

INTRODUCTION

Goal: State the laboratory practice problem. Study and analyze the subject area. Select main database objects based on analysis.

Simplified description of the subject area

Some enterprise purchases products from various suppliers. Suppliers can be both legal entities and individual entrepreneurs. Products purchase is performed as batches and processed in the form of supply contracts. Each supply contract has unique number and might be concluded with single supplier. Preliminary order, invoice or similar document is reason for supply. Each contract document for each product type contains product name, number of items, and cost (in UAH). Supplied products are brought to the warehouse for the purpose of further sales through various channels.

Subject area analysis allowed to identify and describe core business processes related to products supply. It is assumed that enterprise will have to store large enough amount of data related to products supply. Besides, product supply information should be organized in the way that staff and management could perform analytical processing of this information. Therefore, it is necessary to create database used to store and process information related to products supply. Business process analysis allowed to define following information that might be included in the database.

1. Product information

Includes product name, unique product identifier (ID, Stock Keeping Unit, etc.), product measurement unit (item, box, kg, etc.), and other. Products are grouped into various product groups (e.g. grocery, perfumery, household

chemicals, etc.). It is assumed that each product belongs to a single product group.

2. Supplier information

Includes information about business entities on the market that offer products interested for the current enterprise. Suppliers can be both legal entities and individual entrepreneurs. Supplier information includes name, individual tax number, number of the VAT payer certificate (for legal entities); last name, first name, and second name, number of the registration certificate (for individual entrepreneurs; address, phone number (for both types of business entity), and other.

3. Price information

Same products might be offered by various suppliers. Moreover, each supplier can offer the same products with various prices (retail, wholesale, etc.) depending on the purchase amount, contract conditions, etc.

4. Supply information

Each supply is based on contract concluded between the supplier and enterprise. For each supply the following information is known: supplier, supply date, total cost, and supplied products information. Supplied products information includes product name, number of items, and price per item. The costs of supplied products might be different than standard prices offered by supplier (special discounts might be applied for a particular customer, the price for the certain product types might be assigned individually, etc.).

Thus, there might be defined following database tables (entities in data model) based on the analysis above. Each table contains fields (attributes) that describe stored information:

1. Product groups

1.1. Product group ID

1.2. Product group Name

2. Product measurement units

2.1. Product measurement unit ID

2.2. Product measurement unit name

3. Products

3.1. Product ID

3.2. Product name

3.3. Product group

3.4. Product measurement unit

4. Product price types

4.1. Product price type

4.2. Product price type name

5. Suppliers

5.1. Supplier ID

5.2. Supplier name (for legal entity)

5.3. Individual tax number (for legal entity)

5.4. Number of the VAT payer certificate (for legal entity)

5.5. Last name, first name, second name (for individual entrepreneur)

5.6. Number of the registration certificate (for individual entrepreneur)

5.7. Address

5.8. Phone number

6. Market prices

6.1. Product

6.2. Supplier

6.3. Price type

6.4. Price value

6.5. Price offer condition

7. Supply contracts

- 7.1. Contract ID
- 7.2. Supply date
- 7.3. Supplier
- 7.4. Comment (some additional information)
- 8. Supplied products
 - 8.1. Contract ID
 - 8.2. Products
 - 8.3. Supplied amount
 - 8.4. Price per item

Information above might be used to develop IDEF1X data model.

Part 1

Goal: Design IDEF1X models.

Steps:

1. Create logical 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 (fig. 1).

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 (fig. 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” (fig. 3);
- 4) Create entity attributes for which right click on it and select “Attributes...”, in the appeared “Attributes” window (fig. 4) click “New” button and in the appeared “New Attribute” window input “product_group_ID”, select domain type “Number” (fig. 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” (fig. 6);
- 6) Click “Ok” in “Attributes” window (fig. 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.

Table 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
	second_name		String
	reg_cert_number		String

As the result, logical model that contains created entities and attributes should looks as it is shown in fig. 8.

