****

**Faculty of Contemporary Sciences and Technologies**

**Tetovo**

**Database**

**(Sem. 4, 2019/2020)**

**Project:**

***The Database Application for the Problem of Selling Goods***

**Student(s)**: **Mentor**:

1. First student (ID, Name Surname) Prof. dr. Jaumin Ajdari
2. Second Student (ID, Name Surname)
3. Third Student (ID, Name Surname)

May 2019, Tetovo

Abstract.

A database application where will be stored the data and organization of a warehouse. The warehouse sells the goods, it is important to store the data about the goods, data as description of the good, amount, price and other data. The warehouse is interested to store and save the data about the clients which buy from the warehouse and also the employees which helps in selection and preparing the orders for the client buyers. Another important thing is storing the orders where a order is stored as a invoice (something like wider bill).

To solve the above problem, we plan to use Microsoft SQLServer for DBMS (into which will be implemented the database) and Microsoft Access as a tool for interface implementation.

**Key words:** Warehouse of goods, Database, Application, DBMS, SQLServer, MS Access.

Optional (if the project is done from a group of students):

1. The work of the first member of the group (explain he/she works)
2. The work of the second member of the group (explain he/she works)
3. The work of the third member of the group (explain he/she works)

Table of Contents

[1. Problem description 3](#_Toc32592448)

[2. Business rules 3](#_Toc32592449)

[2. Conceptual design 4](#_Toc32592450)

[3. Relational design 7](#_Toc32592451)

[4. Schema normalization 8](#_Toc32592452)

[5. SQL Code and implementation 9](#_Toc32592453)

[5.1. Database diagram 10](#_Toc32592454)

[5.2. Data insertion 10](#_Toc32592455)

[6. The Interface 13](#_Toc32592456)

[6.1. Connection to MS SQLServer 13](#_Toc32592457)

[6.2. Insert forms 13](#_Toc32592458)

[6.3. Tables with data 15](#_Toc32592459)

[6.4. Reports 16](#_Toc32592460)

[6.5. Examples 17](#_Toc32592461)

# Problem description

A retail sell company is interested to automatize its selling activity. Company contains a warehouse of goods, employees who are engaged for the selling, articles and clients.

The organization is: a client comes to the warehouse, he/she makes a order and engaged person realize the order in the meaning that client isn’t allowed to go inside and gather the goods. All the requested goods are packed and bring or send to the client.

The company manager is interested to transform the selling administration from the old traditional way to a computerized way. Therefore, he asks a database application which solves its problem. The manager is interested to store data about articles which are described with the code, name, description, unit measure, quantity, tax, and so on. For the purpose of the advertisement, he is interested to save some client data like name, surname, address, phone number or e-mail as well as the frequency of visiting of the warehouse. Company wants to send information to their client about new actions of company like discounts, or any new article (goods) or retail or something else. The company rule is that clients are not allowed to go inside and for their orders will be engaged a company employee. So, the employee gets the order and he/she is responsible to realize the whole order and bring or send it to client. This application needs to solve only the problem of selling, therefore, the manager isn’t interested, in this application, to save employment data about the employees. The employee needs to be identified with its name, surname, company employee code and internal phone number. Also, the company wants to store sell history, so, is interested to store invoices and each invoice has to be described by its unique code, date of release, value of the order as well as the content of the order (for each goods the quantity, sell price and tax).

Company is interested in a database application.

# Business rules

We should design a database to collect and store data about the goods, clients (customer), employees, invoices and invoice content (invoice description). The database should fulfil the requirements:

1. The client orders the goods (articles) and an order made from a client is entered as an invoice,
2. The client makes more invoices, but, a certain invoice belongs to a client (just to one client, the invoice has only one owner),
3. The invoice contains the release date and the total amount,
4. Each invoice contains its description where are registered all the ordered goods, the amount to each goods and the price. The goods prices can vary from one to another invoice (so for the same goods it is possible to have different prices),
5. Employee prepares the goods for clients (the orders for the client), so, they prepare (implement) the invoices (client isn’t allowed to take by itself the goods)
6. A certain employee implements whole invoice (so, the employee must finish the started job i.e. implementation of the invoice) and the employee implements more invoices.

# Conceptual design

We should find the entity types and possible attributes and draw the entity-relationship diagram (E-R diagram) for given problem. The problem will be solved according to the rule based on four steps which are given on the our exercises.

