or form. For instance, instead of displaying "54" for the length of a road, you will see "54 Km" if a DataTypeSign is provided.

Example records in table Attribute					
Id	6	7	8	9	10
Attribute	Name	Kilometer	Licence plate	Code	Remark
Description <sup>3</sup>	Not in use	Not in use	Not in use	Not in use	Not in use
SortOrder	1	9	7	1	9
OverlapAllowed	<input type="checkbox"/> (No, Overlap is not allowed)	<input type="checkbox"/> (No, Overlap is not allowed)	<input type="checkbox"/> (No, Overlap is not allowed)	<input type="checkbox"/> (No, Overlap is not allowed)	<input checked="" type="checkbox"/>
Data Type (foreign key to table Attribute_Datatype)	85 (text)	86 (km)	85 (text)	85 (text)	85 (tekst)
Created	1-3-2022 13:45	1-3-2022 14:53	1-3-2022 16:03	1-3-2022 16:09	1-3-2022 16:27

### Linkingtable EntityClass\_Attribute

What is the name of a Car with registration number NP-ST-99? And How many wheels does a house have? Questions that are basically irrelevant. In order to ensure that relevant attributes are correctly associated with their corresponding classes, we introduce a linking table: ENTITYCLASS\_ATTRIBUTE. This enables us to establish the appropriate connections between attributes and classes. By making the appropriate connections, it will be impossible to ask the irrelevant questions.

In this table, the ATTRIBUTE is related via AttributeTextQuestionId to the relevant ENTITYCLASS via EntityClassId. The relationship records between ATTRIBUTES and ENTITYCLASS has a validity period (ValidFromDate & ValidUntilDate) . This validity period determines whether an attribute is available to an instance of a particular entity class on the day of entry or not. Mandatorystatus and AttrUniqueInEntityClass are characteristics that are not yet in use in this version. PrivacyRelated, a feature that indicates that the attribute value associated with an instance of a certain class is privacy-sensitive, is only available for informational purposes in this version (i.e., no functionality is attached to it yet)<sup>4</sup>.

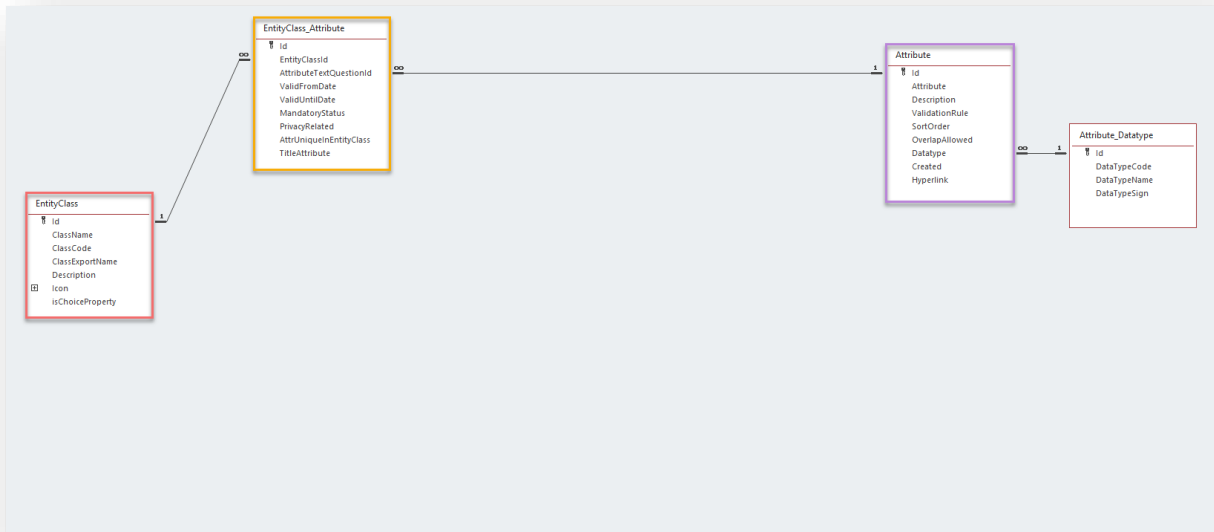
<sup>3</sup> This column is used in TenneT's production concept for a specific reason (notesection for the modeldesigner), but for the purpose of explaining the concept, this field is not relevant.

<sup>4</sup> In a future version, the intention is to link functionality to AttrUniqueInEntityClass, Mandatory Status, and PrivacyRelated.

**AttrUniqueInEntityClass:** An attribute value associated with a specific attribute may only be linked to one instance.

**Mandatory:** It enforces that a value must be entered for a specific attribute for every instance of a particular entity class.

**PrivacyRelated:** Possible to remove all privacy-sensitive attribute values for instances that fall within a search query using 1 command.