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

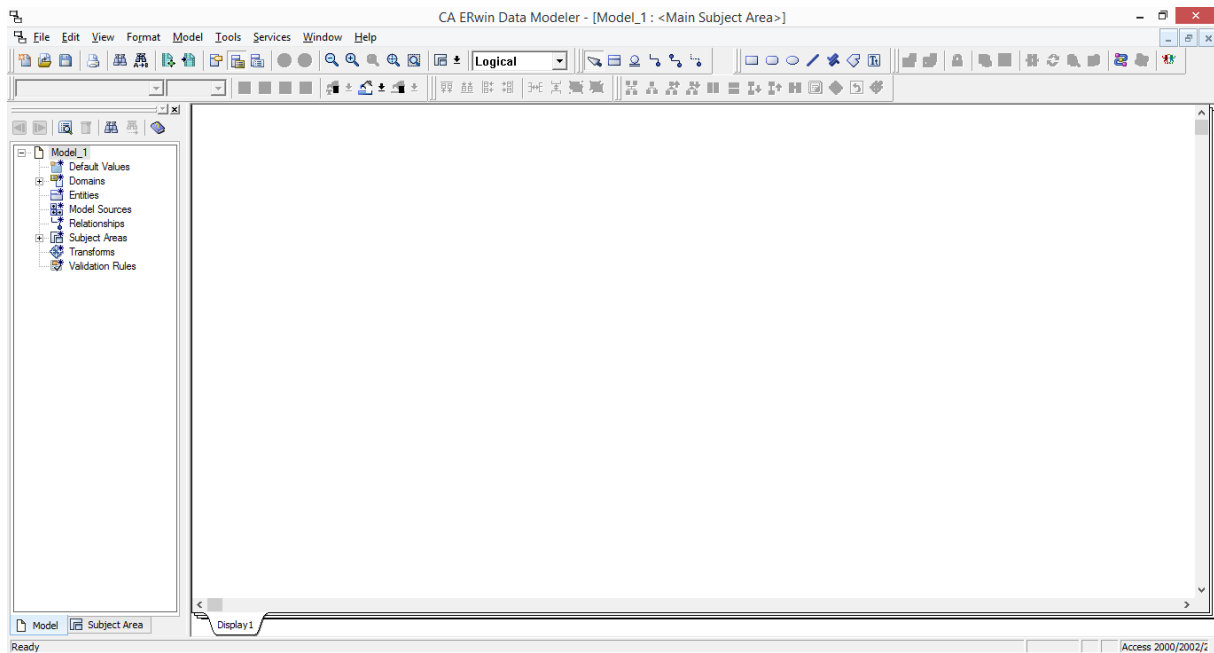


Fig. 1

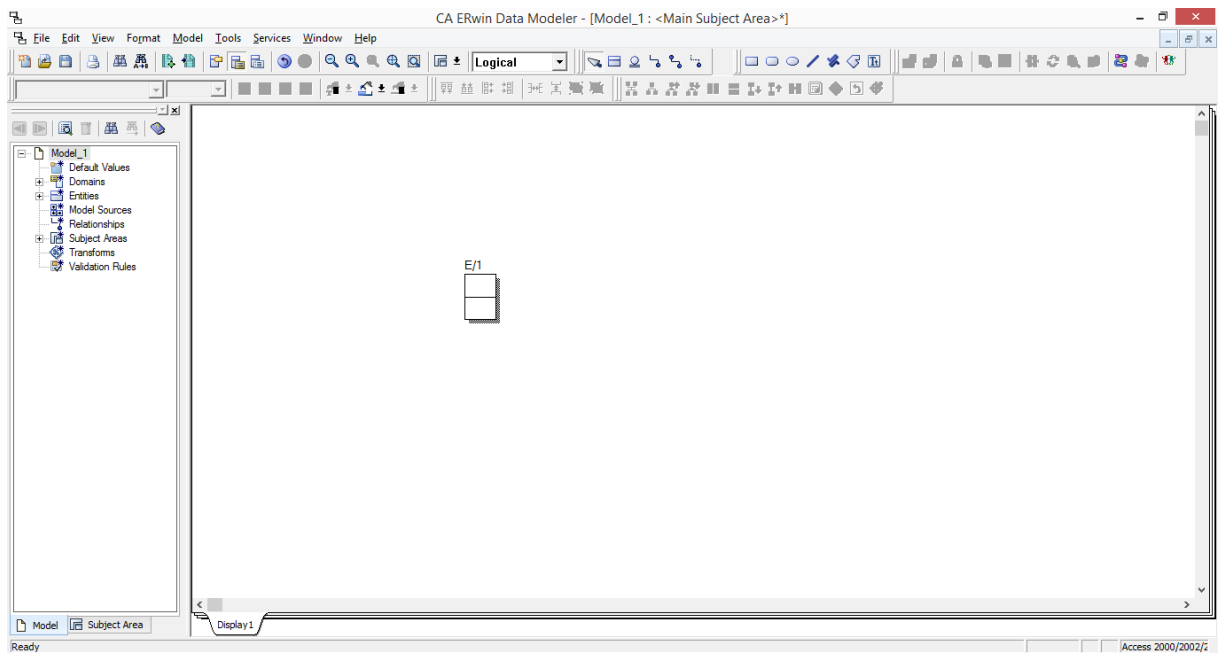


Fig. 2

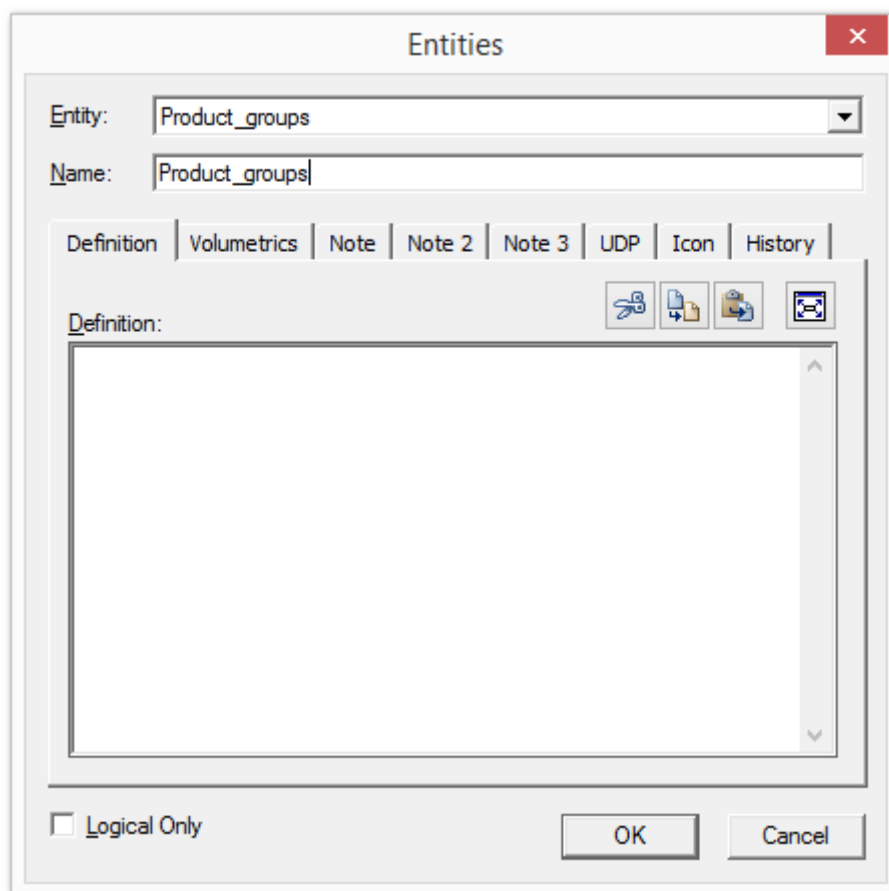


Fig. 3

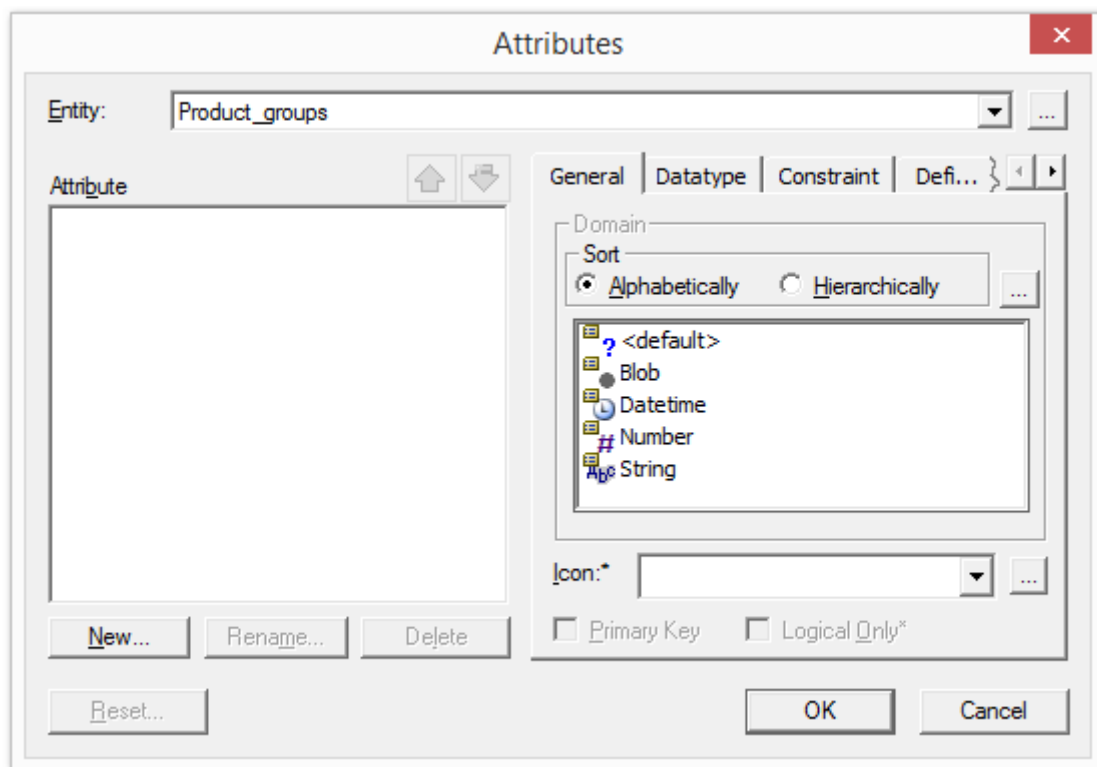


Fig. 4

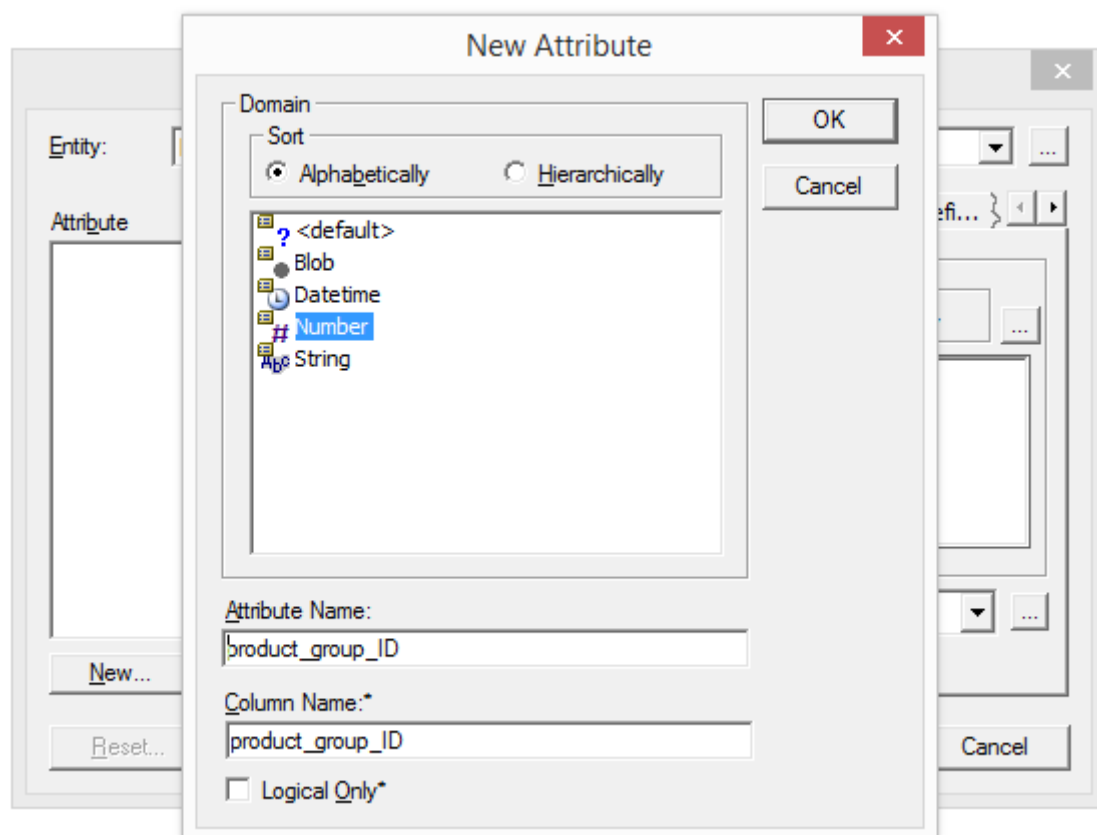