* **Step 1.** We try to find entity types and for each entity type all the possible attributes. By reading the text we find the entity types: Article (product, goods), Client (customer), Employee and Invoice (invoice or bill of the sold goods). Therefore, we have

1. Article 2. Client

3. Employee 4. Invoice

* **Step 2.** Now, we start from Article, all the attributes are directly connected to goods (article), therefore all the attributes will stay as attributes of Article. Regarding the entity type Client, the attributes cid, cname and address gives information (or they describe) about the clients, i.e. cid is ID number of the client, cname is name and last name of the client and address is address of the client. As we know client makes bills (invoices), but, information about invoices are stored into entity types Invoice. Therefore, the client through the attribute Invoice takes (or reads) information from Invoice (entity types) and this cannot stay as an attribute. This attribute will be replaced with a relationship between Client and Invoice.

The same is with Employee entity type, because employees implement the orders (the invoices), and the proposed attribute Invoice must be replaced with a relationship between Employee and Invoice.

Invoice entity type, according to the requested (from the text), have to be described with released date, and amount, but, it must be fulfilled with the articles (to obtain the total amount). It means that attribute Articles is a list of the items (articles), i.e. content of the invoice and this attribute indicates to a relationship between Article and Invoice.

The E – R diagram, know is

* **Step 3**. Based on the step 2 and the requests we are able to define relationships
  + The relationship, named **Order**, between Client and Invoice, starting from the request: the client can make more orders (invoices) and an order (invoice) belongs only to one client, we can conclude that this will result to a relationship with cardinality one-to-many (1:M) in direction Client toward Invoice. Because the invoice will exist only if it is paid (or ordered) from a client, it means that, every invoice must be in relation (or connected) to a client. To be fulfilled the last statement we have to define total participation of the entity type Invoice to relationship Order.
  + The relationship, named **Prepare** is between Employee and Invoice. Starting from the requests: the employee implements more orders (invoices) and an order (invoice) must be totally implemented from one employee, the cardinality of relationship must be one-to-many (1:М) from Employee towards Invoice. Because, all the invoices must be implemented from an employee, this indicates that the entity type Invoice have to have total participation to the relationship Prepare.
  + The relationship, named **Invoice\_description** is between Article and Invoice, and starting from the request: an invoice (order) can contain more articles and the same article (may with different quantity) can be appeared into different invoices (orders), therefore the relationship is of the cardinality many-to-many (N:М). As in invoice is requested to note the item amount (article quantity, goods quantity) and the price can vary from invoice to invoice (depending on the amount or any other reason, let say the client has any discount, the price of the item might be different), the relationship will have its additional descriptive attributes,

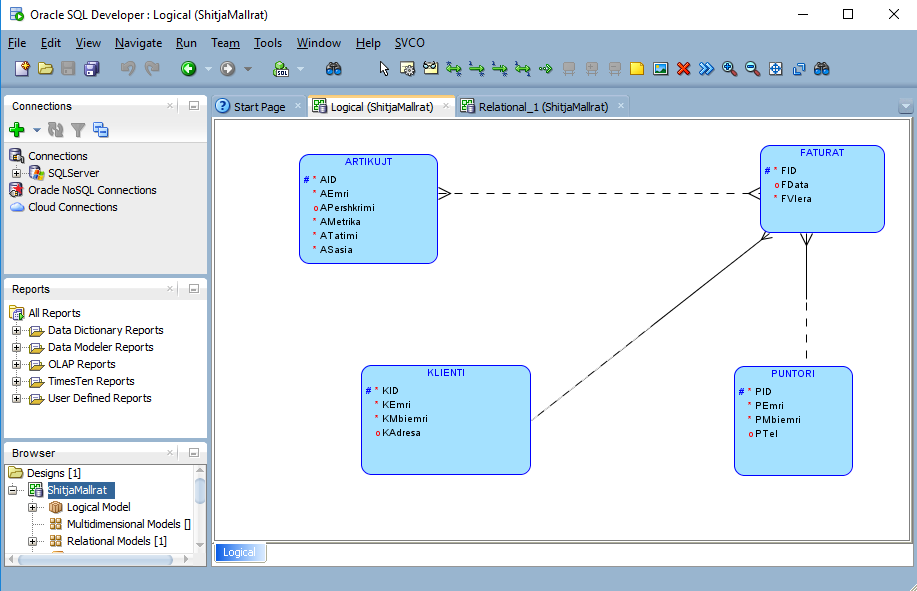


* **Step 4**. In the finish of the step 3 we expect to have ready the E – R diagram. Now, we have just to review and to do any improvement if it is needed. Therefore, we read the requests and find those into E – R diagram

