

2. DATABASE MODELING

Goal: Design IDEF1X models.

2.1 Create the logical data model

- 1) Run ERWin Data Modeler and create new model using New (File -> New);
- 2) In the window “Create Model – Select Template” select “Logical/Physical”;
- 3) In the “Target Database” section select “Database” – Access, and “Version” – 2000/2002/2003. Click “Ok”;
- 4) As the result, ERWin interface will be shown (Figure 2.1).

2.2. Create entities and attributes

Required sequence of steps will be considered using the “Product groups” entity as the example:

- 1) Click “Entity”;
- 2) Move pointer to the place where entity might be placed and click the left mouse button (Figure 2.2);
- 3) Change the name of entity for which right click on it and select “Entity Properties...”, then enter the new name “Product_groups” in the appeared “Entities” window and click “Ok” (Figure 2.3);
- 4) Create entity attributes for which right click on it and select “Attributes...”, in the appeared “Attributes” window (Figure 2.4) click “New” button and in the appeared “New Attribute” window input “product_group_ID”, select domain type “Number” (Figure 2.5), and click “Ok”. Create the second attribute “product_group_name” of type “String”;

5) Set the primary key for this entity for which select attribute “product_group_ID” and click “Primary Key” (Figure 2.6);

6) Click “Ok” in “Attributes” window (Figure 2.7), and now entity “Product_groups” is created.

The rest entities that might be created on this step should be created in the same way.

Description of these entities and attributes is shown in Table 1.1.

Table 1.1

Entity	Attribute	Key	Domain
Measurement_units	measurement_unit_ID	PK	Number
	measurement_unit_name		String
Products	product_ID	PK	Number
	product_name		String
Suppliers	supplier_ID	PK	Number
	address		String
	phone		String
Price_types	price_type_ID	PK	Number
	price_type_name		String
Contracts	contract_ID	PK	Number
	supply_date		Datetime
	comment		String
Supplied_products	amount		Number
	price_per_item		Number
Legal_entities	name		String
	tax_number		String
	VAT_cert_number		String
Ind_entrepreneurs	last_name		String
	first_name		String

Entity	Attribute	Key	Domain
	second_name		String
	reg_cert_number		String

As the result, logical model that contains created entities and attributes should look as it is shown in Figure 2.8.