Examplerecords in table EntityClass_Attribute		
Id	1	2
EntityClassId	4 (see table EntityClass: Road)	6 (see table EntityClass: Material)
AttributeQuestionId	6 (see table Attribute: Name)	6 (see table Attribute: Name)
ValidFromDate	1-3-2023 19:00	11-3-2023 14:50
ValidUntilDate	NULL	NULL
MandatoryStatus	Yes	Yes
PrivacyRelated	No	No
AttrUniqueEntityClass	Yes	Yes
TitleAttribute (choice between 1 <sup>st</sup> , 2 <sup>nd</sup> , 3 <sup>th</sup> )	1 <sup>st</sup>	1 <sup>st</sup>

In this context, the ATTRIBUTE "Name" is associated with the ENTITYCLASS "Road." When utilizing the tool to register master/reference data, this association ensures that the "Name" ATTRIBUTE can be assigned to an instance (a record) belonging to the "Road" ENTITYCLASS. It can (for example) not be assigned to an instance of the "Car" ENTITYCLASS.

Regarding the TITLEATTRIBUTE, you have the option to select one ATTRIBUTE per ENTITYCLASS as the first TitleAttribute, another ATTRIBUTE per ENTITYCLASS as the second TitleAttribute, and one more ATTRIBUTE per ENTITYCLASS as the third TitleAttribute. When viewing overviews that elaborate on the relationships between instances, the Attribute recorded with the lowest assigned number (**1<sup>st</sup>**, followed by **2<sup>nd</sup>** if the **1<sup>st</sup>** is not recorded, and **3<sup>th</sup>** if the **2<sup>nd</sup>** and **1<sup>st</sup>** are not recorded) will be displayed. If none of these attributes are recorded, the Id of the instance will be shown.

- Sidetrip to the tool: To observe the functionality of the **1<sup>st</sup>**, **2<sup>nd</sup>** and **3<sup>th</sup>** attributes, you can follow these steps in the tool:
  - o Ensure that you have registered some data in the tool (if you have followed the step-by-step instructions, this should already be done).
  - o Select an ENTITYCLASS and navigate to the list via the button "Entity Class List".
  - o In the list view that appears, you will be able to see the **1<sup>st</sup>**, **2<sup>nd</sup>** and **3<sup>th</sup>** attributes and their values for each instance that you have linked through the ENTITYCLASS.

Id	Class	Name	Shortname	Number of inhabitants
5684	Place	Arnhem	AH	150000
5696	Place	Nijmegen	NIJ	45

- o If you select a specific instance and click on "details," you will once again see the **1<sup>st</sup>**, **2<sup>nd</sup>** and **3<sup>th</sup>** attributes in the header of the details view.
- o If the instance has any relationships with other instances, you can observe this in the relationship tab. Here, you will find a sentence displaying the **1<sup>st</sup>** attribute (which is filled in) of both named instances involved in the relationship.

Relation sentence	Validity period	RT-id	Id	EZE Id
Road A325 is the road connected to Arnhem	29-6-2023		30	5700 184

If this explanation is not entirely clear at the moment, don't worry. It will become clearer as you process your data within the prototype.

## Linkingtable Entity\_E2EType

Instances form relationships with other instances. In our example, you want to establish a connection between a road to a city. Or maybe you may desire to registrate the material composition of a road by linking it to a specific material type.

There are various types of relationships.

Each relationship type is recorded in the table ENTITY\_E2EType. In this table, you describe the relationship without actually connecting records. In addition to the **Id** (automatically generated unique number), you give each record a title: **RelationName**. You also enter a **RelationshipSentence**. For example, if you want to connect record **X** with record **Y**, the relationship sentence could be: *is the record having the color*. When you view the relationships of an instance in a list view, this relationship record will be displayed as follows: **X is the record having the color Y**.