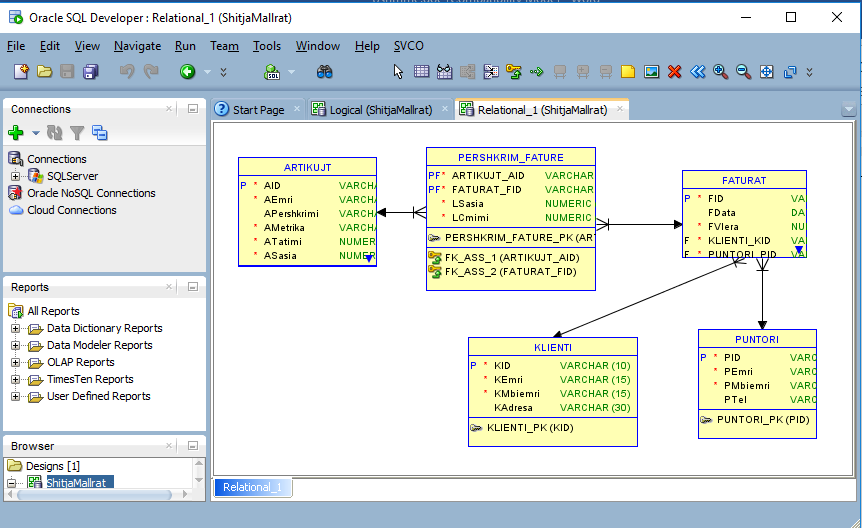


Solution by use of Oracle SQL Developer

1. Conceptual modeling (E – R diagram, Logical Design)



1. Relational Model



# Relational design

The E-R diagram for given problem is



With use of the relational mapping rules we convert the E – R diagram to relational model (relational diagram). We solve step by step the conceptual elements within E – R diagram.

1. Entity types, mapping of the entity types









1. Relationships, Mapping of the relationships,

***Order***: 1:N between Client and Invoice, Invoice has total participation (mandatory participation). Invoice relation will be changed



***Prepare***: 1:N between Employee and Invoice, Invoice has total participation (mandatory participation). According to the rule of mapping of the relationship with key constraint, the relation Invoice again will be changed



***Invoice\_description***: M:N between Article and Invoice, the relationship has additional its attributes and according to the rule of mapping of relationship with no key constraints, we obtain a new relation



1. The whole solution



# Schema normalization

According to above relational diagram the relations are

Article(AID, name, description, metric, tax, quantity)

Invoice(ID, date, value, CID, EID)

Client(CID, cname, address)

Employee(EID, name, surname)

Invoice\_Description(AID, ID, amount, price)

The first four relations contain primary key which consists only one attribute and all non-key attributes depend only from the primary key, therefore, those relations are in Boyce - Codd form (BCNF). The last relation has primary key composed from two attributes, so, we need to analyze the other non-key attributes. Regarding the attribute amount, this depends from AID because always when we buy something the amount depends on the thing which we buy. Also, different bills can have different amount from the same article, which help us to conclude for the attribute amount which always depends from both key attributes. Similar reasoning we can apply to the next attribute price. Because the price could be different from bill to bill this attribute depends not only from AID but to both key attributes.

Finally, the database schema is in Boyce-Codd normal (BCNF) form.

# SQL Code and implementation

SQL, Database creation and table creation, the tables creation order is determined by the arrows which are used into step 3 (part 3), therefore, firstly, we create the tables which are not referenced from any other table and always the table which references to any other table will be created after the table to which refers (after table to which points).