2.3. Save created model for which click “Save Model” and input the file name “delivery”

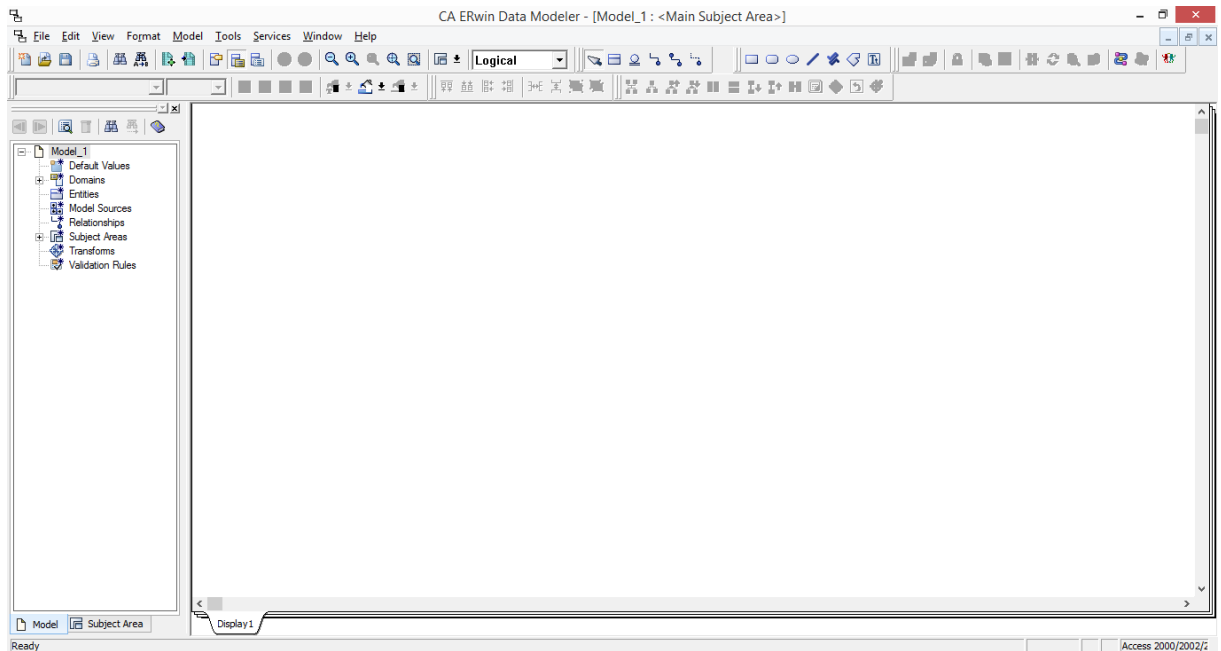


Figure 2.1

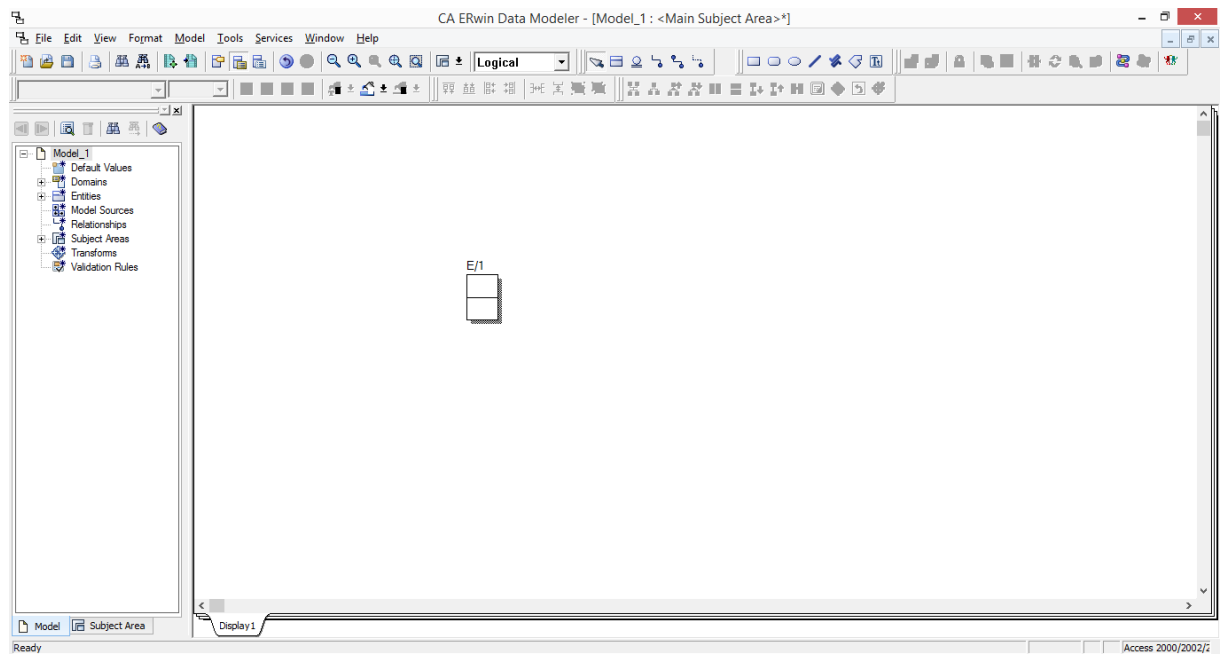


Figure 2.2

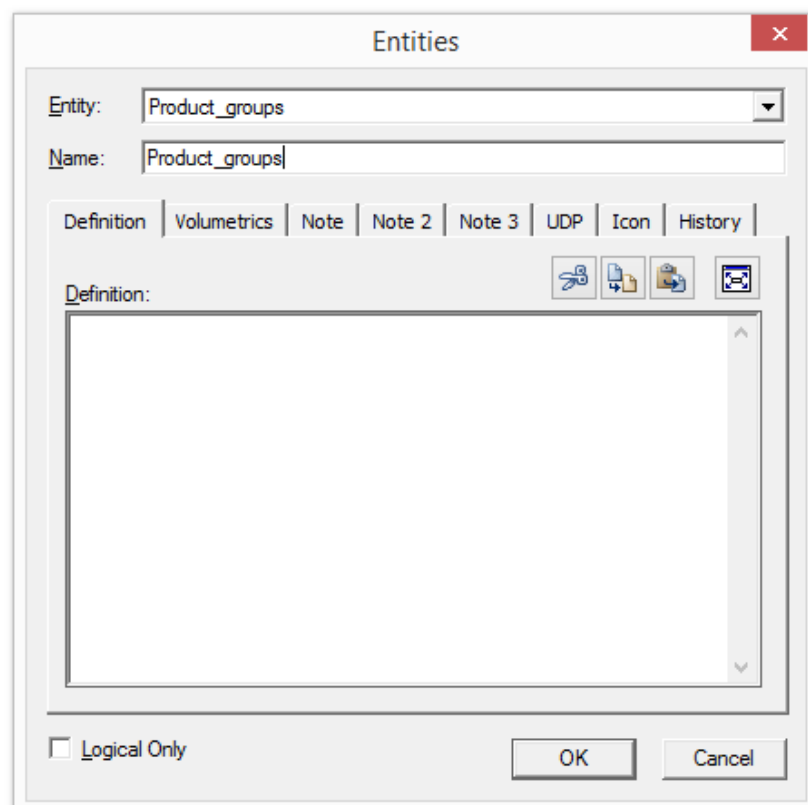


Figure 2.3

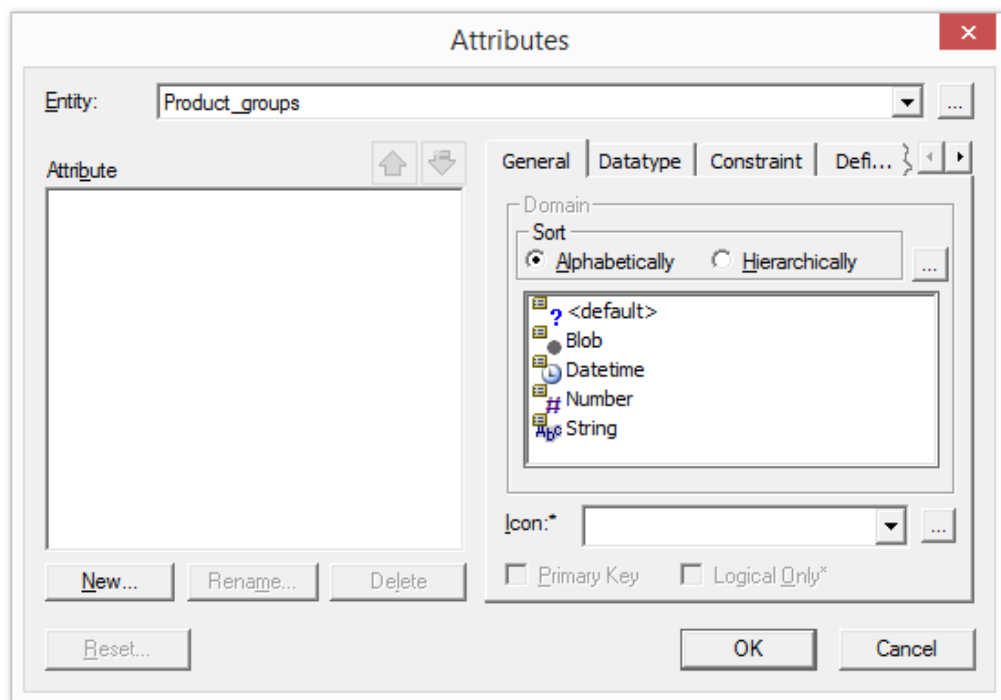


Figure 2.4

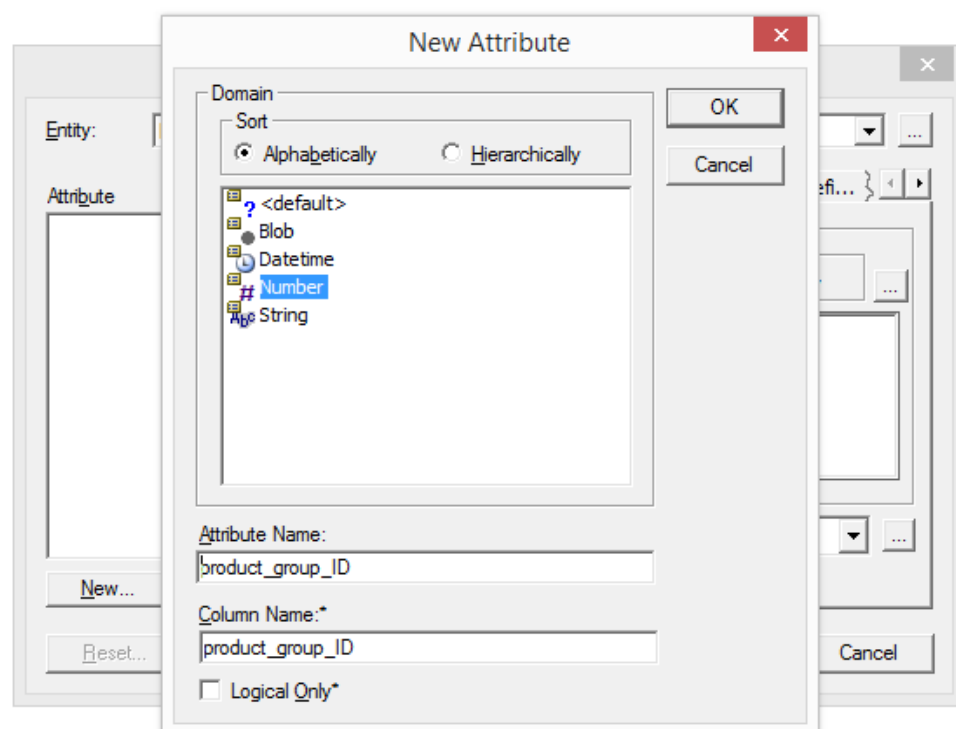


Figure 2.5

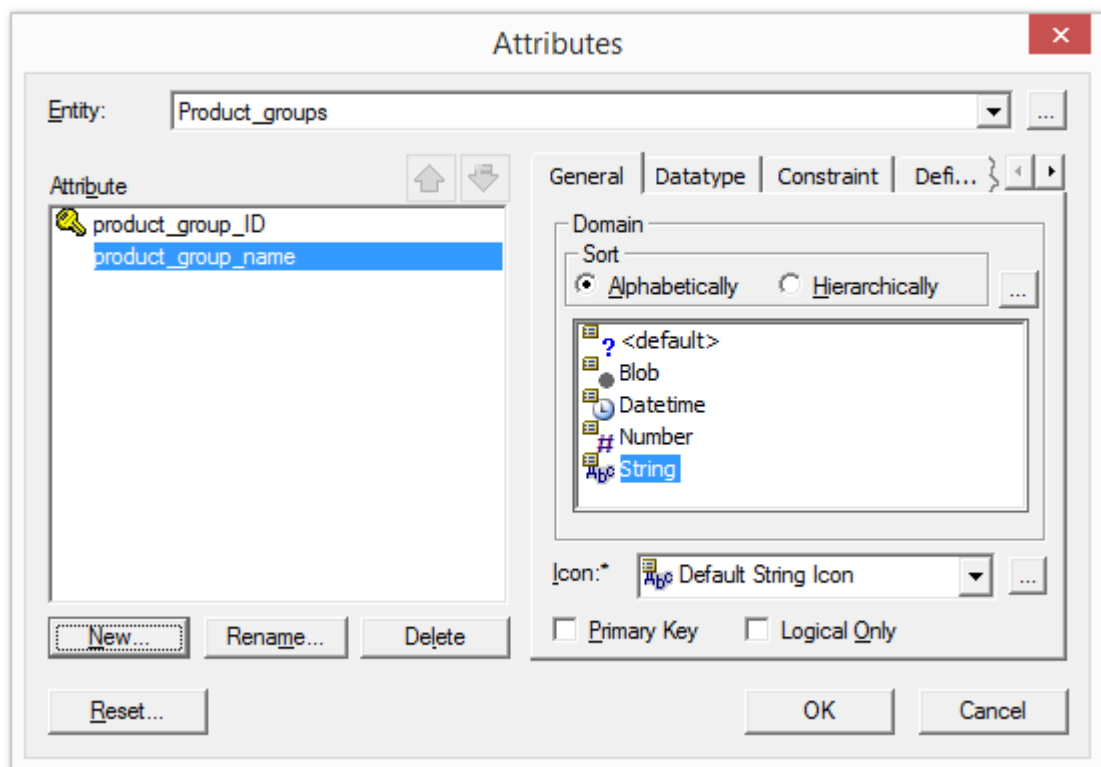


Figure 2.6

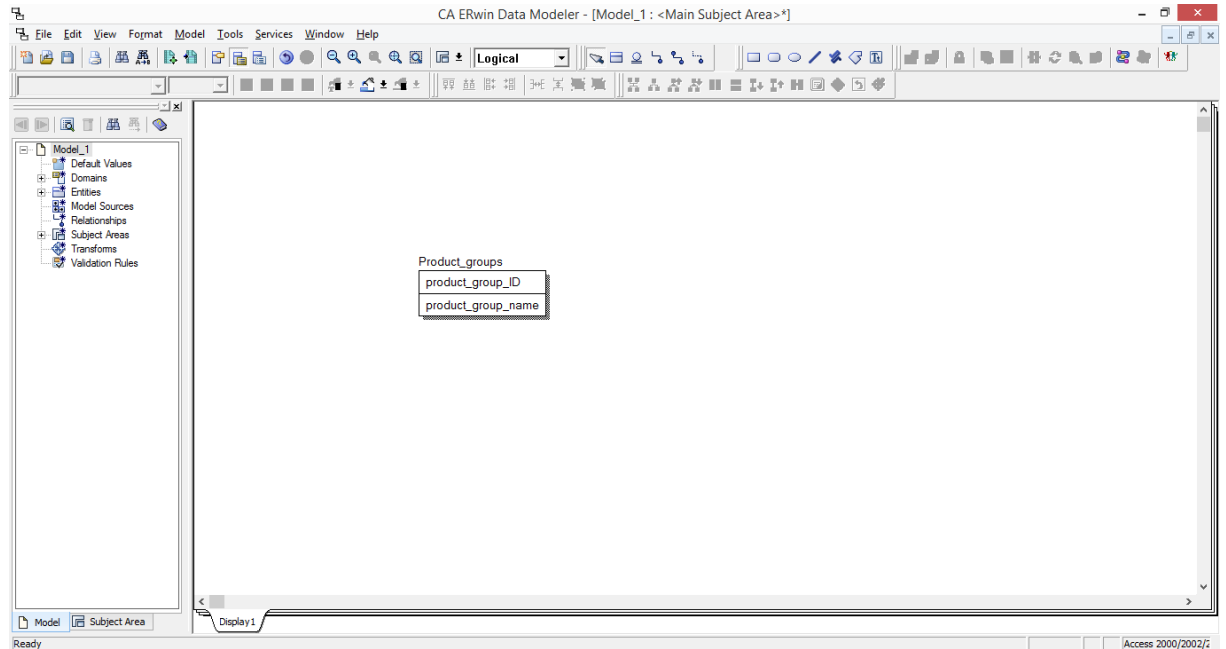


Figure 2.7

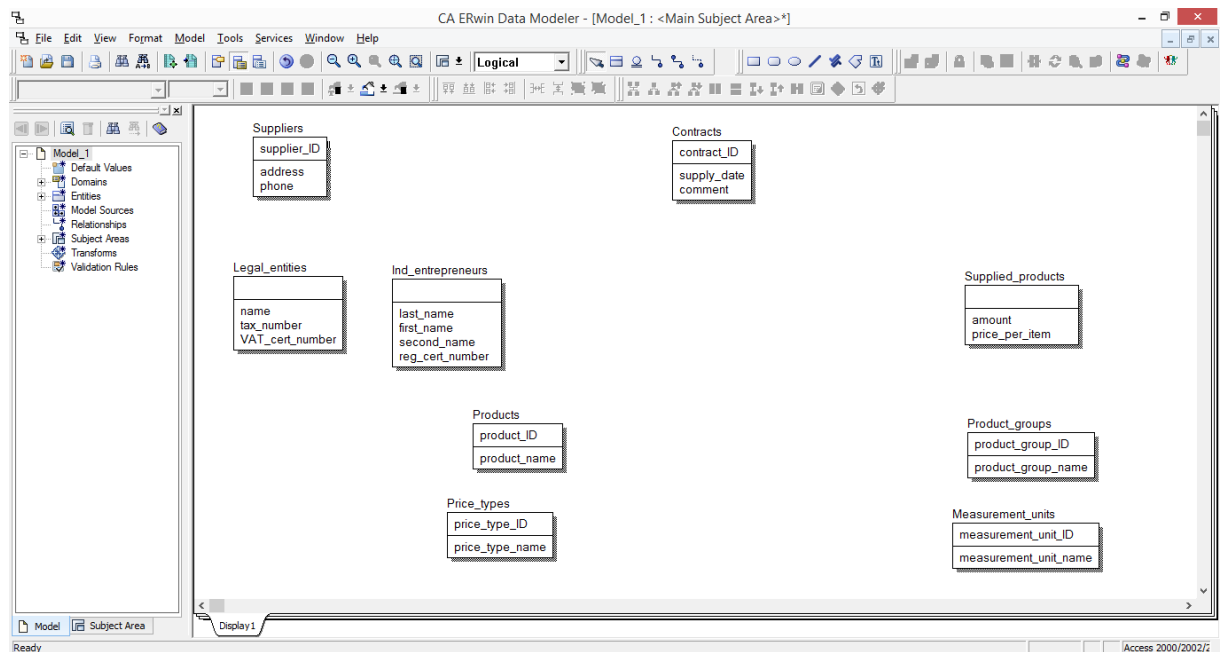


Figure 2.8

2.4. Create relationships

To describe the sequence of actions required to create relationships between entities we will consider the example of creation of the non-identifying relationship between entities “Products” and “Product_groups”:

- 1) Click “Non-identifying relationship”;
- 2) Click on the entity “Product_groups” and then on “Products” to create non-identifying relationship between these entities (Figure 2.9). Primary Key (PK) of the “Product_groups” entity is now Foreign Key (FK) of the “Products” entity. Since the FK is not a part of the PK, created relationship is non-identifying.

Notice.

In case if relationship on diagram is not shown as it should be, click on the relationship and move it into the required place.

Relationships between the created entities might be created in the same manner. The list of relationships between the entities and their types are shown in Table 2.2.

Table 2.2

Parent entity	Child entity	Relationship type
Product_groups	Products	Non-identifying
Measurement_units	Products	Non-identifying
Suppliers	Contracts	Non-identifying
Products	Supplied_products	Identifying
Contracts	Supplied_products	Identifying

As the result, diagram will be looking as it is shown in Figure 2.10.

2.5. Create categorical relationships for child entities in the inheritance hierarchy

The inheritance hierarchy (or categorical hierarchy) is a special type of entities relationships, which have common attributes. In this example it is “Suppliers” entity. Since suppliers can be both legal entities and individual entrepreneurs, storing data about these types using single entity is inconvenient. Hence, besides the “Suppliers” entity, “Legal_entities” and “Ind_entrepreneurs” were created as well.