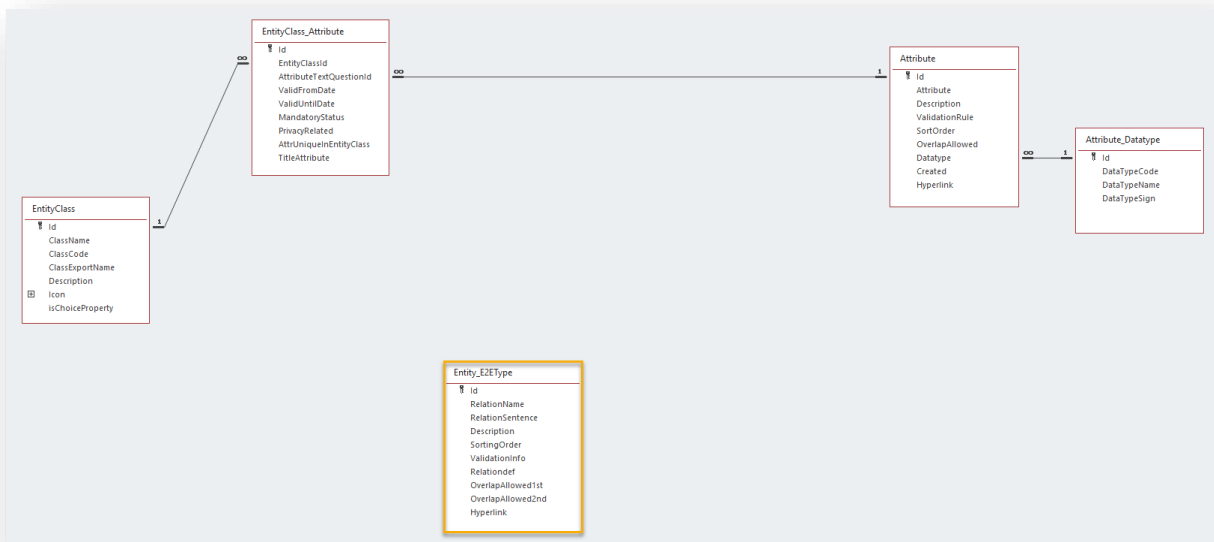




Table Entity_E2EType	
Id	Keyvalue; automatically generated number
RelationName	The name of the relationship
RelationSentence	Here the link is "articulated". The title attribute (1st element), of the first instance is shown, then the relationsentence, then the title attribute (1st element) of the second instance. <Instance 1> <RELATIONSENTENCE><Instance 2> <u>Nijmegen</u> is connected to the road <u>A325</u>
Description	Description (optionally)
Sortingorder	<u>1</u> (is shown first), <u>9</u> (will be shown after 1 (and 2, 3, 4, .., 8) (note: sort order is alphabetical)
ValidationInfo	Not working yet. In the future you can enter an excel-like-formula that will validate entries related to a specific relationtype.
Relationdef	You can optionally enter information about the type of relationship (1-n, n-1, or n-m <sup>5</sup> ). This is for informational purposes only, and there is no functionality linked to the input.
OverlapAllowed1st	This field serves to indicate that, in relation to the 1 <sup>st</sup> instance, overlapping of a relationship is permitted or not. Here's an example to clarify:  Relation type: Road is made with material. <ul style="list-style-type: none"> <li>- If OverlapAllowed1st is set to "Yes", you are permitted to register more than one material for a road, even if there is a time overlap.</li> <li>- If OverlapAllowed1st is set to "No", you can only define one material type for a road within a given period. If you attempt to register a second material type with a time overlap in this case, the input will be rejected.</li> </ul>
OverlapAllowed2nd	This field serves to indicate that, in relation to the 2 <sup>nd</sup> instance, overlapping of a relationship is permitted or not. Here's an example to clarify: Relation type: Road is made with material. <ul style="list-style-type: none"> <li>- If OverlapAllowed2nd is set to "Yes", you are allowed to register multiple roads made from the same material, even if there is a time overlap. In this case, it is possible to have multiple roads made of the material Asphalt (which is indeed the typical scenario).</li> <li>- If OverlapAllowed2nd is set to "No", you can only define one road that is made from a specific material within a given period. If you try to register a second road made from the same material, the input will be rejected (which doesn't make sense, so the setting should be: Overlap allowed is Yes).</li> </ul>
Hyperlink	In the tool, we provide the option to register a hyperlink address for each relation type. This allows direct access to the corresponding page related to the subject (relationtype) directly from the application. We can provide info concerning the relationship on those pages: capturing the essence of the relationship, storing metadata, etc. Please note: this functionality is not a core data management function. Registration, and accessing the webpages cannot be performed directly from the GUI in this version of the tool.

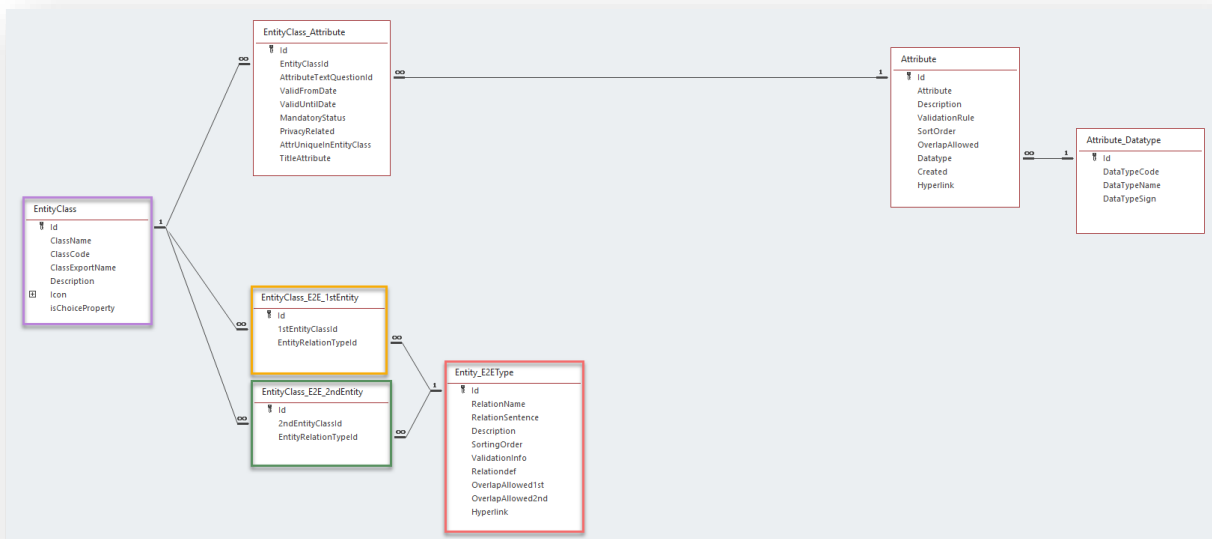
<sup>5</sup> In regards to relationships, this prototype is capable of handling all possible types; one-to-one, one-to-many, many-to-one, and many-to-many relationships.