--Create new database

use master

go

--now we are able to delete database, if exists

if (DB\_ID('GoodsSelling') is not null)

drop database GoodsSelling

go

--recreate the database

create database GoodsSelling

go

--use of the database (makes it as the working database)

use GoodsSelling

go

--Table creation

create table ARTICLE

(

aid varchar(10),

aname varchar(15) constraint aname\_not\_null not null,

--define not null as a constraint

adescription varchar(15) constraint adescription\_not\_null not null,

ametric varchar(6) constraint ametric\_not\_null not null,

atax numeric (5,2) constraint atax\_default default 0,

--define default value as a constraint

aquantity numeric (10,2),

aprice numeric(10,2),

constraint Article\_PK primary key (aid)

)

go

create table CLIENT

(

cid varchar(10) constraint Client\_PK primary key,

cname varchar(20) constraint cname\_not\_null not null,

caddress varchar(30)

)

go

create table EMPLOYEE

(

eid varchar(10) constraint Employee\_PK primary key,

ename varchar(20) constraint ename\_not\_null not null,

esurname varchar(20) constraint esurname\_not\_null not null

)

go

create table INVOICE

(

Iid int,

idate date,

ivalue numeric (10,2) default 0,

clid varchar(10) constraint cid\_not\_null not null,

eid varchar(10) constraint eid\_not\_null not null,

constraint Invoice\_PK primary key (iid)

)

go

create table INVOICE\_DESCRIPTION

(

aid varchar(10),

iid int,

amount numeric (10,2),

Discount numeric(5,2) default 0.0,

price numeric (10,2),

constraint Invoice\_description\_PK primary key (aid, iid)

)

Go

--Relational integrity, defining the foreign keys this

--part also can be defined inside the table creation

alter table INVOICE add

constraint InvoiceClient\_FK foreign key (clid) references CLIENT(cid),

constraint InvoiceEmployee\_FK foreign key (eid) references EMPLOYEE

go

alter table Invoice\_description add

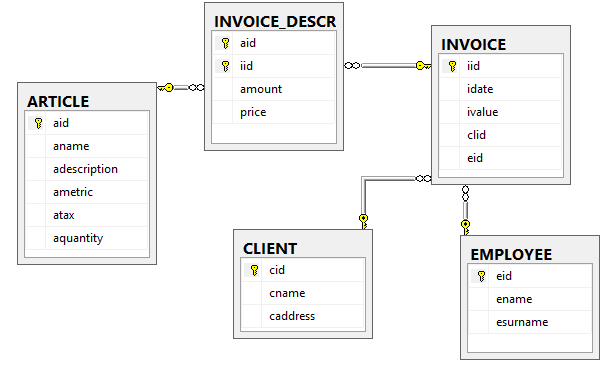
constraint Invoice\_description\_Article\_FK foreign key (aid) references ARTICLE,

constraint Invoice\_description\_Invoice\_FK foreign key (iid) references INVOICE

go

## Database diagram

Database diagram (relation diagram)



## Data insertion

Filling the tables with data,

Articles

INSERT INTO ARTICLE (aid, aname, adescription, ametric, atax, aquantity, aprice)

VALUES

('U0001', 'Sugar', 'White sugar', 'kg', 5, 1000, 34),

('U0002', 'Sugar', 'Gray sugar', 'kg', 5, 500, 44),

('U0003', 'Flour', 'T400', 'kg', 5, 1500, 22),

('U0004', 'Flour', 'T500', 'kg', 5, 1500, 20),

('U0005', 'Oil', 'Olive oil', 'liter', 5, 200, 205),

('U0006', 'Oil', 'Sunflower oil', 'liter', 5, 1000, 67),

('U0007', 'Butter', 'Sunflower butter', 'kg', 18, 100, 67),

('U0008', 'Patatos', 'Fresh patatos', 'kg', 18, 100, 25),

('U0009', 'Tomatoes', 'Fresh tomatoes', 'kg', 18, 100, 65),

('U0010', 'Carrots', 'Fresh tomatoes', 'kg', 18, 100, 65),

('U0011', 'Mushrooms', 'Fresh Mushrooms', 'kg', 18, 100, 65),

('U0012', 'Onions', 'Fresh onions', 'kg', 18, 100, 25),

('U0013', 'Garlic', ' ', 'kg', 18, 100, 85),

('H0001', 'Head&Shoulders', 'Shampon 400ml', 'liter', 18, 200, 225),

('H0002', 'Arial', 'Detergent 3kg', 'kg', 18, 200, 185),

('H0003', 'Dove', 'Body saop 150 mg', 'kg', 18, 200, 185),

('H0004', 'Colgate Max White', 'Toothpaste', 'kg', 18, 200, 185),

('H0005', 'Lacalut', 'Toothpaste', 'kg', 18, 200, 185)

Client

INSERT INTO CLIENT (cid, cname, caddress)