Fig. 5

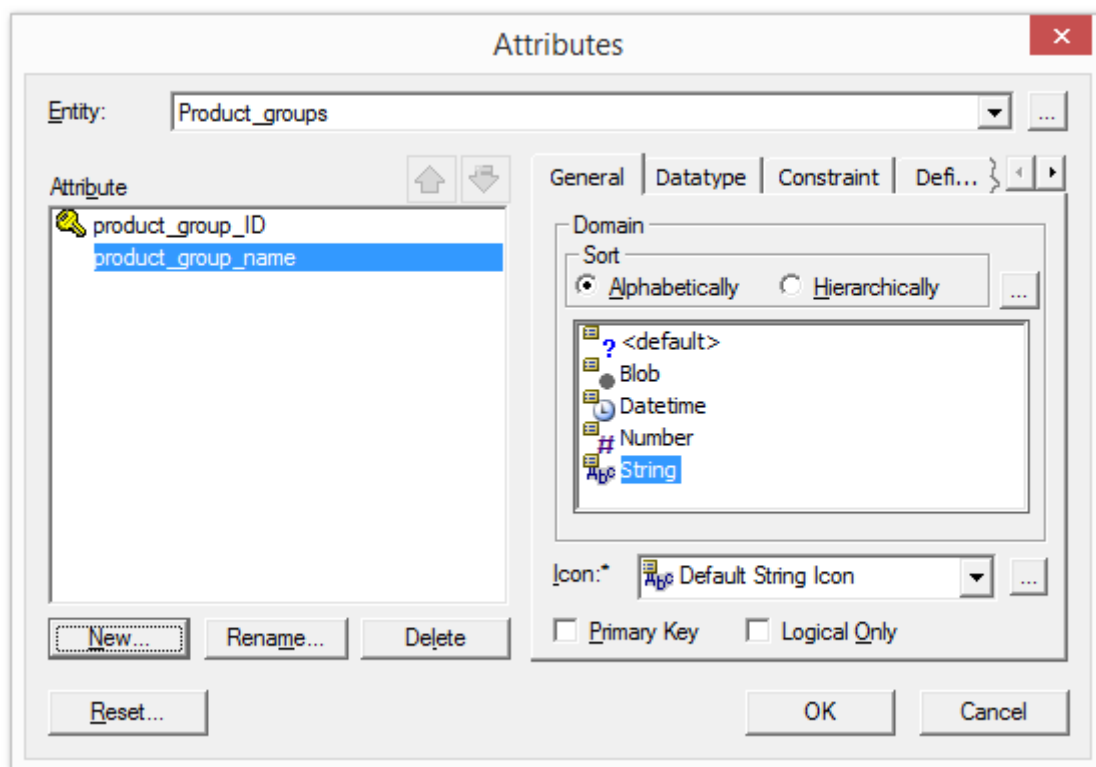


Fig. 6

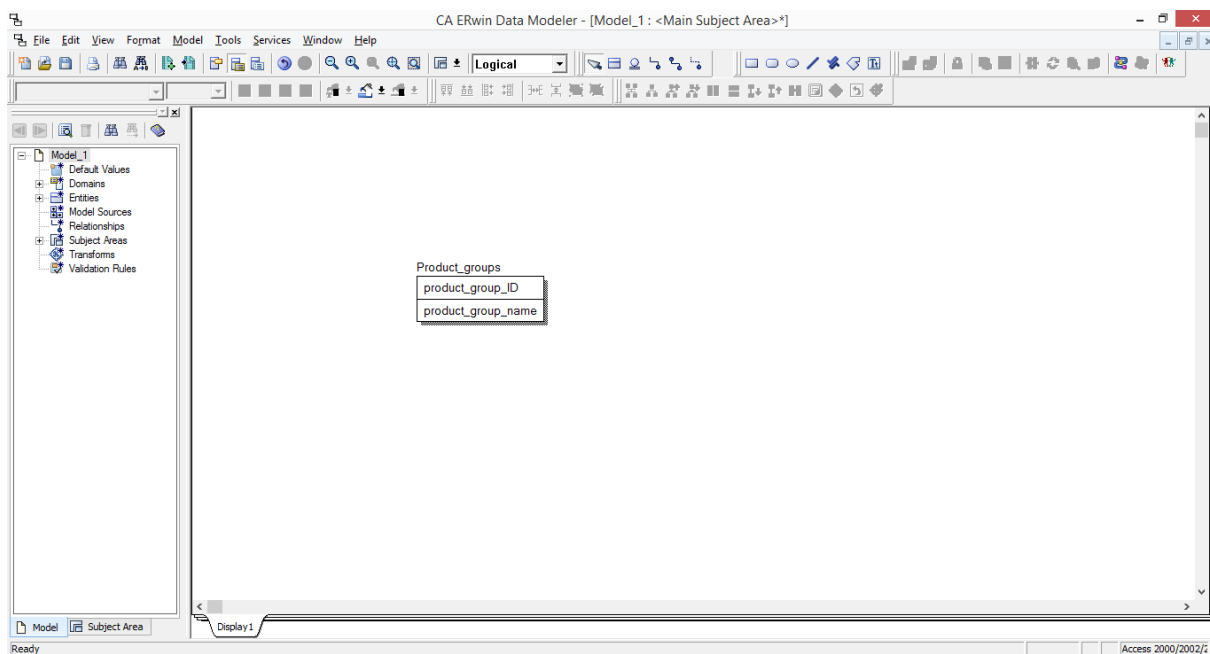


Fig. 7

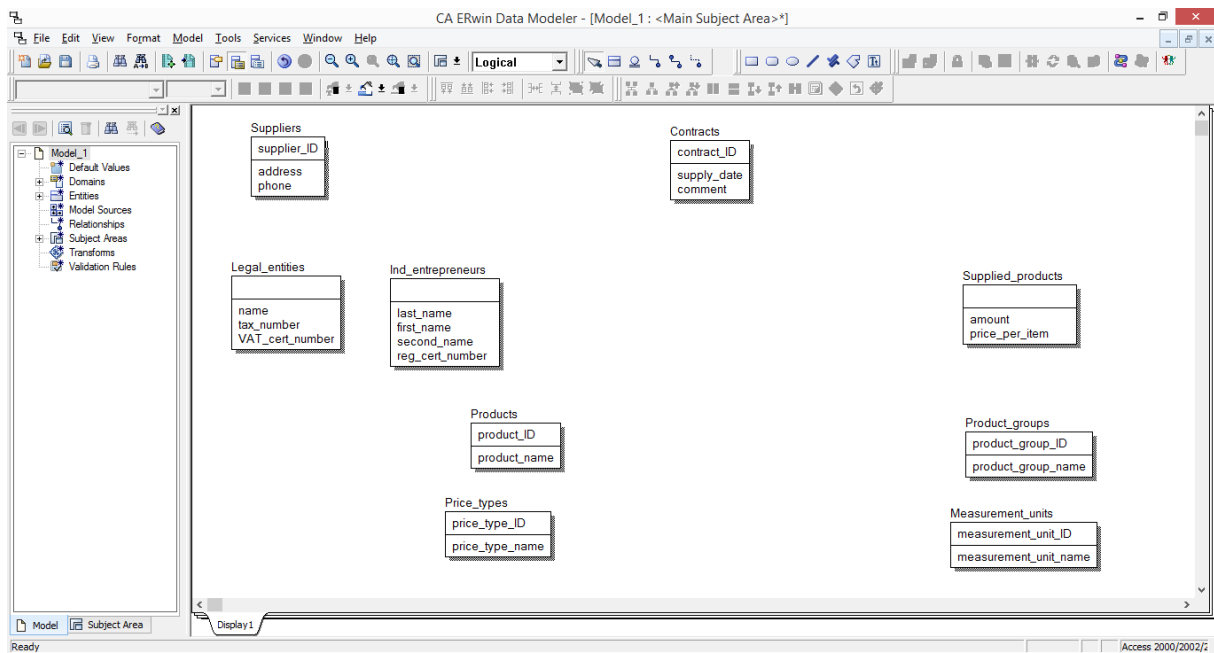


Fig. 8

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 (fig. 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.