Examplerecords in table Entity_E2EType		
Id	113	114
RelationName	City connected to Road	Road is made with material
RelationSentence	is the city connected to road	Is the road made of
Description	Elaborates what city is connected to what road	what material is the road made of
SortingOrder	5	6
ValidationInfo	<Empty>	<Empty>
Relationdef	<Empty>	<Empty>
OverlapAllowed1st	Yes	No
OverlapAllowed2nd	Yes	Yes
Hyperlink	www.confplpg_cityconnectedtoroad.html	www.confplpg_roadmadeofmaterial.html

### Linking tables EntityClass-Entity\_E2EType: EntityClass\_E2E\_1stEntity & EntityClass\_E2E\_2stEntity

When defining a relationship, it is necessary to specify the types of instances (i.e., instances belonging to which ENTITYCLASS) that can be registered as the first instance and the types of instances that can be registered as the second instance within that relationship. For instance, when registering a relationship with the name "City linked to road," you would want to limit the selection of instances as follows: only *cities* can be chosen as the first instance, and only *roads* can be chosen as the second instance.

To accomplish this, you need to associate one or more entity classes with a particular relationship as the 1<sup>st</sup> ENTITYCLASS, allowing instances from those classes to be selected as the 1<sup>st</sup> instance. Similarly, you need to associate one or more entity classes with the same relationship as the 2<sup>nd</sup> ENTITYCLASS, enabling instances from those entity classes to be chosen as the 2<sup>nd</sup> instance.



The Id, as in every table in this database, is a unique number that is automatically generated and used as an internal key. The other fields are 1st/2nd EntityClassId and EntityRelationTypeId, which are reference key fields pointing to the ENTITYCLASS and ENTITY\_E2EType tables.

Example records in table EntityClass_E2E_1stEntity	
Id	1
1stEntityClassId	5 (See EntityClass table: Place)
EntityRelationTypeId	113 (See Entity_E2EType table: City Connected to Road)

Example records in table EntityClass_E2E_2ndEntity	
Id	1
1stEntityClassId	4 (See EntityClass table: Road)
EntityRelationTypeId	113 (See Entity_E2EType table: City Connected to Road)

## The data tables

With the aforementioned steps, we have established our model (albeit a simple one, to provide a general understanding). We have defined the ENTITYCLASSES, which correspond to main tables in a relational world. We have also determined the ATTRIBUTES, representing column names or referential tables in the relational world, and associated them with their respective ENTITYCLASSES, via the table ENTITYCLASS\_ATTRIBUTE, ensuring (in a relational world that) the correct columns exist in the tables.

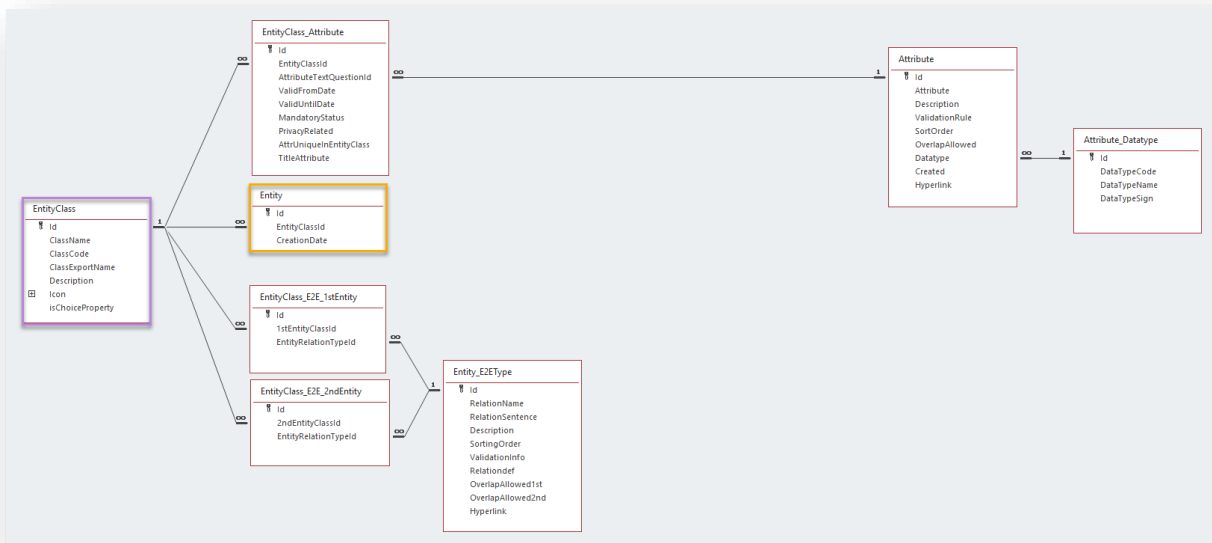
Furthermore, we have defined ATTRIBUTE\_DATATYPES such as text, hyperlink, kW, MVar, cm, litre, etc., and linked these datatypes to the corresponding ATTRIBUTES. Additionally, we have defined relationships (Entity\_E2EType) and connected them to the ENTITYCLASSES via the linking tables ENTITYCLASS\_E2E\_1ST/2NDEntity. In the relational world, these relationships are represented by drawing connections between the tables.

Once our data model has been configured, we can proceed to incorporate the actual data (register the master and reference data records, as well as the relationships between those records).