VALUES

('0001', 'Artan Artani', 'Tetovo'),

('0002', 'Blerton Blerti', 'Gostivar'),

('0003', 'Florim Flori', 'Kicevo'),

('0004', 'Flutra Fuati', 'Tetovo'),

('0005', 'Olti Olti', 'Skopje'),

('0006', 'Oli Oli', 'Ohrid'),

('0007', 'Butrint Butrinti', 'Tetovo'),

('0008', 'Patriot Petriti', 'Skopje'),

('0009', 'Tomi Tomi', 'Tetovo'),

('0010', 'Meri Meri', 'Skopje')

Employee

INSERT INTO EMPLOYEE (eid, ename, esurname)

VALUES

('E0001', 'Toni', 'Toni'),

('E0002', 'Arta', 'Arta'),

('E0003', 'Flori', 'Florimi'),

('E0004', 'Mustaf', 'Mustafa'),

('E0005', 'Petrit', 'Petriti')

Invoice

INSERT INTO INVOICE (iid, idate, ivalue, cid, eid)

VALUES

(2, '2018-03-09', 689.50, '0001', 'E0001'),

(4, '2018-03-09', 789.00, '0002', 'E0002'),

(6, '2018-03-09', 715.00, '0003', 'E0002'),

(8, '2018-03-09', 377.90, '0004', 'E0003'),

(10, '2018-03-09', 354.00, '0007', 'E0004'),

(12, '2018-03-09', 170.00, '0008', 'E0004'),

(14, '2018-03-09', 618.95, '0009', 'E0005'),

(16, '2018-03-09', 65.00, '0010', 'E0005')

Invoice\_Description

INSERT INTO INVOICE\_DESCRIPTION(aid, iid, amount, price, Discount)

VALUES

('H0003', 4, 3.00, 185.00,0.00),

('H0004', 8, 1.00, 185.00, 0.00),

('H0005', 14, 2.00, 185.00, 5.00),

('U0001', 2, 5.00, 34.00, 0.00),

('U0001', 12, 5.00, 34.00, 0.00),

('U0002', 6, 5.00, 44.00, 0.00),

('U0002', 8, 3.00, 44.00, 5.00),

('U0003', 10, 10.00, 22.00, 0.00),

('U0004', 4, 5.00, 20.00, 0.00),

('U0005', 2, 2.00, 205.00, 5.00),

('U0006', 4, 2.00, 67.00, 0.00),

('U0006', 10, 2.00, 67.00, 0.00),

('U0007', 14, 3.00, 67.00, 5.00),

('U0008', 6, 5.00, 25.00, 3.00),

('U0009', 6, 5.00, 65.00, 5.00),

('U0010', 2, 2.00, 65.00, 0.00),

('U0010', 6, 1.00, 65.00, 0.00),

('U0011', 16, 1.00, 65.00, 0.00),

('U0012', 8, 3.00, 25.00, 10.00),

('U0013', 14, 1.00, 85.00, 10.00)

# The Interface

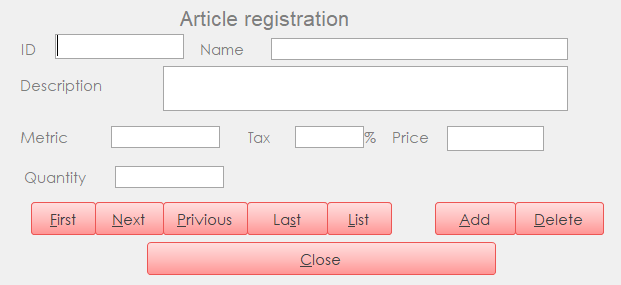
## Connection to MS SQLServer

Database is stored into MS SQLServer and the use of the database is implemented from MS Access. The connection from Access to SQLServer is implemented by use of ODBC connection.