Table 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 fig. 10.

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.

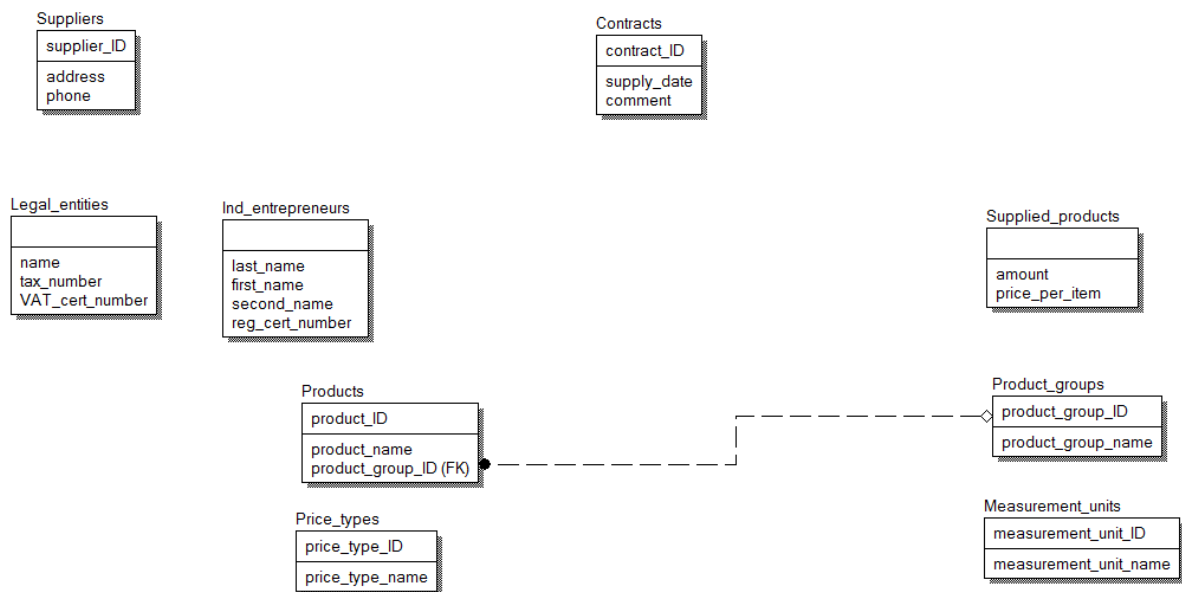


Fig. 9

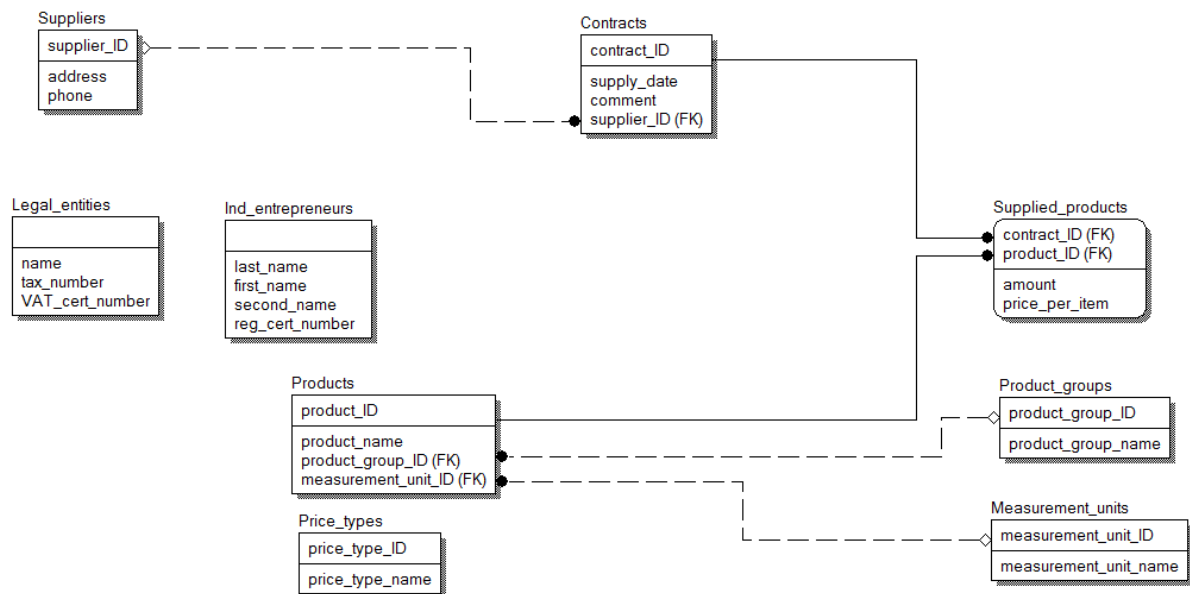


Fig. 10

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 (fig. 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 the on “Ind_entrepreneurs” entity” to create categorical relationship (fig. 12). Thus, PK of “Suppliers” entity will become a PK of “Ind_entrepreneurs” as well.

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 (fig. 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 (fig. 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”.