## Table Entity (Instance)

In the ENTITY table<sup>6</sup>, the actual instances (records) will be registered.

In this table, we will register the actual instances, such as the "A325" highway between Nijmegen and Arnhem or a specific place like "Nijmegen." However, we will not register the name of the road (e.g., "A325") or cityname ("Nijmegen") in this table and/or at this stage. Instead, the table will contain a unique number (Id), serving as the key field for instance (record) identification, and an EntityClassId. The EntityClassId acts as a reference to the ENTITYCLASS table, indicating that the instance belongs to a particular ENTITYCLASS, such as "Place." Additionally, a CreationDate will be automatically generated, although it holds no functional value.



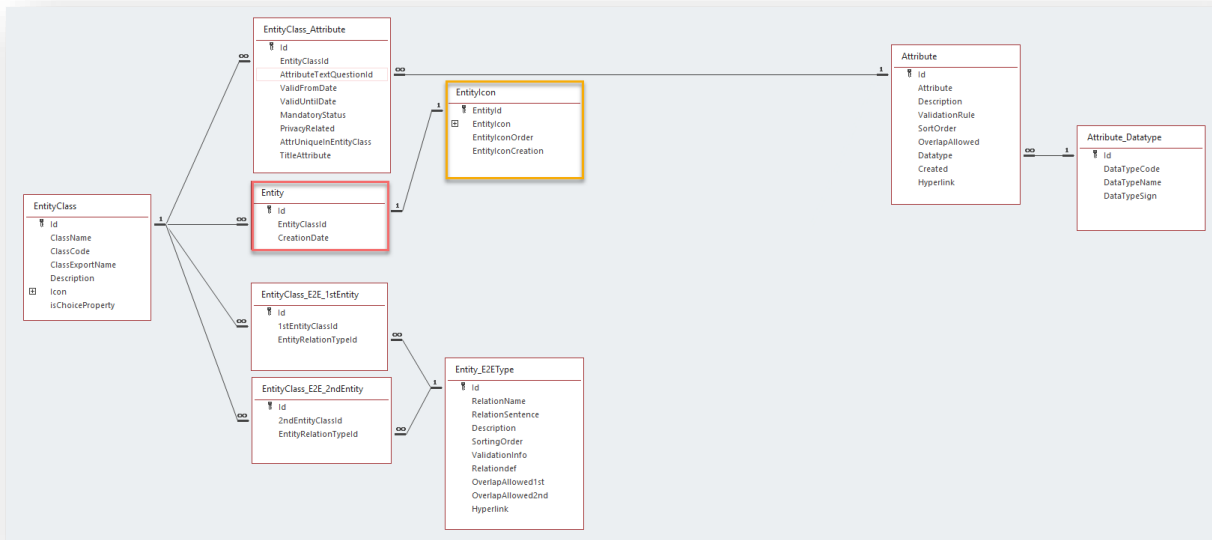
Example records in table Entity				
Id	1134	1135	1136	1137
EntityClassId	4 see EntityClass: Road	5 See entity class: Place	5 See entity class: Place	6 See entity class: Material
CreationDate	11-4-2023 17:53	11-4-2023 17:56	11-4-2023 17:57	11-4-2023 17:59





We have successfully added instances to the database. Each of the four instances has been assigned a unique Id. We have additional information indicating that the instance with Id 1134 belongs to the "Road" ENTITYCLASS, while the instances with IDs 1135 and 1136 belong to the "Place" ENTITYCLASS. Furthermore, the instance with Id 1137 is classified as a "Material" type-instance. However, apart from this classification, we do not possess any further details about the instance at this point.

<sup>6</sup> The more appropriate name for this table would be "Instance" instead of "Entity." Early in the design of this tool, I assigned an incorrect name. Unfortunately, due to the complexities involved in renaming a table in MsAccess, I have decided to retain the table with its original, less suitable name.

## Table EntityIcon

The table ENTITYICON allows us to store various attachments, including icons and other file types such as PDF, Word, Excel, and PNG, which are specifically associated with an instance. In this application, we utilize the standard attachment functionality of Microsoft Access, enabling the ability to link multiple attachments within a single record.