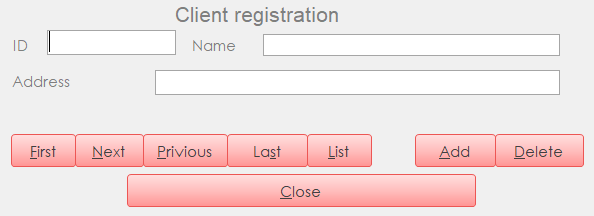
## Insert forms

To insert data into our database we use the forms:

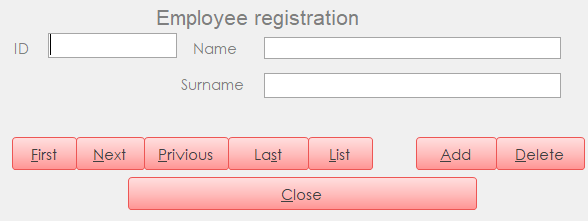
To insert a row into table ARTICLE is created the form



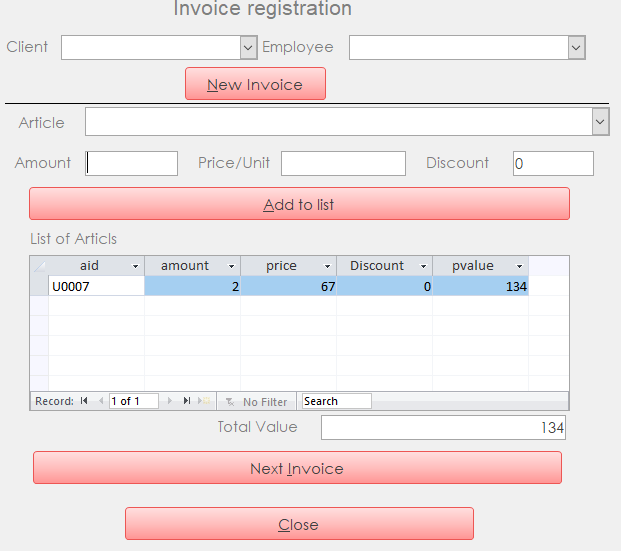
For insert to CLIENT, we use the form



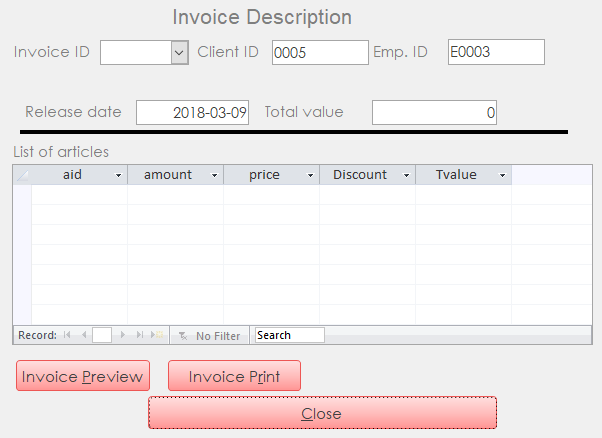
For EMPLOYEE the form



For INVOICE the form



For INVOICE\_DESCRIPTION



## Tables with data

Table ARTICLE

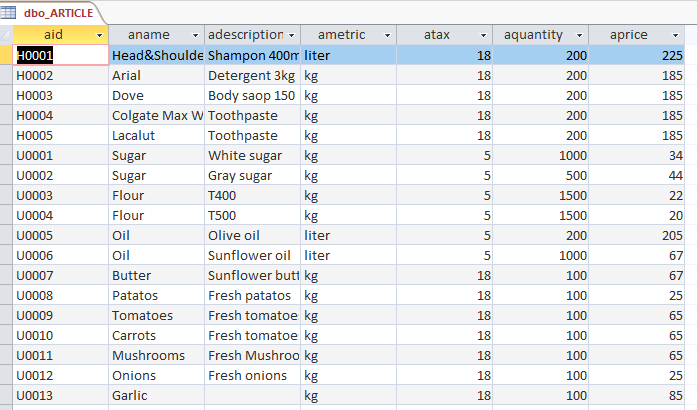


Table CLIENT

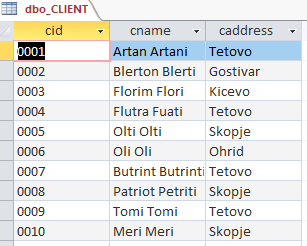


Table EMPLOYEE

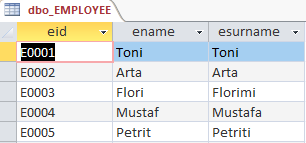


Table INVOICE

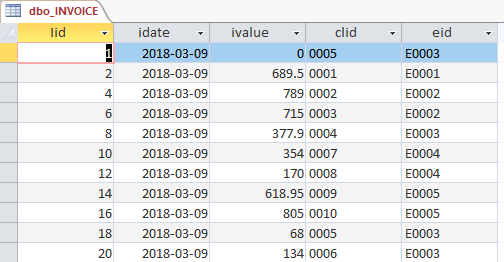
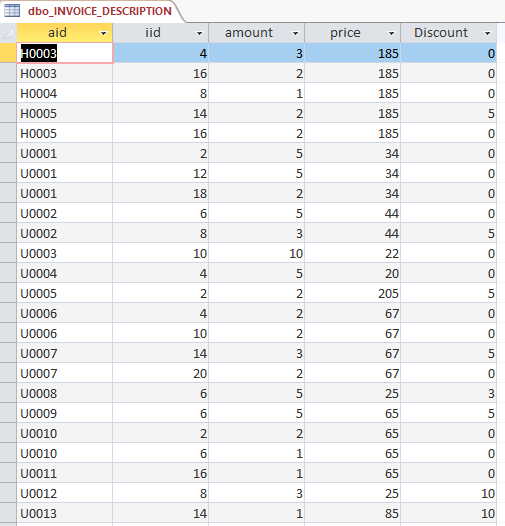
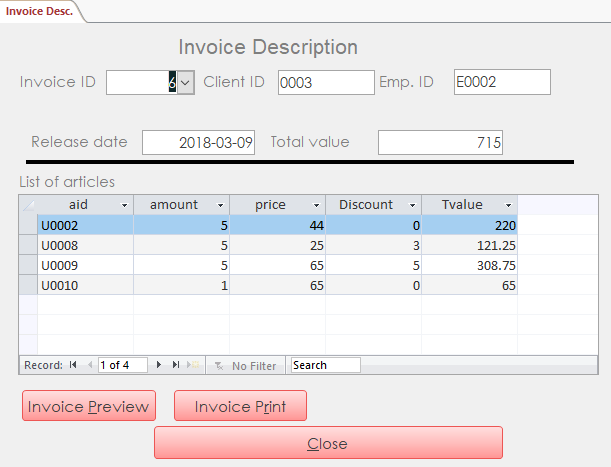


Table INVOICE\_DESCRIPTION

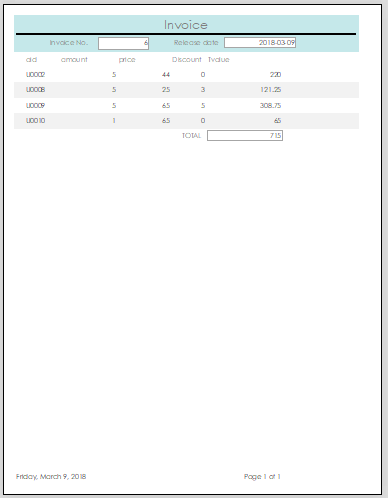


## Reports

Invoice description i.e. the list of the articles into a invoice can be shown with the next form



Also, the content of the invoice can be shown with the next report