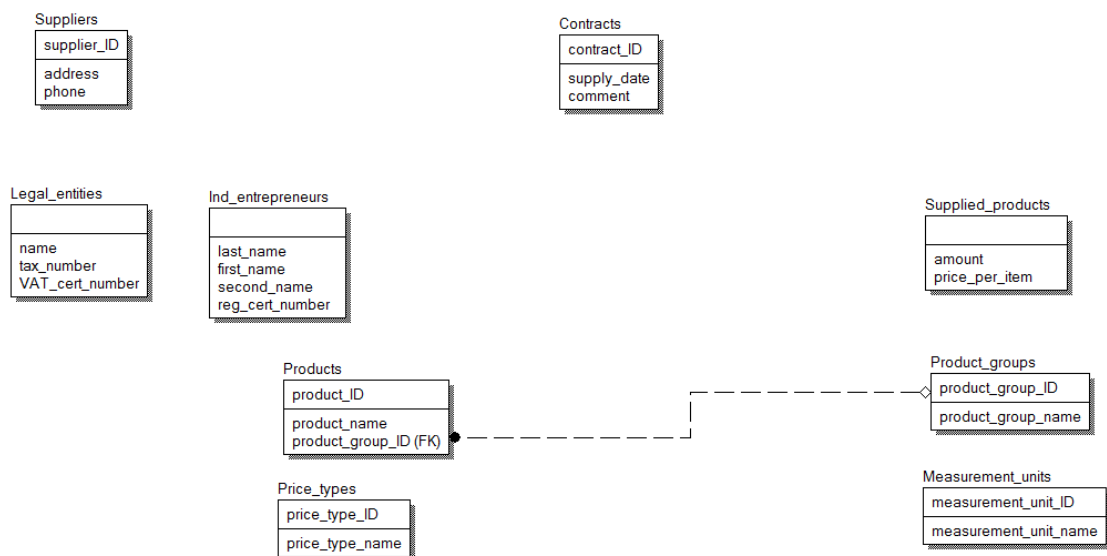


Figure 2.9

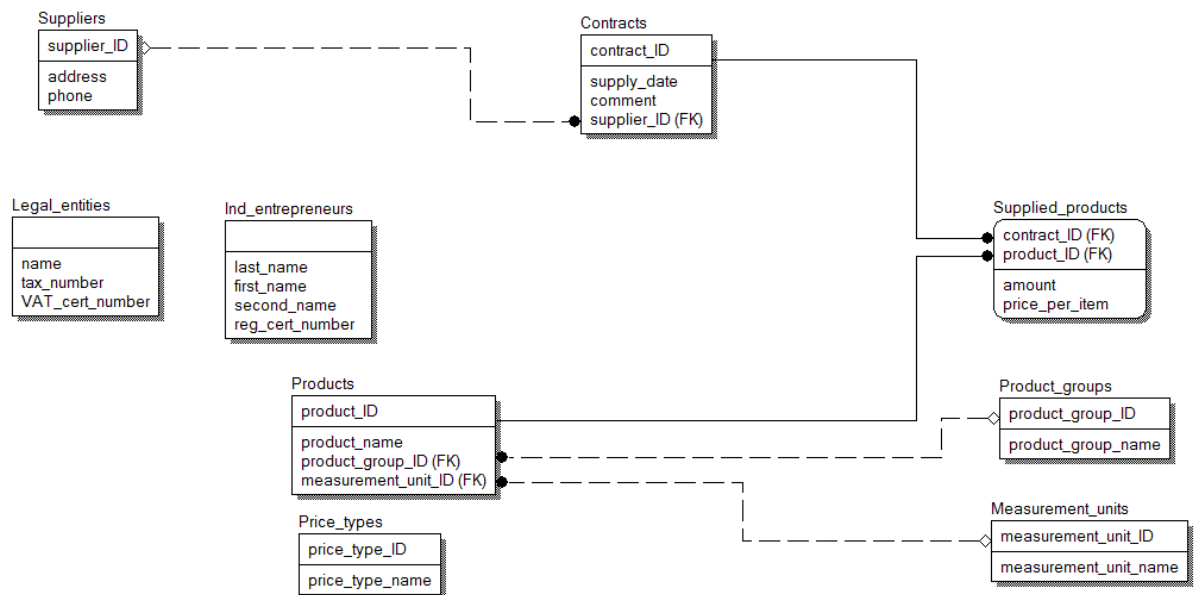


Figure 2.10

Steps required to create inheritance hierarchy that will include “Suppliers”, “Legal_entities”, and “Ind_entrepreneurs” entities:

- 1) Click on “Complete sub-category”;
- 2) Click on “Suppliers” entity and then on “Legal_entities” to create categorical relationship (Figure 2.11). Thus, PK of “Suppliers” entity will migrate and become also a PK of “Legal_entities” entity;
- 3) Click on “Complete sub-category” once again;
- 4) Click on categorical relationship and then on “Ind_entrepreneurs” entity to create categorical relationship (Figure 2.12). Thus, PK of “Suppliers” entity will become a PK of “Ind_entrepreneurs” as well.

2.6. Create “many-to-many” relationship between entities

“Many-to-many” relationship will be used to create entity that provides storage of the market state information (price offers of suppliers). Creation of “many-to-many” relationship between “Products”, “Suppliers”, and “Price_types” entities includes following steps:

- 1) Click on “Many-to-many relationship”;

- 2) First click on “Suppliers” and then on “Products” to create relationship (Figure 2.13);
- 3) Right click on the created relationship, select “Create Association Entity”, and click “Next”;
- 4) Type “Market_prices” into the “Entity Name” field in order to create new entity “Market_prices” that provides “many-to-many” relationship between “Suppliers” and “Products” entities (Figure 2.14);
- 5) Existence of “many-to-many” relationship shows that each supplier can offer various products on market and vice-versa each product might be offered by various suppliers. Since this relationship does not provide information about price, it is necessary to right click on “Market_prices” entity, select “Attributes”, and create another one attribute “price” of type “Number”.

