# FINITE PRODUCTS ADMINISTRATION WITH MS ACCESS

**ILEANA HAUER, ANTOANETA BUTUZA, CORNELIA MUNTEAN** [*]

**ABSTRACT:** *As software applications become more and more complicated, it is essential to capture the requirements, model the system design in different phases, and communicate frequently among end users, business analysts and developers. Relational database management systems (RDBMS) are the dominant database technology and play an important role in enterprise software applications. We used MS Access to develop an application which underlines the finite products flux beginning with the execution stage to the sale stage.*

**KEY WORDS:** *software applications; database; relational database management systems.*

**JEL CLASSIFICATION:** *M15.*

## 1. INTRODUCTION

It is important to have a properly designed database so that accurate information can be provided to an organization. It is much easier to create an effective design initially so that necessary modifications to the database are kept at a minimum. Discovering problems after a database has been put into operation can be detrimental to a business, institution or organization.

There is a three-phase process in developing a database (Benaroch & Kauffman, 2000):

- Logical design: defining tables, fields, Primary and Foreign keys, establishing table relationships and levels of data integrity.
- Implementation of the logical design: using a DBMS to create tables and their relationships, using the tools to implement levels of data integrity.

[*] *Assist., Ph.D., West University of Timişoara, Romania, ileana.hauer@feaa.uvt.ro*
*Lecturer, Ph.D., West University of Timişoara, Romania, antoaneta.butuza@feaa.uvt.ro*
*Assist. Prof., Ph.D., West University of Timişoara, Romania, cornelia.muntean@feaa.uvt.ro*

o Development of end-user application.

Relational databases have a strong (mathematical) theoretical foundation (Codd, 1970; Chen, 1976), and a wide range of database software products available for implementing relational databases. Dr. E. F. Codd applied mathematical theories known as first order predicate logic and set theory to design relational databases. These theories are the foundation for the Relational Database Model (RDM). They are important because they makes the RDM predictable and reliable. It is not necessary to fully understand these theories to develop a sound database design.

But why we should use a relational database? The purpose of a database is to organize data so it can be easily selected, sorted, managed, and updated. By using a relational database, each piece of information only needs to be stored once. The benefit is that if a piece of data is updated it needs to be updated in only one place and the update will then be recognized throughout the database. Otherwise, several updates would need to be made in several different places, which can be cumbersome and lead to a messy situation if not done accurately.

Data is retrieved by specifying fields and tables using a standard query language known as Structured Query Language (SQL). Most DBMSs (Database Managements Systems) use SQL to build, modify, maintain and manipulate databases. Thorough knowledge of SQL isn’t always necessary since most DMBSs use a graphical interface to generate SQL statements and retrieve data. It is good, however, to have basic knowledge of SQL. Large volumes of centrally located shared data have come about in recent history, creating the need for client/server software. Data security and integrity can be implemented through the database server.

## 2. MATERIAL AND METHOD

### 2.1. Microsoft Office Access short history

Microsoft Office Access, previously known as Microsoft Access, is a relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. It is a member of the Microsoft Office suite of applications, included in the Professional and higher editions or sold separately. In mid-May 2010, the current version of Microsoft Access 2010 was released by Microsoft in Office 2010; Microsoft Office Access 2007 was the prior version. Access stores data in its own format based on the Access Jet Database Engine. It can also import or link directly to data stored in other applications and databases.

Prior to the introduction of Access, Borland (with Paradox and dBase) and Fox (with FoxPro) dominated the desktop database market. Microsoft Access was the first mass-market database program for Windows. With Microsoft's purchase of FoxPro in 1992 and the incorporation of Fox's Rushmore query optimization routines into Access, Microsoft Access quickly became the dominant database for Windows - effectively eliminating the competition which failed to transition from the MS-DOS world. Access 2007 introduced a new database format: accdb. ACCDB supports complex data types such as multivalue and attachment fields. These new field types are essentially recordsets in fields and allow the storage of multiple values in one field. With Access 2010, a new version of the ACCDB format supports hosting on a SharePoint 2010 server for exposure to the web.

## 2.2. Designing the database

Effective database designers will keep in mind the principles of normalization while they design a database. Normalization is a database design approach that seeks the following four objectives (Năstase, et al., 2001; Chen, 1976): minimization of data redundancy; minimization of data restructuring; minimization of I/O by reduction of transaction sizes, and enforcement of referential integrity. If the normalization goes to extremes it means that each piece of information may not appear more than once in a database (Muntean, 2003). Practically, however this is not always possible or desirable.