## Examples

**Query 1.** Find all the invoices done from a client with name Flutra Fuati.

Relational algebra

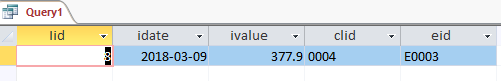
SQL

SELECT i.\*

FROM INVOICE i, CLIENT c

WHERE c.cname = 'Flutra Fuati' AND i.clid = c.cid

Results



**Query 2.** Find the content of the invoice done from the Flutra Fuati.

Relational algebra

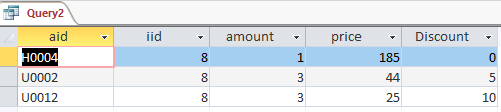
SQL

SELECT id.\*

FROM INVOICE i, CLIENT c, INVOICE\_DESCRIPTION id

WHERE c.cname = 'Flutra Fuati' AND i.clid = c.cid AND i.iid = id.iid

Results



**Query 3.** Find all the invoices realization from the employee with id E0001.

Relational algebra

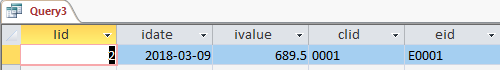
SQL

SELECT i.\*

FROM INVOICE i

WHERE i.eid = 'E0001'

Results



**Note**. This project as its part contains the Microsoft Access application GoodsSelling (GoodsSelling.accdb), as an interface which help to work with the database implemented into MS SQLServer.