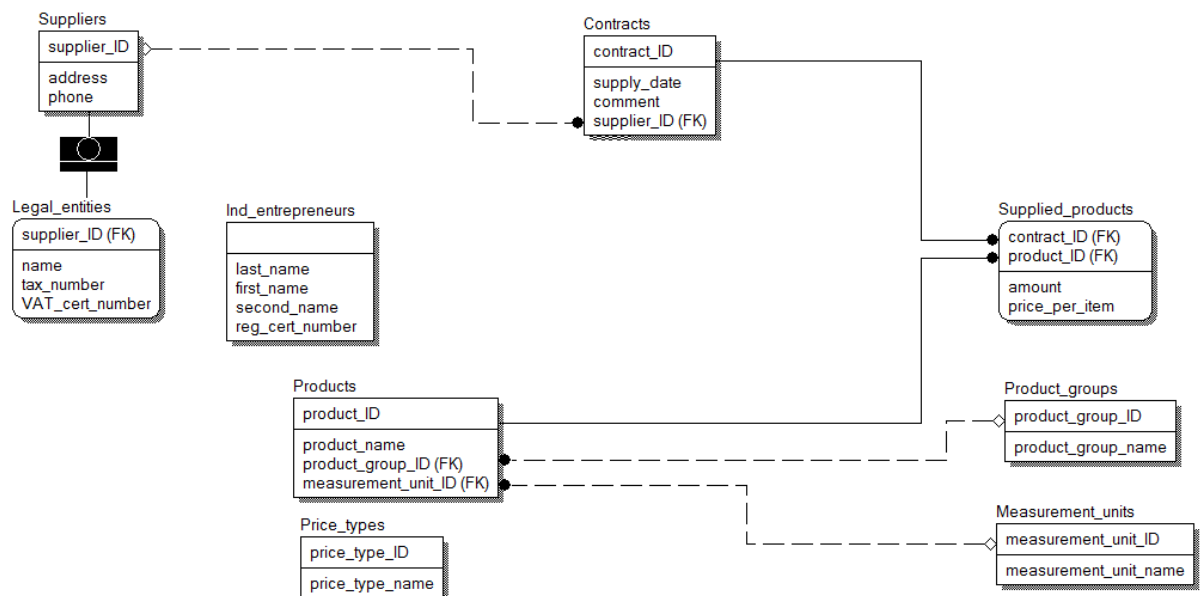


Figure 2.11

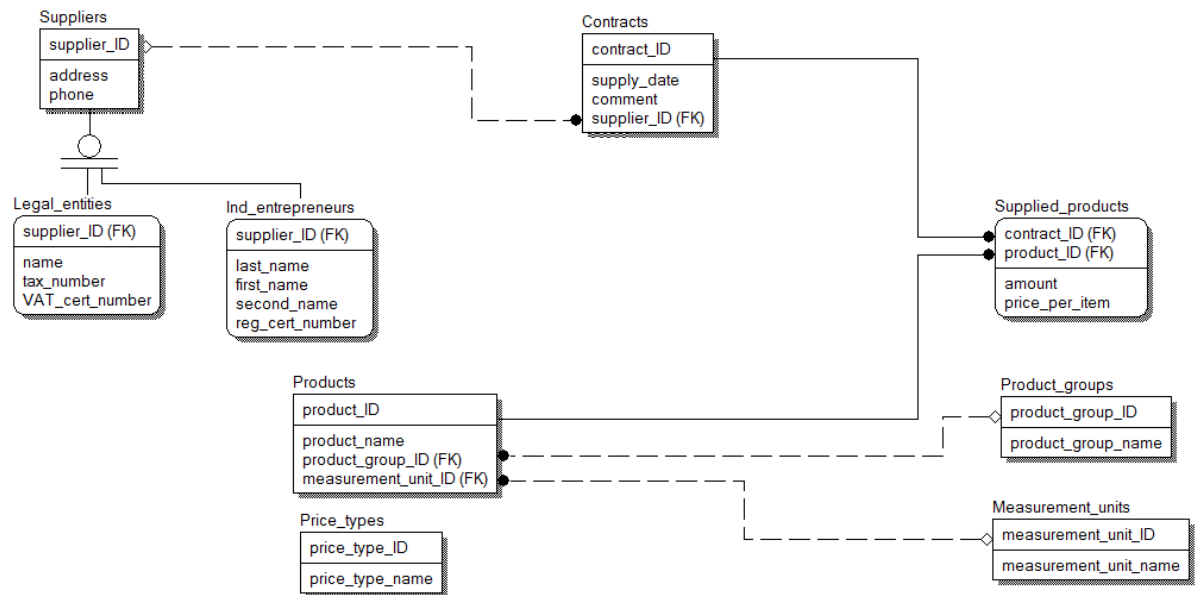


Figure 2.12

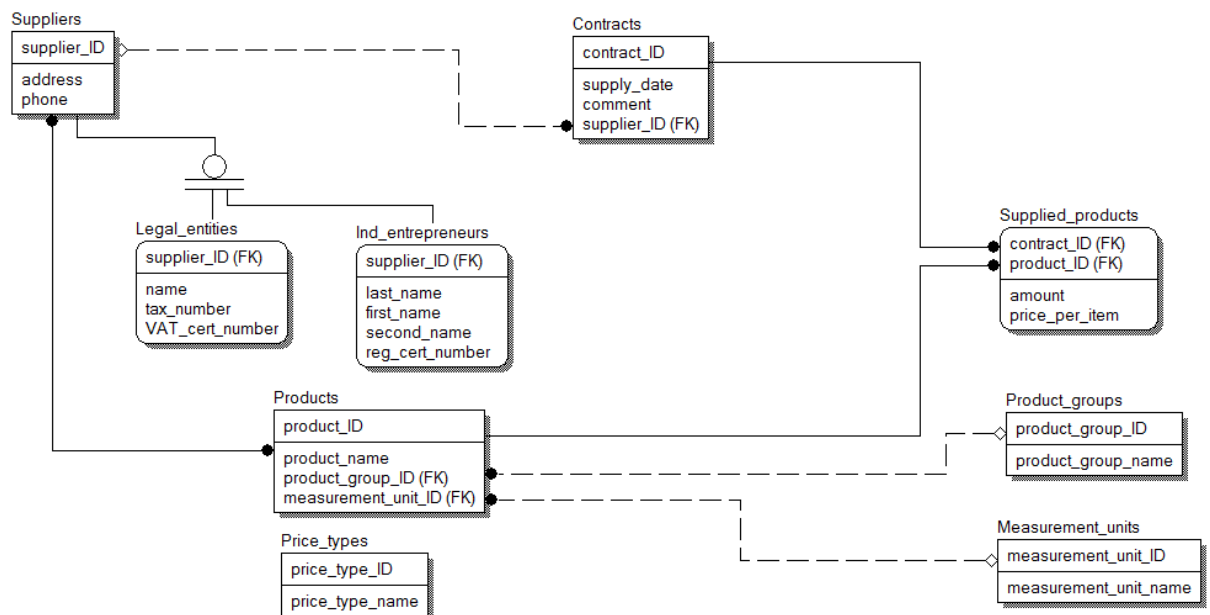


Figure 2.13

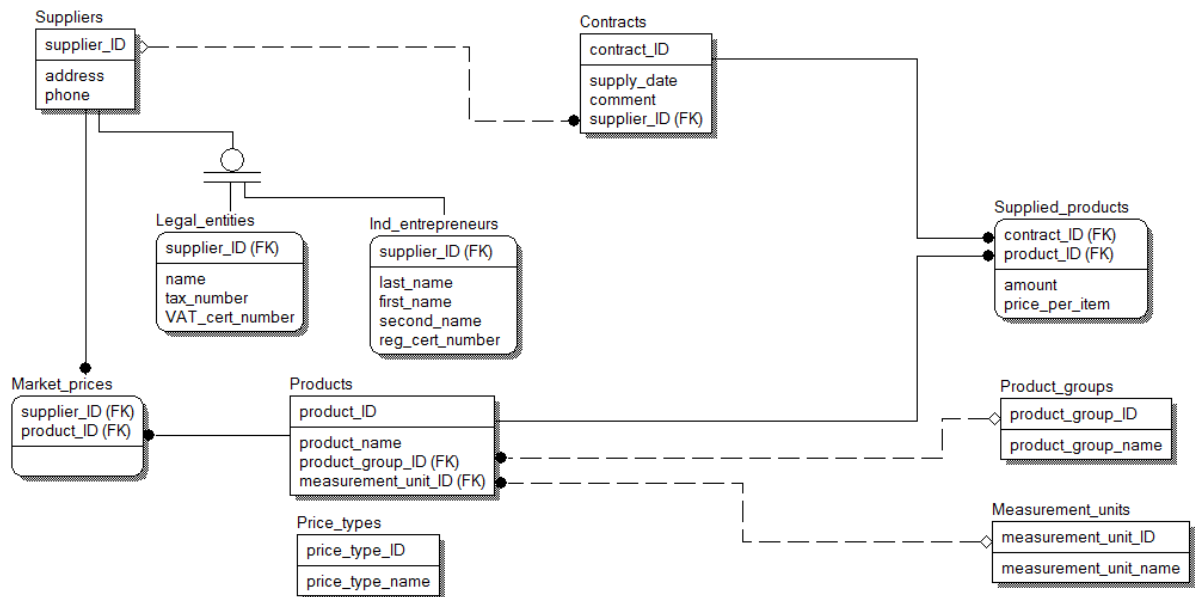


Figure 2.14

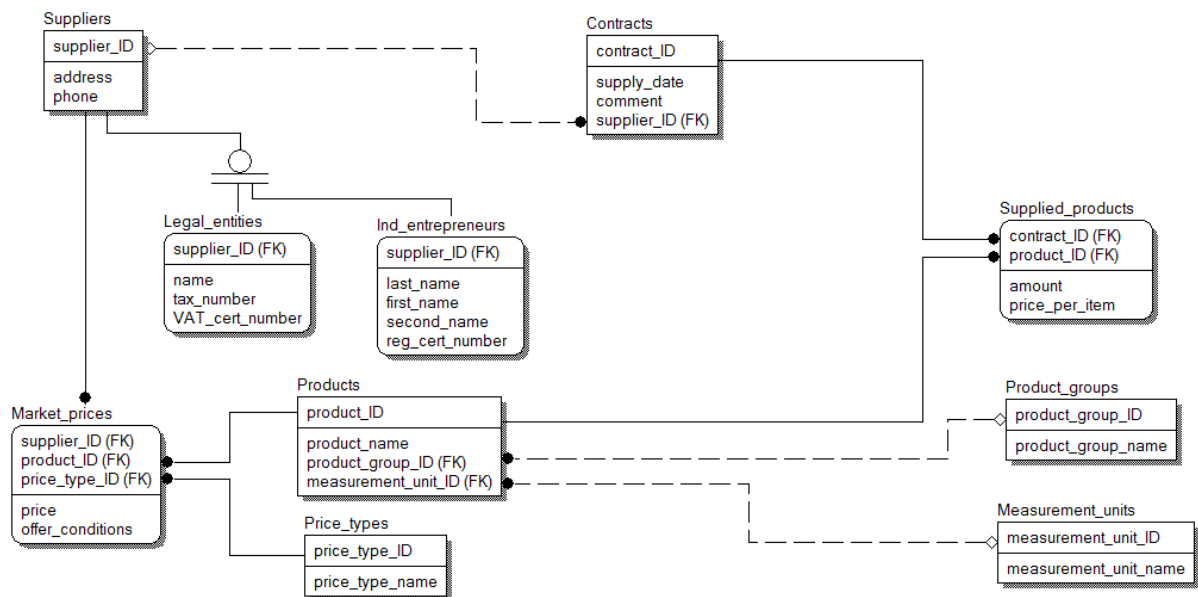


Figure 2.15

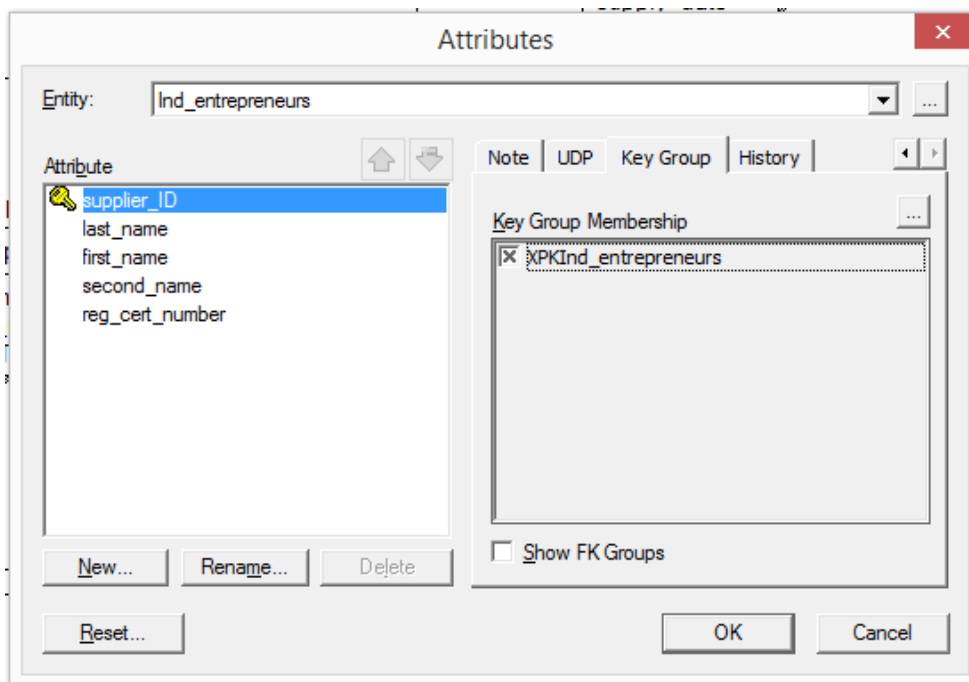


Figure 2.16

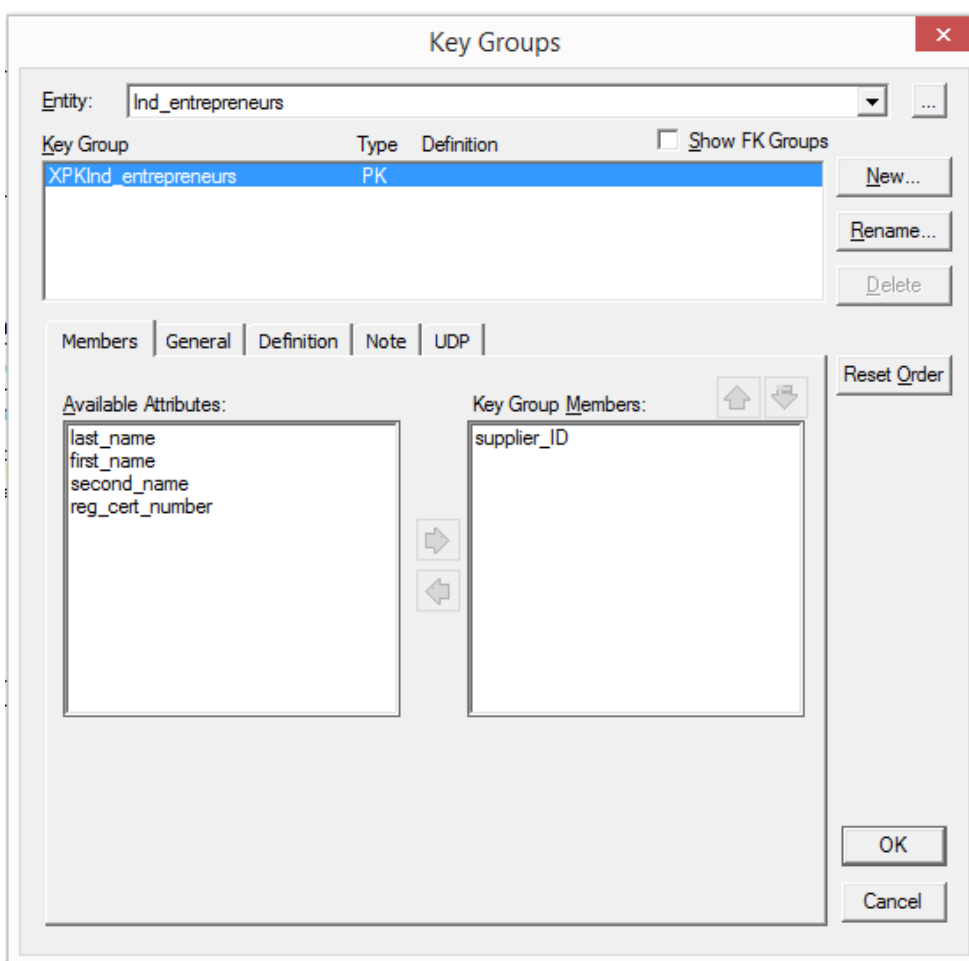
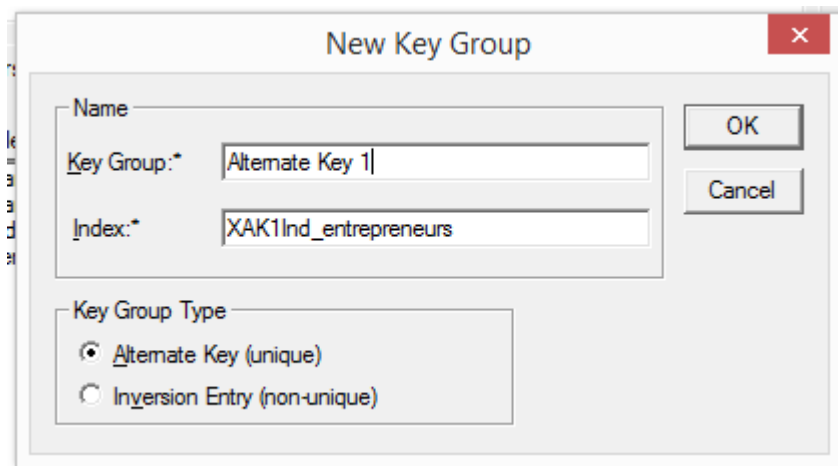


Figure 2.17

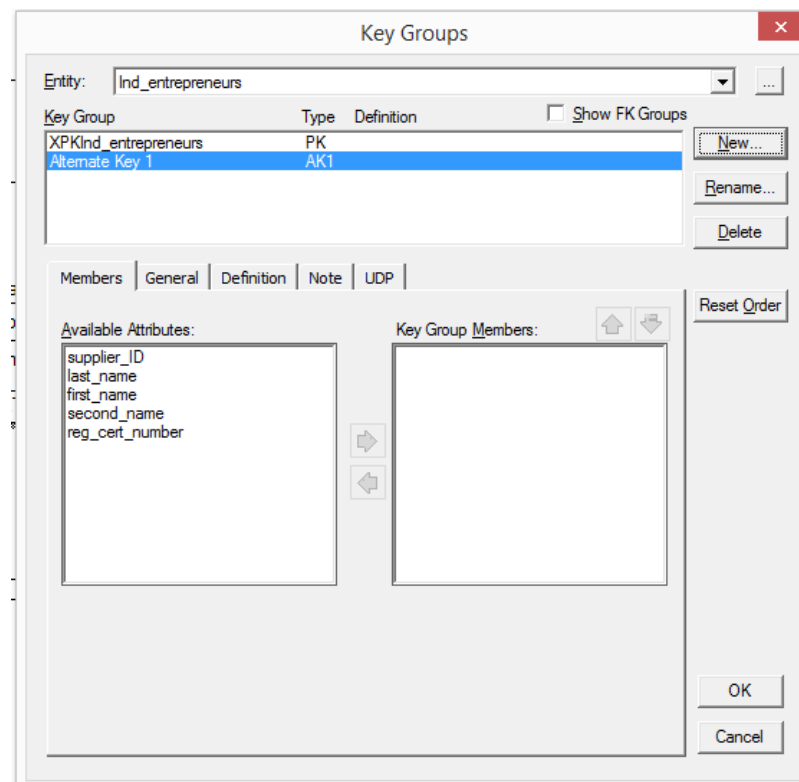


The 'New Key Group' dialog box is used to create a new key group. It contains the following fields and options:

- Name:**
 - Key Group:*** Alternate Key 1
 - Index:*** XAK1Ind_entrepreneurs
- Key Group Type:**
 - ☒ Alternate Key (unique)
 - ☐ Inversion Entry (non-unique)

Buttons: OK, Cancel

Figure 2.18



The 'Key Groups' dialog box is used to manage key groups for a specific entity. It contains the following elements:

- Entity:** Ind_entrepreneurs
- Table:**

Key Group	Type	Definition
XPKInd_entrepreneurs	PK	
Alternate Key 1	AK1	
- Buttons:** New..., Rename..., Delete
- Checkboxes:** ☐ Show FK Groups
- Tabs:** Members, General, Definition, Note, UDP
- Members Section:**
 - Available Attributes:** supplier_ID, last_name, first_name, second_name, reg_cert_number
 - Key Group Members:** (Empty list)
 - Buttons:** Add (right arrow), Remove (left arrow)
- Buttons:** Reset Order, OK, Cancel