The following concepts and techniques are important to keep in mind when designing an effective database:

a) An entity is a logical collection of things that are relevant to your database. The physical counterpart of an entity is a database table.
b) An attribute is a descriptive or quantitative characteristic of an entity. The physical counterpart of an attribute is a database column (or field).
c) A primary key is an attribute (or combination of attributes) that uniquely identify each instance of an entity. A primary key cannot be null and the value assigned to a primary key should not change over time. A primary key also needs to be efficient. For example, a primary key that is associated with an INTEGER data type will be more efficient than one that is associated with a CHAR data type. Primary keys should also be non-intelligent; that is, their values should be assigned arbitrarily without any hidden meaning. Sometimes none of the attributes of an entity are sufficient to meet the criteria of an effective primary key. In this case the database designer is best served by creating an "artificial" primary key.
d) A relationship is a logical link between two entities. A relationship represents a business rule and can be expressed as a verb phrase. Most relationships between entities are of the "one-to-many" type in which one instance of the parent entity relates to many instances of the child entity.
e) The second type of relationship is the "many-to-many" relationship. In a "many-to-many" relationship, many instances of one entity relate to many instances of the other entity. "Many-to-many" relationships need to be resolved in order to avoid data redundancy. "Many-to-many" relationships may be resolved by creating an intermediate entity known as a cross-reference (or XREF) entity. The XREF entity is made up of the primary keys from both of the two original entities. Both of the two original entities become parent entities of the XREF entity. Thus, the "many-to-many" relationship becomes resolved as two "one-to-many" relationships.
f) A "foreign key" exists when the primary key of a parent entity exists in a child entity. A foreign key requires that values must be present in the parent entity before like values may be inserted in the child entity. The concept of maintaining foreign keys is known as "referential integrity".

g) Relationships between two entities may be classified as being either "identifying" or "non-identifying". Identifying relationships exist when the primary key of the parent entity is included in the primary key of the child entity. On the other hand, a non-identifying relationship exists when the primary key of the parent entity is included in the child entity but not as part of the child entity's primary key. In addition, non-identifying relationships may be further classified as being either "mandatory" or "non-mandatory". A mandatory non-identifying relationship exists when the value in the child table cannot be null. On the other hand, a non-mandatory non-identifying relationship exists when the value in the child table can be null.

h) Cardinality helps us further understand the nature of the relationship between the child entity and the parent entity. The cardinality of a relationship may be determined by asking the following question: "How many instances of the child entity relate to each instance of the parent entity?". There are four types of cardinality: (1.) One to zero or more (common cardinality), (2.) One to one or more (P cardinality), (3.) One to zero or one (Z cardinality), and (4.) One to exactly N (N cardinality).

## 3. RESULTS AND DISCUSSIONS

We used Microsoft Access to create a database with several tables (see figure 1). For each table we introduced the fields, the suitable attributes and the primary keys. The relationships between tables is one-to-many, it means that one record in a primary table is related to many records in a related table (figure 1).

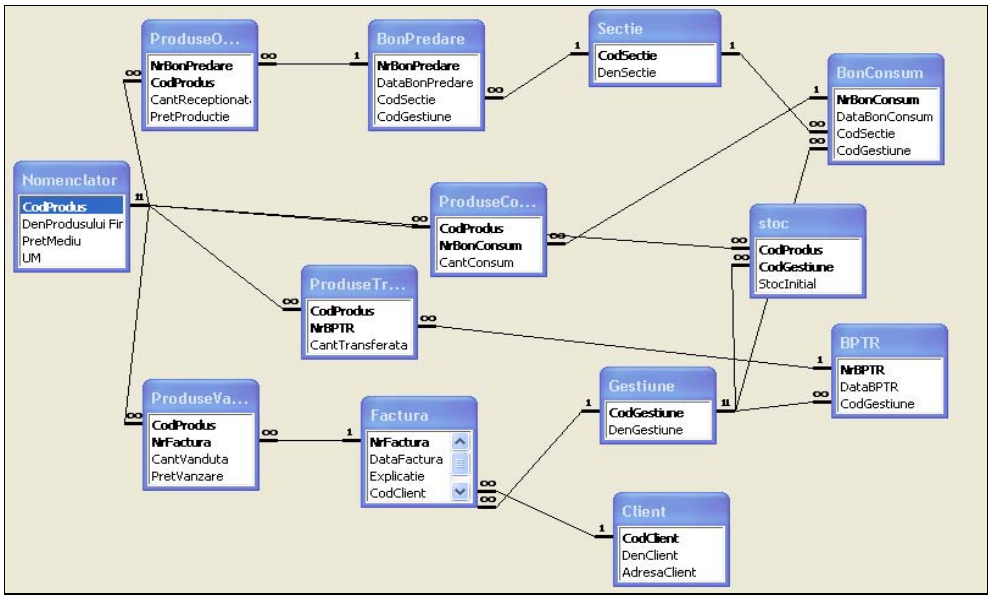


**Figure 1. Table relationships**

There is a menu form at beginning with several command buttons that allows a choice of different options. It contains the forms and reports which are presented using data and information regarding the Management of finite products. Forms shown in figure 2 are defined as templates that allow both data entry and data visualize.