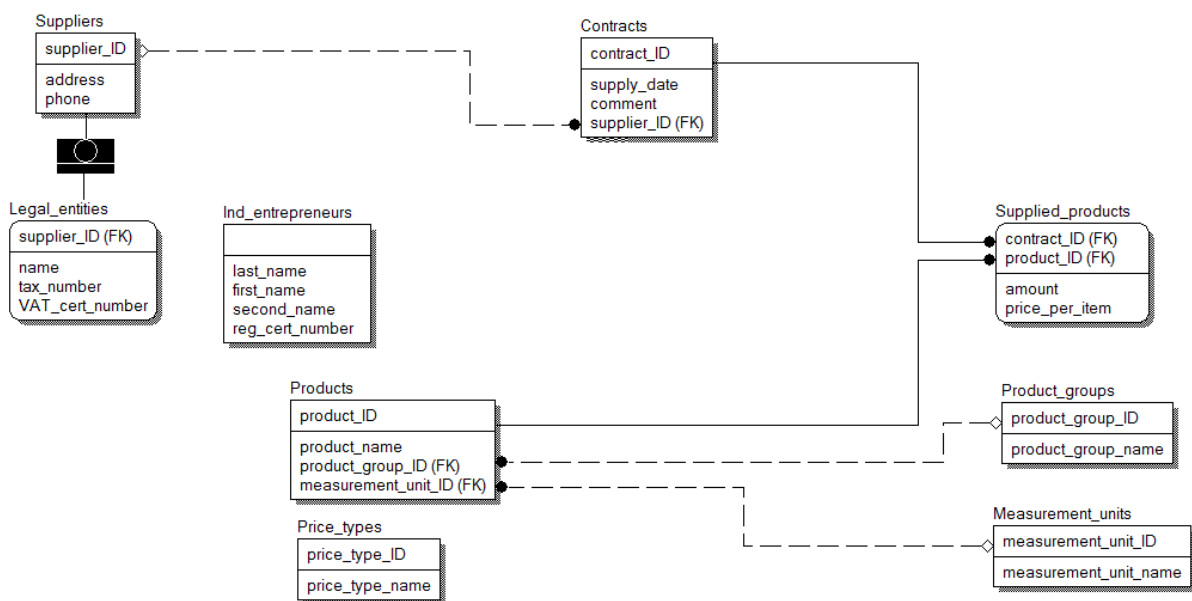


Fig. 11

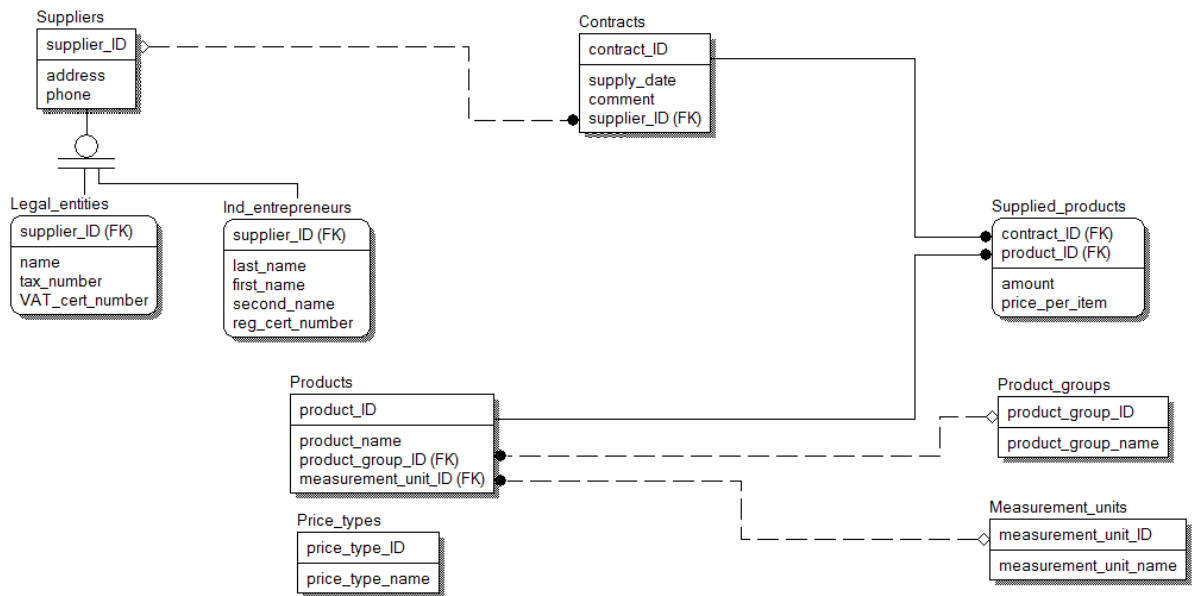


Fig. 12

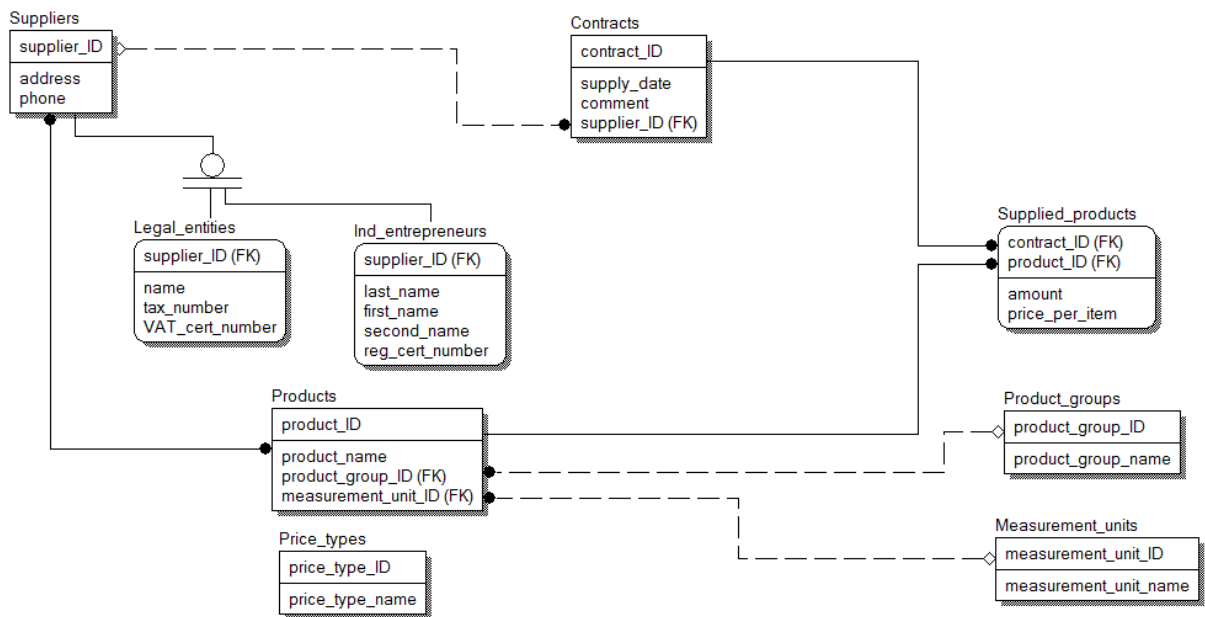


Fig. 13

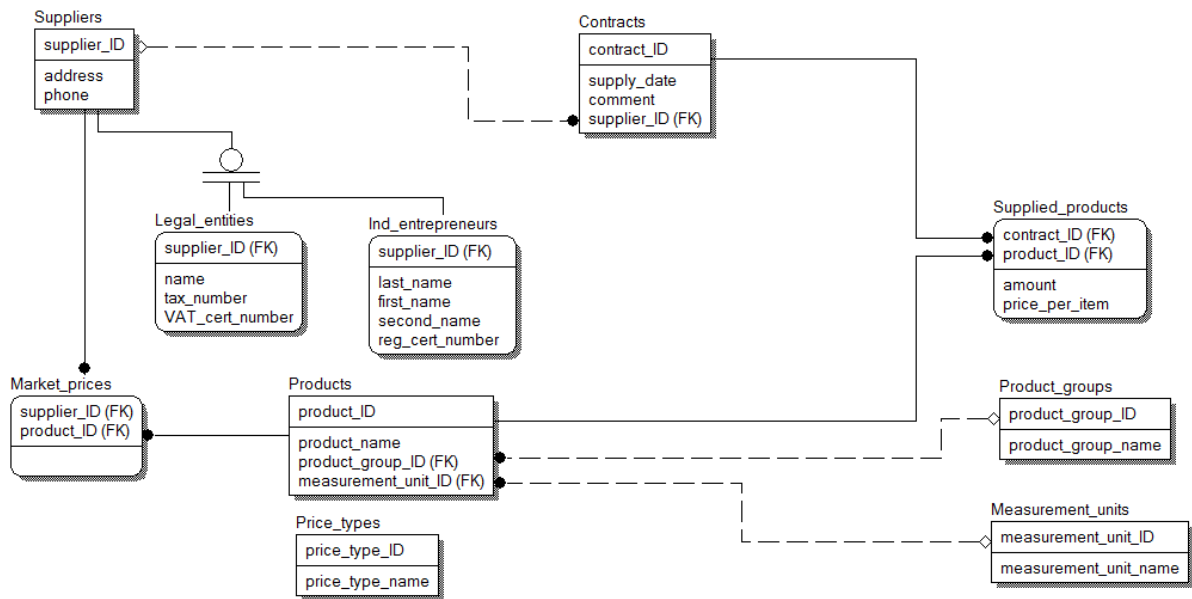


Fig. 14