Figure 2.19

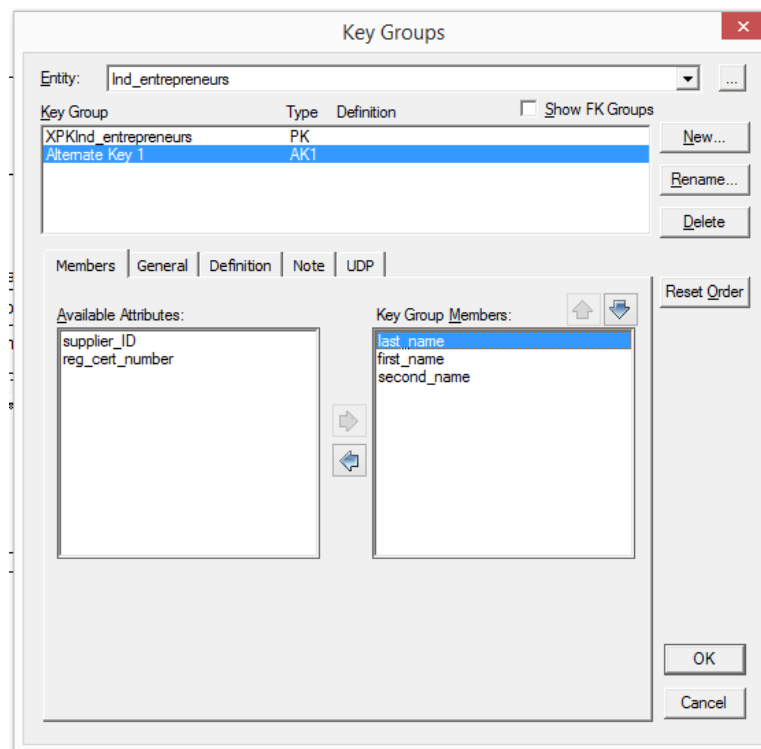


Figure 2.20

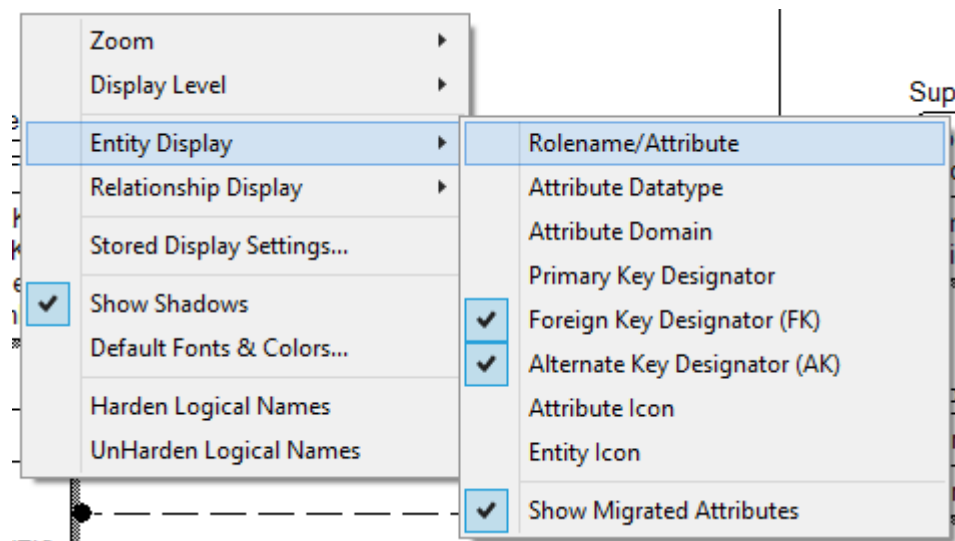


Figure 2.21

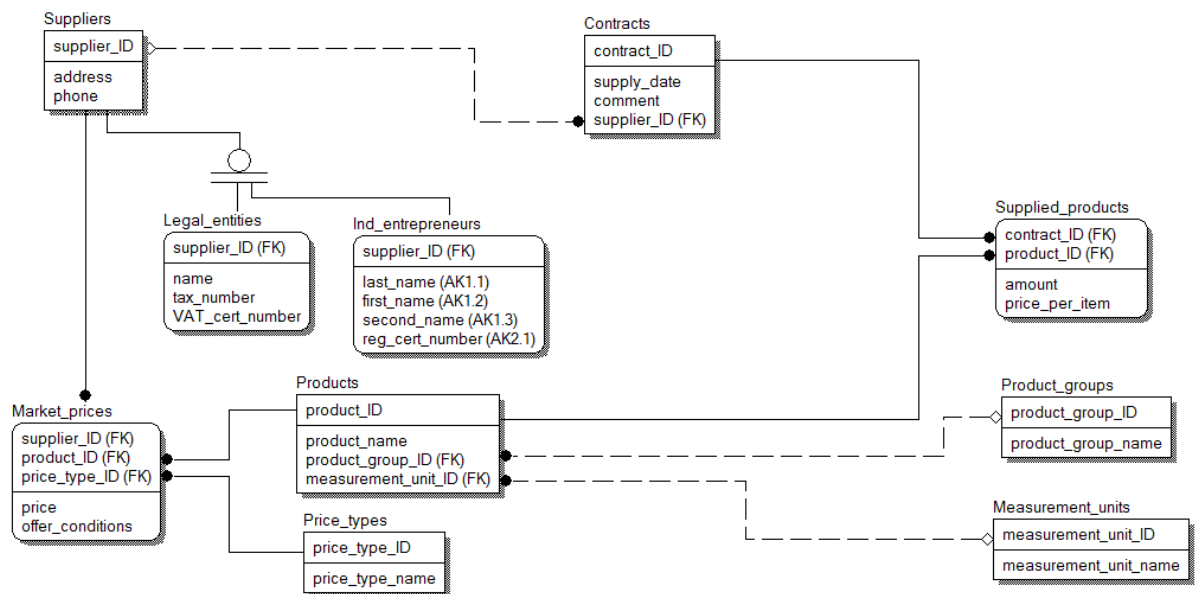


Figure 2.22

6) However, each supplier can offer certain product with single price. In order to provide storing information of various product prices offered by a single supplier it is necessary to click “Identifying relationship”, and then click on “Price_types” entity, and then on “Market_prices” to create relationship. PK of “Price_types” will become a part of “Market_prices” PK (Figure 2.15).

7) In order to provide customers with information about the offer conditions, it is necessary to add another attribute “offer_conditions” of type “String” (Figure 2.15).

2.7. Create alternate keys

It is common when during data modeling it is required to define another one or several alternate keys besides the PK. As usual, alternate keys are potential keys that were not selected as primary. Having alternate keys allows to control unique attributes that does not belong to the PK. Let’s assume that individual entrepreneurs are required to be stored with unique last name, first name, and second name, as well as with unique registration

certificate number. It requires to create two alternate keys. The sequence of the required steps is following:

- 1) Right click on “Ind_entrepreneurs” and select “Attributes...”;
- 2) Open “Key Group” tab (Figure 2.16) and click “...”. The “Key Groups” window with the data of keys of this entity will be shown (Figure 2.17);
- 3) Click “New” to create new alternate keys using “New Key Group” window (Figure 2.18), then click “Ok” (Figure 2.19);
- 4) To create the list of key attributes move “last_name”, “first_name”, and “second_name” from the “Available Attributes” into “Key Group Members” (Figure 2.20);
- 5) Now create the second alternate key in the same manner including only “reg_cert_number” into key group members;
- 6) After closing “Key Groups” window, “Key Group” tab will contain the modified list of the current entity keys;
- 7) Close “Attributes” window by clicking “Ok”;
- 8) In order to display information about the alternate keys, right click on any empty space of the model, select “Entity Display”, “Alternate Key Designator (AK)” (Figure 2.21). As the result, alternate keys among the entity attributes will be shown (Figure 2.22).

2.8. Modify relationship properties

Properties modification of previously created relationships is a common situation of the data modeling process. It depends on default properties of relationships that do not correspond to the requirements of the designed database. Let's consider the example of properties modification using the relationship between “Products” and “Product_groups” as the example. This relationship shows that each product belongs to the certain product group. However, some product might not belong to any product group. This is possible because FK “Product_group_ID” in “Products” entity might be empty (Null). This property was set by default but it is not compliant with the

requirements declared as the result of subject area analysis and, therefore, it should be modified. There is sequence of steps used to modify relationship properties:

- 1) Double click on relationship to open “Relationships” window where relationship properties are displayed (Figure 2.23);
- 2) Select “No Nulls” in “Nulls” section;
- 3) Click “Ok” to close “Relationships” window;
- 4) Analyze how relationship representation was changed on diagram;
- 5) Change the same properties of other non-identifying relationships;
- 6) Save all changes made in the model.

The screenshot shows the 'Relationships' dialog box with the following details:

- Relationship:** R/2 (Product_groups to Products)
- Name:** R/2
- Buttons:** New..., Delete
- Tabs:** General, Definition, Rolename, RI Actions, UDP
- Verb Phrase:**
 - Parent-to-Child: [Empty field]
 - Child-to-Parent: [Empty field]
- Relationship Cardinality:**
 - Summary: Zero-or-One-to-Zero-One-or-More
 - Cardinality:**
 - ☒ Zero, One or More
 - ☐ One or More (P)
 - ☐ Zero or One (Z)
 - ☐ Exactly: [Empty field]
 - Relationship Type:**
 - ☐ Identifying
 - ☒ Non-Identifying
 - Nulls:**
 - ☒ Nulls Allowed
 - ☐ No Nulls
- Buttons:** Logical Only (checkbox), Reset Cardinality
- Buttons:** OK, Cancel

Figure 2.23

2.9. Modify categorical relationships in the inheritance hierarchy

Categorical relationships might require modification as well. There are several examples related to modification of categorical relationship properties or their transformation.

Attention! Following changes do not need to be saved!