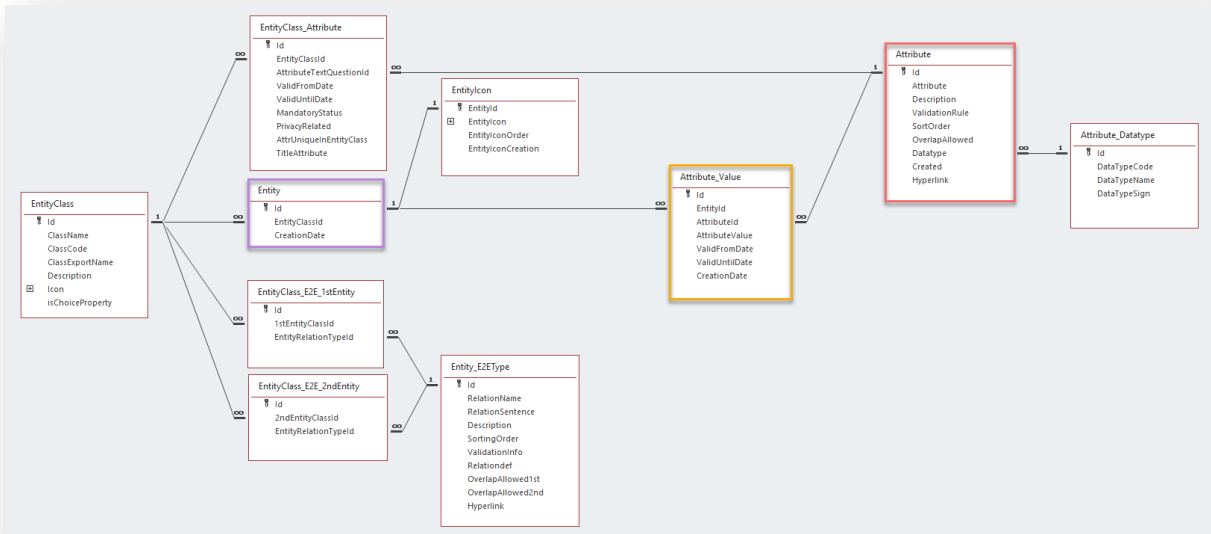
Example records in table Entity_Icon				
EntityId	1134	1135	1136	1137
EntityIcon				
EntityIconOrder (not in use)	-	-	-	-
EntityIconCreation	11-4-2023 17:53	11-4-2023 17:56	11-4-2023 17:57	11-4-2023 17:59

In this version, the "viewing" (or providing) of attachments on the screen where the details of an instance are displayed has been implemented. However, it is not yet possible to input an attachment through the screens. Functionality related to this will be offered in the next version of this tool. If you still want to use this feature, you will need to directly enter a record for the respective instance in the ENTITYICON table. The ENTITYICON table, which has a one-to-one relationship with the ENTITY table, can be accessed via the sidebar. Enter the Id of the relevant instance. In the attachment column EntityIcon (the column with a paperclip as the title), right-click and select "manage attachments", click on Add, select your attachment(s), and confirm your choice. You can add more attachments later if you wish.

I use this function purely to display an image (Icon) of an instance, such as a company logo or an image associated with a specific instance. If you add an image as the first attachment, it will be visible when you open the detail screen for the respective instance.

## Table Attribute\_Value

The next table, named ATTRIBUTE\_VALUE, establishes associations between instances (from the ENTITY table) and attributes (from the ATTRIBUTE table) along with their respective values.



Exempladata in table Attribute_value						
Id	EntityId	AttributeId	AttributeValue	ValidFromDate	ValidUntilDate	Creationdate
1	1134	6 (Name)	A325	1-4-2000		11-4-2023 15:43
2	1135	6 (Name)	Nijmegen	1-1-1230		11-4-2023 15:45
3	1136	6 (Name)	Arnhem	1-1-1233		11-4-2023 15:46
4	1137	6 (Name)	Asfalt	1-1-2000		11-4-2023 15:47
5	1135	9 (Code)	NM	1-1-1230		11-4-2023 16:45
6	1136	9 (Code)	ARN	1-1-1233		11-4-2023 16:45

In this table, an instance is assigned a unique key value: Id. Furthermore, there is a referring key called EntityId that indicates the instance to which the record in the ATTRIBUTE\_VALUE table corresponds. Similarly, another referring key called AttributeId references the ATTRIBUTE, which represents the "Question" being asked. Following this is the AttributeValue, which can be considered the answer to the question (Attribute) posed for the respective ENTITY.

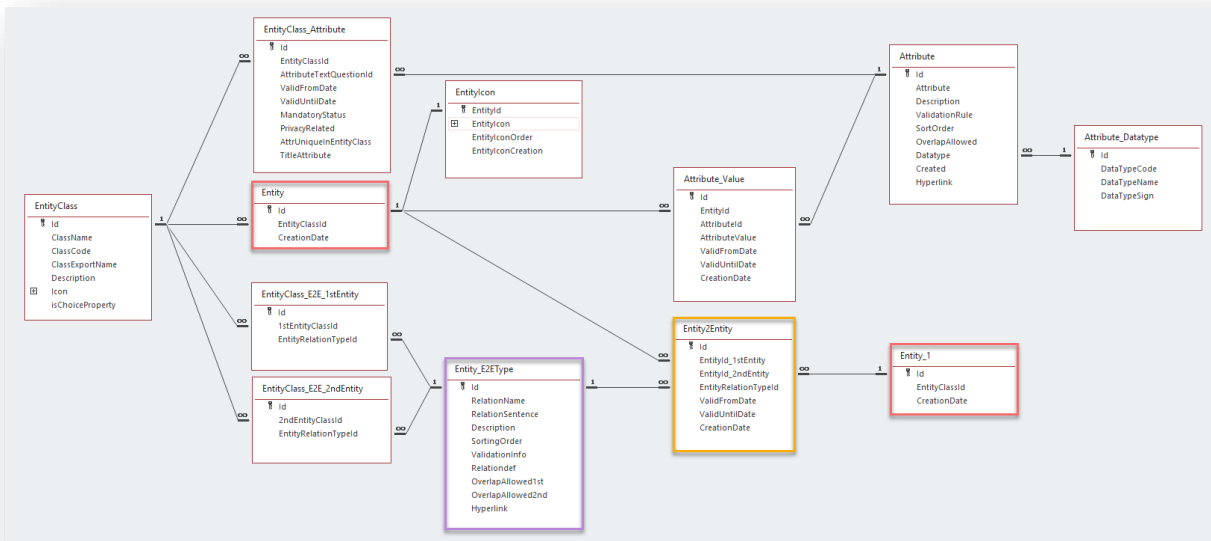
This AttributeValue has a validity period expressed through the fields ValidFromDate and ValidUntilDate. ValidFromDate represents the start date from which the value is valid, while ValidUntilDate indicates the end date. If this end date is unknown or infinite, you leave this field empty.