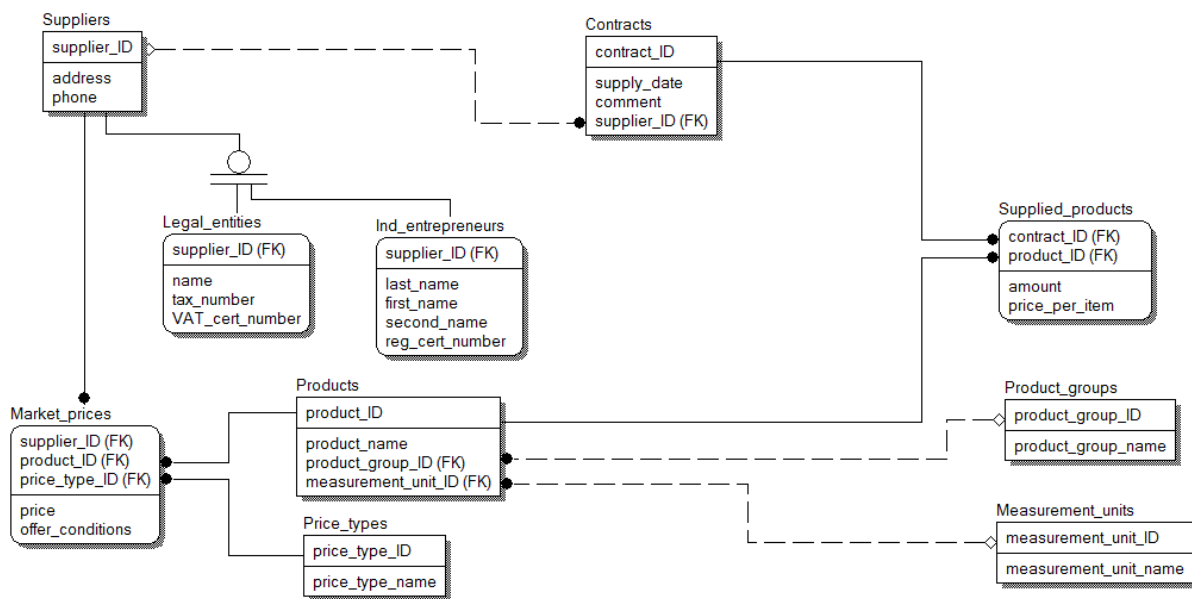


Fig. 15

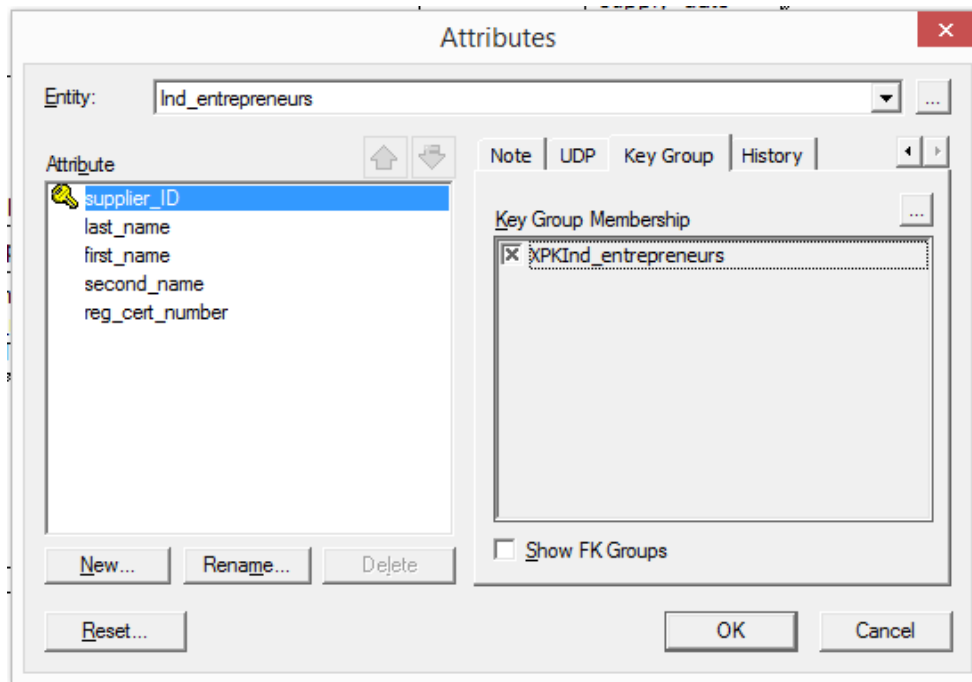


Fig. 16

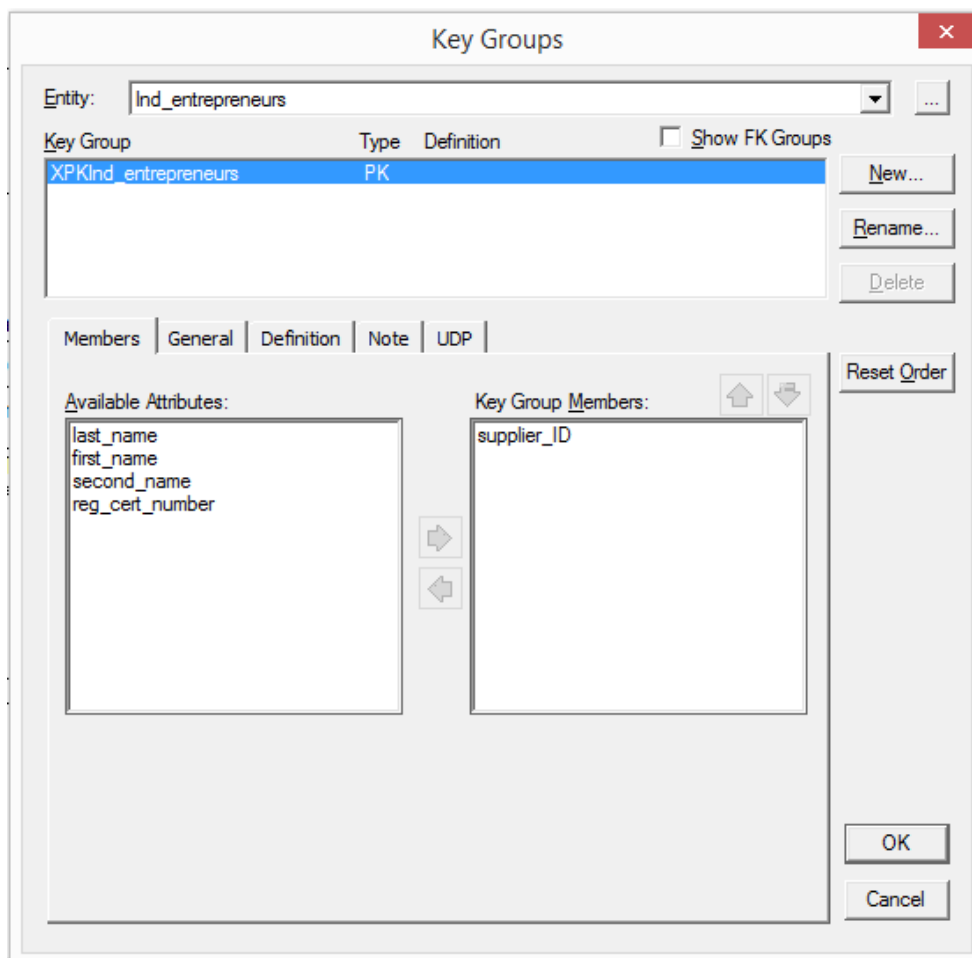
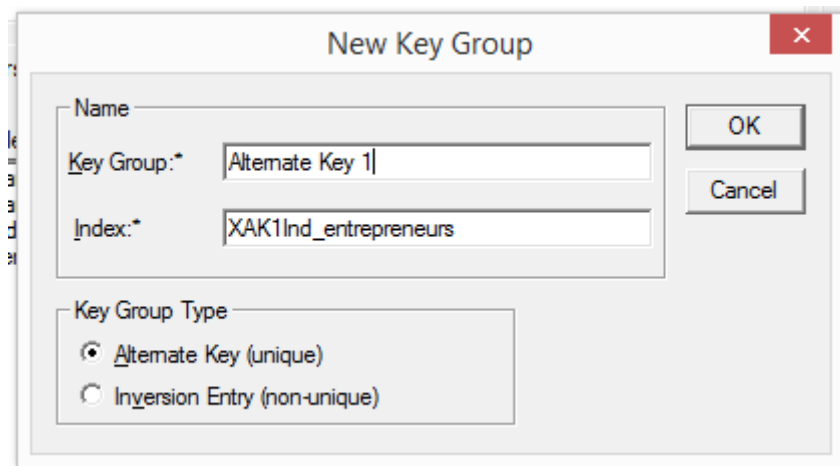
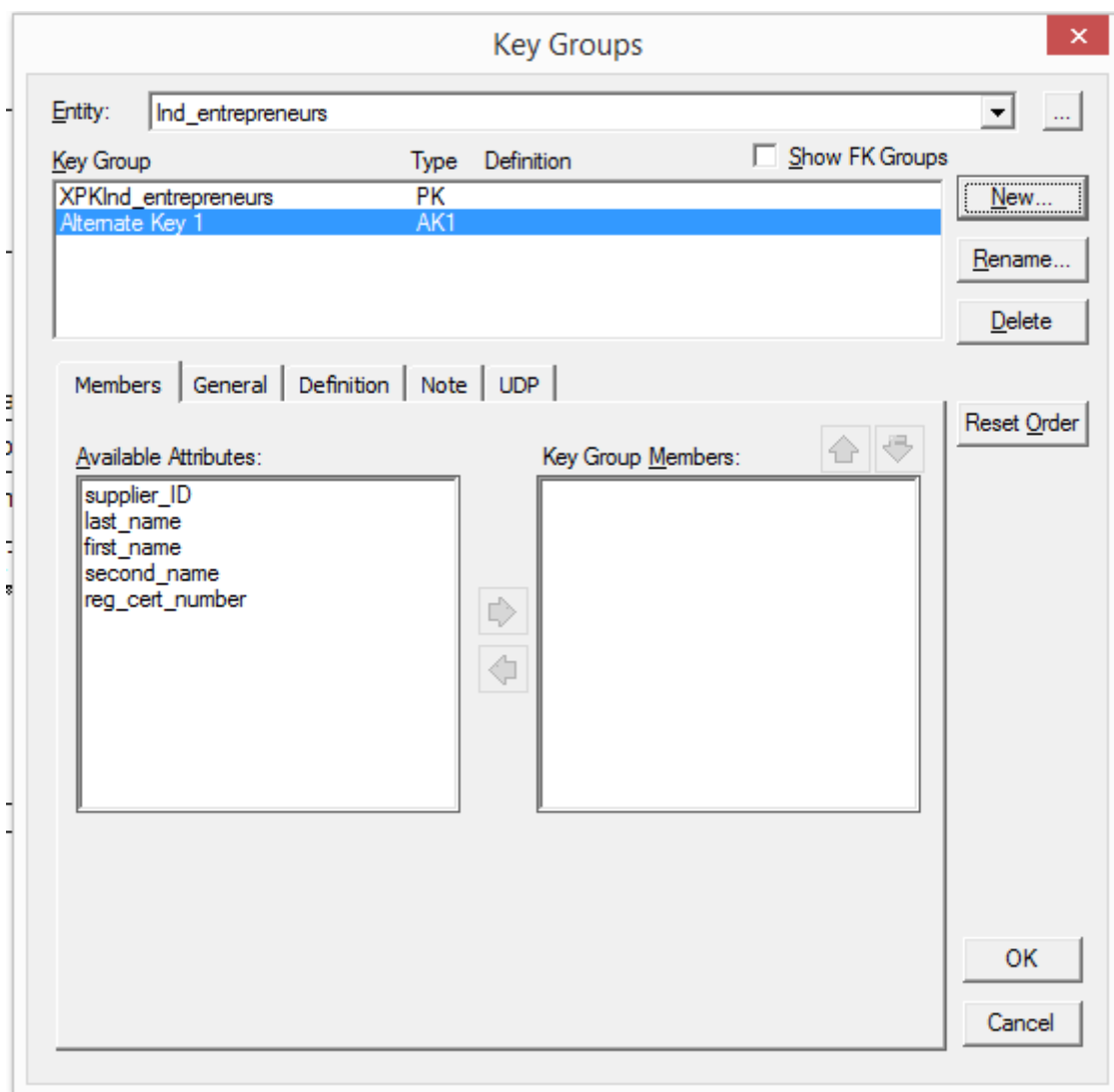