Example 1. Change categorical hierarchy type

There are two types of categorical hierarchies – complete and incomplete. Categorical relationship implemented earlier is the example of the complete category. Following steps required to change category type:

- 1) Right click on the categorical relationship and select “Subtype Relationship...” which demonstrates relationship properties;
- 2) Turn “Subtype Type” into “Incomplete” mode;
- 3) Click “Ok” and check how diagram will change after category type was changed into incomplete;
- 4) Restore the complete category type.

Example 2. Replace inheritance hierarchy with identifying relationships

This replacement is intended to simplify model or improve it based on subjective design decisions. To replace inheritance hierarchy with identifying relationships it is required to perform following steps:

- 1) Click on categorical relationship and then click on “Supertype-Subtype Identity”;
- 2) In the opened wizard click “Next” twice and then “Finish”. The result of transformation is shown in figure 24. As it is shown, categorical relationship was replaced with two identifying relationships;
- 3) Close the model without saving changes.

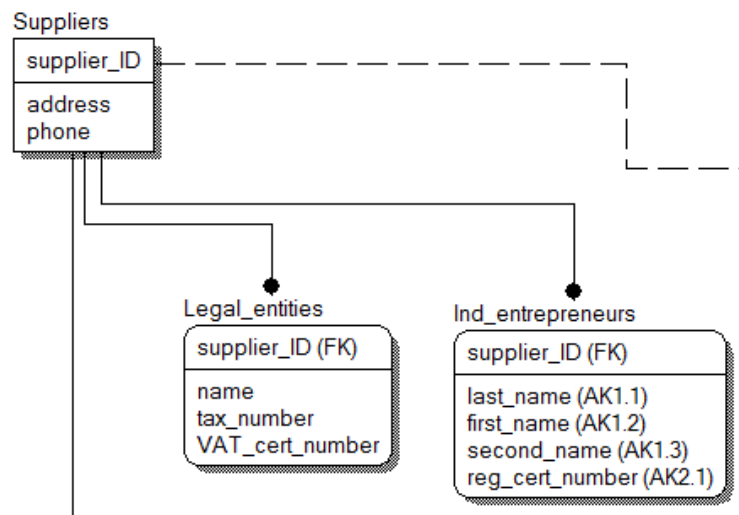


Figure 2.24

Example 3. Migrate PK and non-key attributes within inheritance hierarchy from child to parent

This replacement might be intended to simplify or improve model as well. This task requires following steps:

- 1) Open model;
- 2) Click on the categorical relationship and then on “Supertype-Subtype Rollup”;
- 3) Skip all steps in the wizard window. The result is shown in figure 25. As it is shown, only one entity “Suppliers” is left. The list of attributes of this entity includes attributes of “Legal_entities” and “Ind_entrepreneurs” entities. This structure has obvious shortcomings related to the normalization requirements;
- 4) Close the model without saving changes.

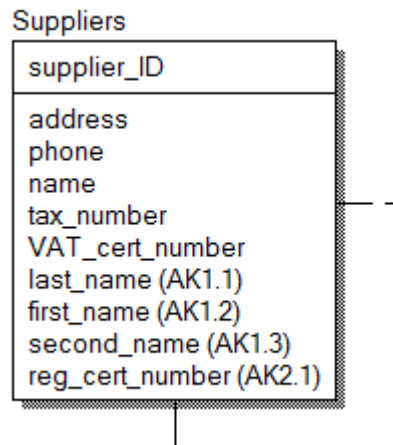


Figure 2.25

Example 4. Migrate PK and non-key attributes within inheritance hierarchy from parent to child

This replacement has shortcomings related to the normalization requirements as well. It requires following steps:

- 1) Open model;
- 2) Click on the categorical relationship and then on "Supertype-Subtype Rolldown";
- 3) Skip all steps in the wizard window. The result is shown in figure 26. As it is shown, only two entities "Legal_entities" and "Ind_entrepreneurs" are left. The list of attributes of these entities includes attributes of "Suppliers" entity;
- 4) Close the model without saving changes;
- 5) Open the model again.

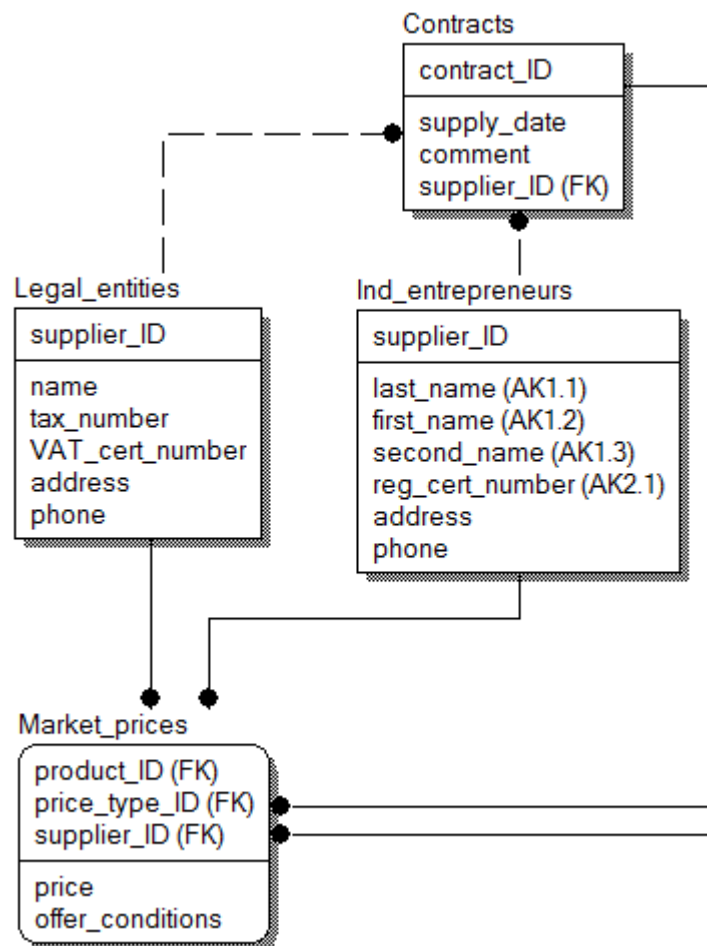


Figure 2.26

2.10. Transfer to the physical data model

Logical data model might be created without DBMS definition. However, the real database might be created with considering features of the concrete DBMS. In this work DBMS was defined when the model was created. Physical data model that considers features of the selected DBMS is created automatically while logical model is designed. To access physical data model the mode should be changed from “Logical” to “Physical”. As the result, the physical model will be shown (Figure 2.27).

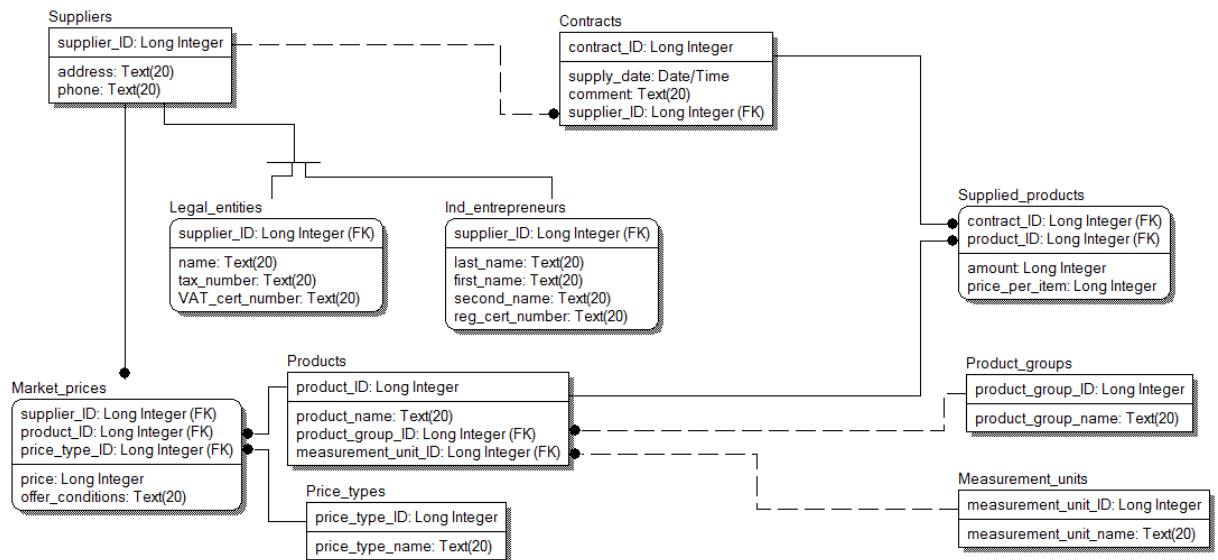


Figure 2.27

2.11. Questions

1. Briefly describe the main stages of performed work.
2. Depict models developed during the work (final view of the models).
3. Describe purpose of core elements of the ERWin interface, their features used to design data model.