In record 1135, we had initially registered an instance without specific knowledge that it referred to Nijmegen. At that point, based on the entry in the ENTITY table, we only knew that it belonged to the “Place” ENTITYCLASS. Later, we associated this instance (1135) with an Attribute (Name) to capture its AttributeValue. Therefore, in the ATTRIBUTE\_VALUE table, we record the AttributeValue where the EntityId (1135) is stored, answering the question posed by the AttributeId (6) for the “Place” ENTITYCLASS. The AttributeValue in this case is "Nijmegen."

Upon entry, a CreationDate (with no functional purpose) is automatically generated and recorded in the system.

### Table Entity2Entity

Lastly, we establish relationships between instances by relating instances to other instances. This is accomplished through the ENTITY2ENTITY table. Within this table (having an Id for every record), we specify the first recorded instance (EntityId\_1stEntity), the second recorded instance (EntityId\_2ndEntity), the nature of the relationship (EntityRelationTypeId, referring to the relationtype table ENTITY\_E2ETYPE), and the period during which this relationship is applicable (ValidFromDate and ValidUntilDate). The registration date (CreationDate) is automatically generated with the date and time of the registration moment. This serves no functional purpose.



The specific relationship "type" is stored in the ENTITY\_E2ETYPE table. As an example, a record could be City Connected to Road, indicating the relationship between a city and a road.

Exempladata in table Entity2Entity						
Id	EntityId_1stEntity	EntityId_2ndEntity	EntityRelationTypeld	Valid from Date	Valid Until Date	Creation date
100	1135 Id for Nijmegen	1134 Id for A325	113 Id for Relationclass "City connected to road"	1-1-1989		15-4-2023
101	1134 Id for A325	1137 Id for Asphalt	114 Id for Relationclass "Road is made with material"	1-1-1989		15-4-2023
102						
103						
104						
105						

And thus, we have completed the process. Data modelling (via the *datamodel* tables) and MRD registration (via the *data* tables).

The data is stored according to EAV technology but can be drawn relationally. At TenneT, we extract data from operational environments and process it through normalization rules (Codd & co) in the data model using this tool. We then analyze the result with all processspecialists, and if all processes seem feasible, we represent the model relationally. Afterward, we tap into the master data from another local information silo to also incorporate that data into the model as designed in the tool. If this new information silo has implications for the current data model, we simply adjust the data model accordingly. We continue this process until all local data sources have been processed. Once we reach that point, we have generated our enterprise-wide data model.

The EAV methodology is exceptionally well-suited for this "reverse engineering" of data processing. Furthermore, for data analysts among us, this storage methodology seamlessly integrates with graph database and analysis technologies.



## The last things:

Overlap allowed (Y/N)

Attribute	Entity_E2EType
<input checked="" type="checkbox"/> Id	<input checked="" type="checkbox"/> Id
Attribute	RelationName
Description	RelationSentence
ValidationRule	Description
SortOrder	SortingOrder
OverlapAllowed	ValidationInfo
Datatype	Relationdef
Created	OverlapAllowed1st
Hyperlink	OverlapAllowed2nd
	Hyperlink

Within our system, there are two tables: ATTRIBUTE and ENTITY\_E2EType, which contain columns related to "overlap allowed." When for a record in a table this column is ticked as true, it indicates that overlap is allowed. Conversely, when it is not ticked (set false), it means that overlap is not permitted. But what exactly does "overlap" refer to?

In the ENTITY table, the instance itself does not have a validity period. At the highest level, the table just records a unique identifier (Id) used for identification purposes, as well as a reference ID (EntityClassId) to the ENTITYCLASS table, which determines the type of the entity.

However, for all other information associated with a specific instance, there is a validity period. This means that attributes such as name, code, hyperlink, comment, or any other attribute value are linked to the instance *for a specific period of time*. Similarly, in the case of defined relationships between entities, the relationship itself has a validity period.

By specifying whether overlap is allowed, we can control whether multiple attribute values of the same attribute can be linked to an instance concurrently, with each value having its own active validity period. In other words, if overlap is permitted, it is possible to have multiple attribute values of a specific attribute simultaneously linked to an instance, even if their validity periods overlap.

So, when overlap is allowed, you have the flexibility to enter multiple attribute values for a specific instance. Let's take an example with an instance having ID 567:

Instance with ID 567:

- Name: John , valid from 1-1-1989 until 31-12-2004
- Name: Mark, valid from 1-1-2002 until <empty>

In this case, on 2-1-2002, the instance with ID 567 has both the names John and Mark associated with it. However, this situation is considered undesirable because ideally, a record should have only

one name on a given date. Therefore, for the attribute "Name," the setting for "Overlap allowed" should be set to "No" to prevent such overlapping values.