Fig. 17



The 'New Key Group' dialog box is shown. It has a title bar with a close button. Inside, there are two text input fields: 'Key Group:*' with the value 'Alternate Key 1' and 'Index:*' with the value 'XAK1Ind_entrepreneurs'. To the right of these fields are 'OK' and 'Cancel' buttons. Below the input fields is a section titled 'Key Group Type' containing two radio buttons: 'Alternate Key (unique)' (which is selected) and 'Inversion Entry (non-unique)'.

Fig. 18



The 'Key Groups' dialog box is shown. It has a title bar with a close button. At the top, there is an 'Entity:' dropdown menu set to 'Ind_entrepreneurs'. Below this is a table with columns 'Key Group', 'Type', and 'Definition'. The table contains two rows: 'XPKInd_entrepreneurs' with type 'PK' and 'Alternate Key 1' with type 'AK1'. The 'Alternate Key 1' row is highlighted. To the right of the table are buttons for 'New...', 'Rename...', and 'Delete'. Below the table is a tabbed interface with tabs for 'Members', 'General', 'Definition', 'Note', and 'UDP'. The 'Members' tab is active. It shows two list boxes: 'Available Attributes' (containing 'supplier_ID', 'last_name', 'first_name', 'second_name', 'reg_cert_number') and 'Key Group Members' (empty). Between the list boxes are two arrow buttons. To the right of the list boxes is a 'Reset Order' button. At the bottom right are 'OK' and 'Cancel' buttons.

Key Group	Type	Definition
XPKInd_entrepreneurs	PK	
Alternate Key 1	AK1	

Fig. 19

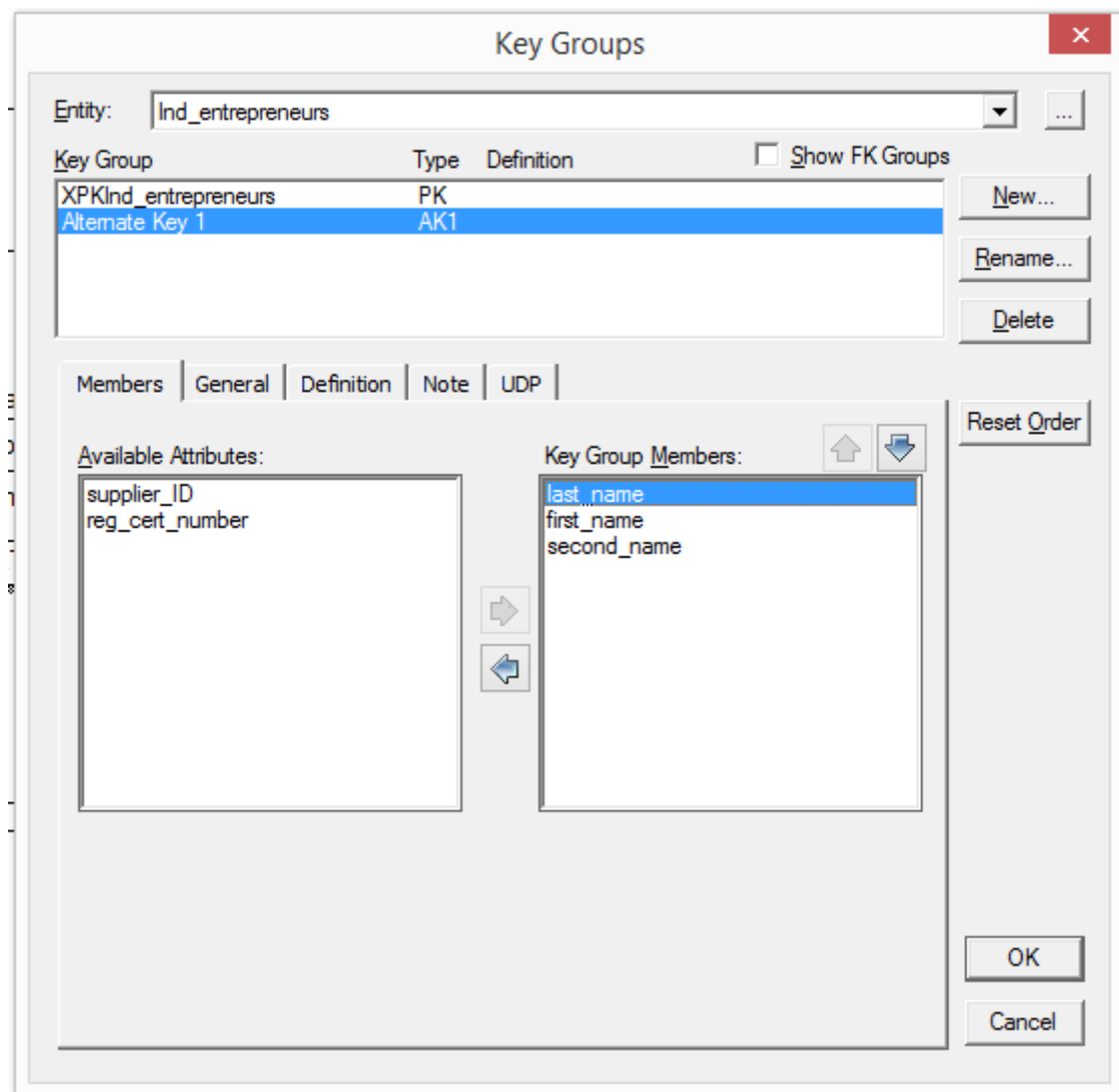


Fig. 20

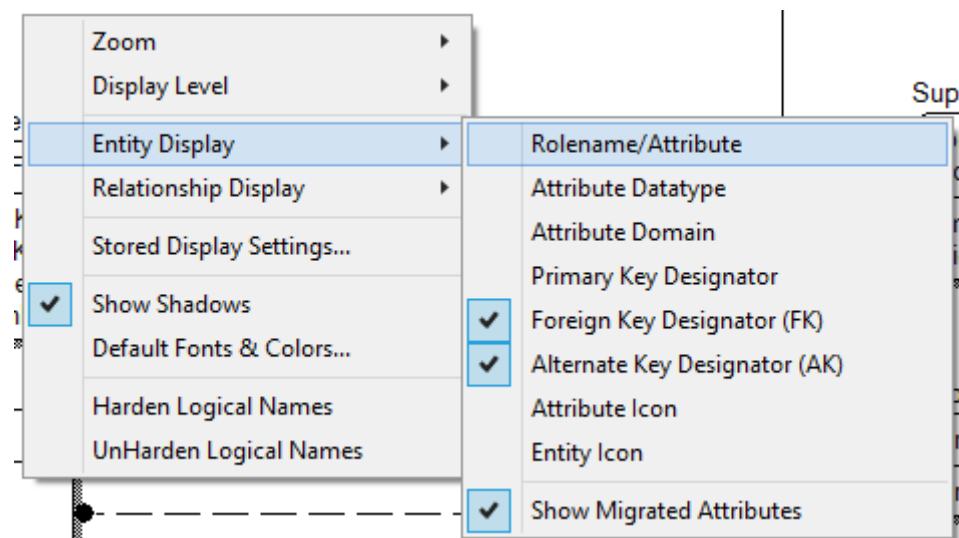


Fig. 21

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 (fig. 16) and click “...”. The “Key Groups” window with the data of keys of this entity will be shown (fig. 17);
- 3) Click “New” to create new alternate keys using “New Key Group” window (fig. 18), then click “Ok” (fig. 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” (fig. 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)” (fig. 21). As the result, alternate keys among the entity attributes will be shown (fig. 22).

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 (fig. 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
- Bottom Buttons:** OK, Cancel

Fig. 23

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.

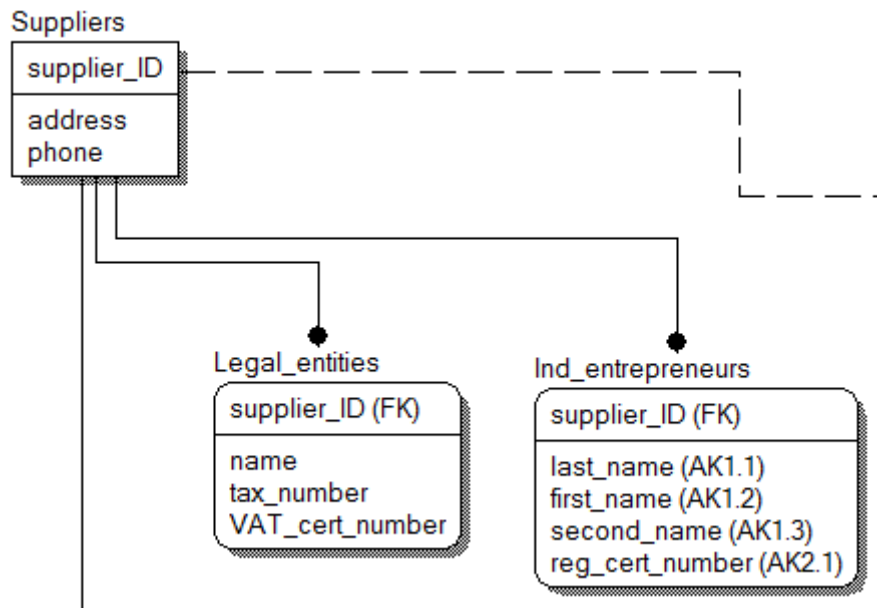


Fig. 24

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.

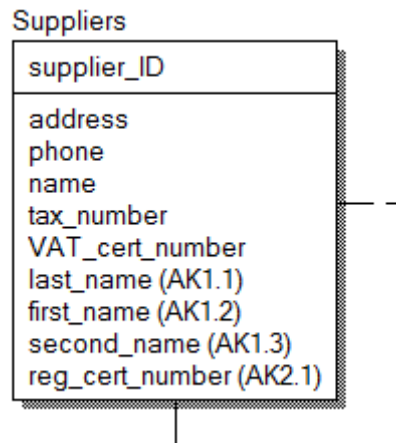


Fig. 25

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.

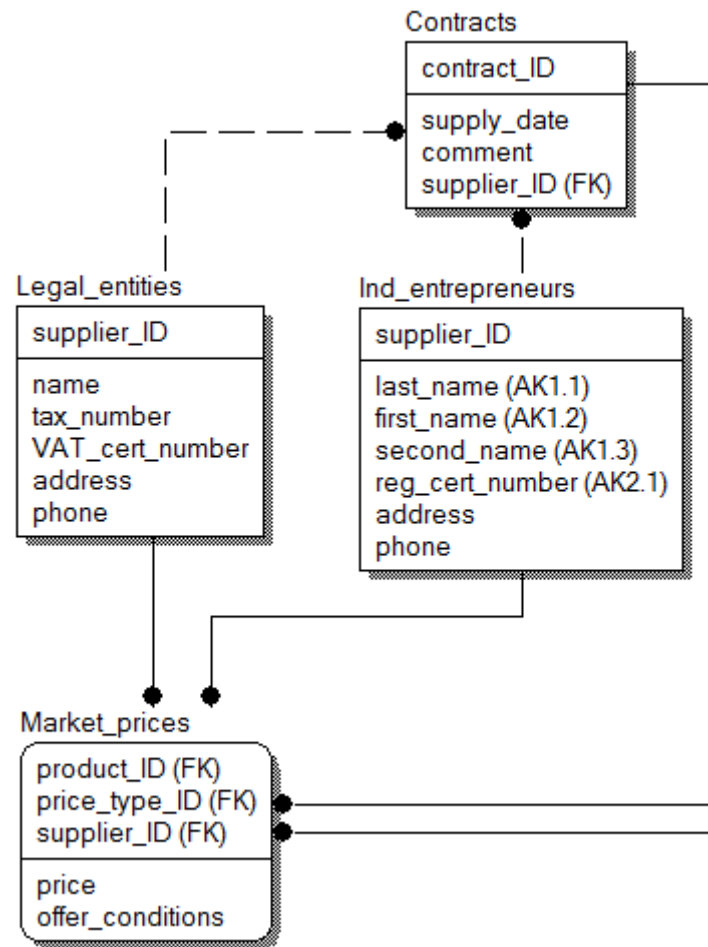


Fig. 26

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 (fig. 27).

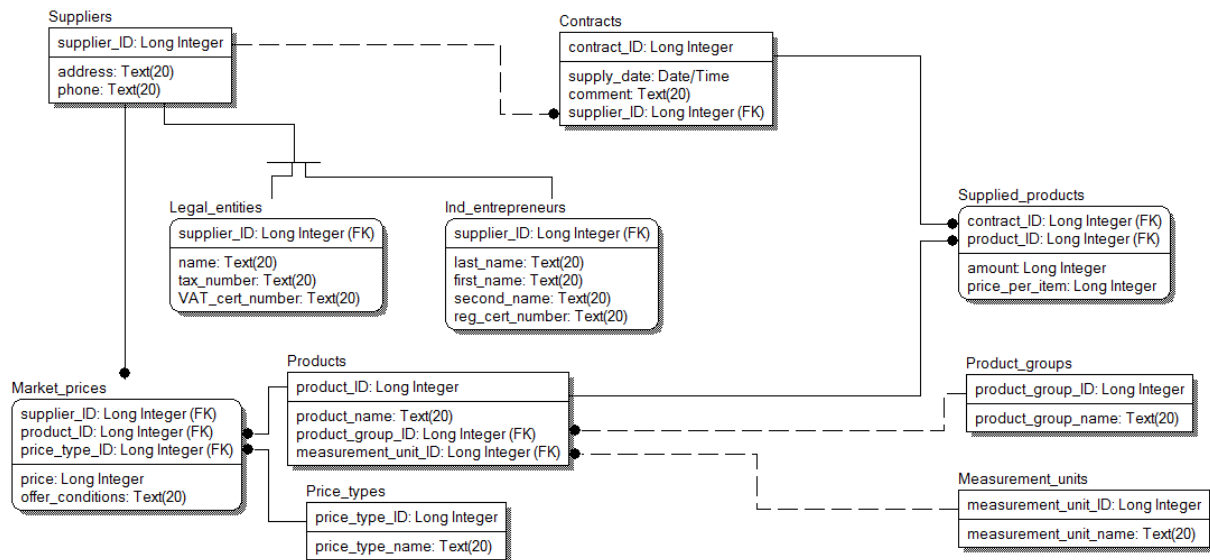


Fig. 27

11. Save created model and finish work.

12. Report requirements

- 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.