On the other hand, for attributes like "Remark," you might consider allowing overlap if it serves a purpose or is deemed appropriate in your specific scenario. This means that different remarks can be associated with the instance concurrently, even if their validity periods overlap.

Concerning the relations: In the table ENTITY\_E2ETYPE, there are two columns related to overlap: OverlapAllowed1st and OverlapAllowed2nd. To provide a clearer understanding, let's use the relationship "Road is made of material." In this example, we define the following:

- OverlapAllowed1st (for the first entity): False
- OverlapAllowed2nd (for the second entity): True

This means that for the relationship "Road is made of material," we have specified that:

A road may consist of at most one type of material (such as asphalt). This implies that a road record cannot have overlapping instances concerning instances from the entity class materials.

On the other hand, a type of material can be used in multiple roads. This means that a material instance can be associated with multiple road instances, allowing for overlap.

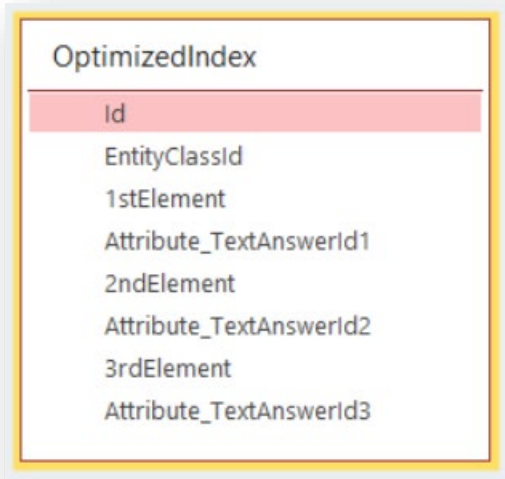
By setting OverlapAllowed1st to false and OverlapAllowed2nd to true for the "Road is made of material" relationship, we ensure that each road instance is associated with a single material type, while a material type can be used in multiple road instances, allowing for overlapping relationships.

In summary, the decision to allow or disallow overlap for specific attributes depends on the requirements and desired behavior for managing attribute values within your system.

## SortingOrder/SortOrder

The `SortingOrder` determines the sequence in which data is presented in lists. When you access an instance and view its related attribute values or related instances, the data in the lists within the application forms is displayed based on a predefined sorting order. It's important to note that this sorting order follows an alphabetical arrangement, rather than a numerical one.

## OptimizedIndex



OptimizedIndex
Id
EntityClassId
1stElement
Attribute_TextAnswerId1
2ndElement
Attribute_TextAnswerId2
3rdElement
Attribute_TextAnswerId3

EAV has a drawback when it comes to data retrieval, especially when implementing an EAV model in MsAccess, a database program designed for standard relational database models. To query data in this scenario, the standard SQL language must be utilized. However, queries tend to become more complex, leading to a decrease in performance. To address this issue and optimize performance, an auxiliary table was implemented. This table indexes data per instance based on predefined settings, the 1st, 2nd, and 3rd attribute per entity class. This technical solution serves as a performance optimization measure and is implemented solely for that purpose.

## Disclaimer

This prototype (the file) is made available as open source. The prototype is created in MsAccess.

### The terms

This file is created in MsAccess and will be provided “as is” and without any form of warranty, including without limitation merchantability, fitness for a particular purpose, absence of defects or errors, accuracy, non-infringement of intellectual property rights. In no event shall TenneT nor its employees be liable for any direct, indirect, incidental, special, exemplary or consequential damages (including, but not limited to, procurement of substitute goods or services, loss of use, data or profits, or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this MsAccess file, even if advised of the possibility of such damage. The file is not to be used for whatever production environment.

This MsAccess file does not include a license to use the MsAccess program of Microsoft, nor does it grant permission to use the trade names, trademarks, service marks, or names of TenneT and/or third parties.. The user of this MsAccess file is solely responsible for obtaining the applicable licenses from Microsoft or any other license necessary.

If you make modifications to the MsAccess file, please provide the modification including explanation to the author. The author will continuously improve the file and make all updates available.

If you use the principle and/or file as explained above and elaborated in the MsAccess file in your software and/or in a publication/article/expression (commercially or not commercially), please include the credentials of the author (TenneT / Coen Hendrikx).

#### *Be aware:*

The MsAccess tool (the file) is constantly evolving, and not fully tested. There will be bugs.

## Sources used

- Icons used in this document: [iconsmind.com](https://www.iconsmind.com) (via [iconarchive.com](https://www.iconarchive.com))
- Wikipedia
- ChatGPT

## Note

The terminology surrounding EAV (Entity-Attribute-Value) varies among different sources of information. Sometimes, entities are confused with instances, records with instances, and parameters with attributes. The terminology used in the tool is based on the knowledge available at that time. Unfortunately, it couldn't be changed as doing so would lead to issues in the code, rendering certain parts of it non-functional.

So, for clarity:

- Where I registered the table ENTITY CLASS, I meant ENTITY
- Where I registered the table ENTITY, I meant INSTANCE

There are likely to be more incorrect terms associated with attributes and tables.