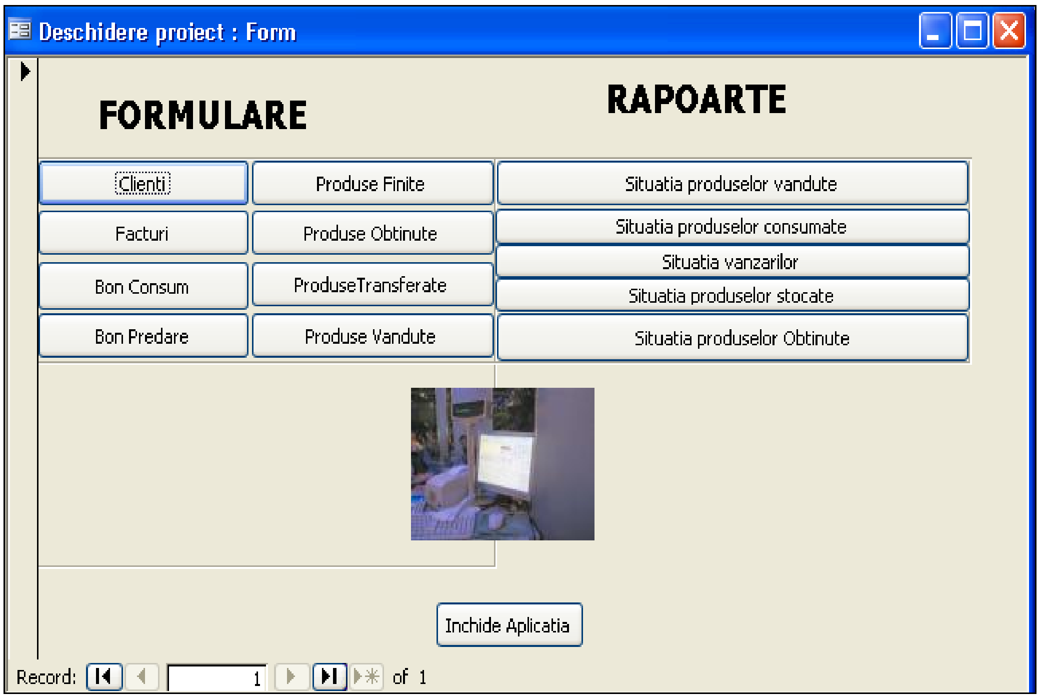


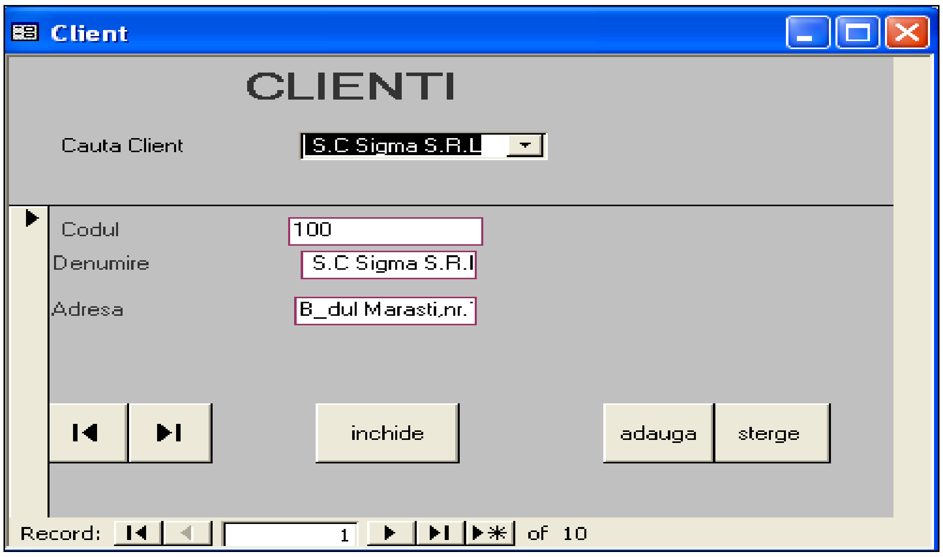
**Figure 2. Menu form with command buttons**

If we want to introduce a new customer in the database the command button Clienti will be selected. This form allows us to view the form Clienti for entering data about new customer (figure 3). This form has Clienti table as a data source. Data entered in this form will be stored in a table named Clienti.

By activating command buttons: Clienti, Facturi, Bon Consum, Bon Predare, Produse Finite, Produse Obtinute, Produse Transferate, Produse Vandute, are displayed forms that provides a detailed picture of Customers, Invoices, Consumer bill, Delivery bill, Finite products, Products obtained, Transferred products and Products sold (figure 4).

Each of these forms have other command buttons that allow us to close, to add, and also to delete records. Each command button has an associated procedure in Visual Basic.

The command buttons Rapoarte, which are located on the right side of the form from figure 2, can display different end situations (figure 5). The information processing is reflected in several reports. One of them, the report 'Situatia produselor stocate' provides information about storage of products: product code, name, code, quantity stored.



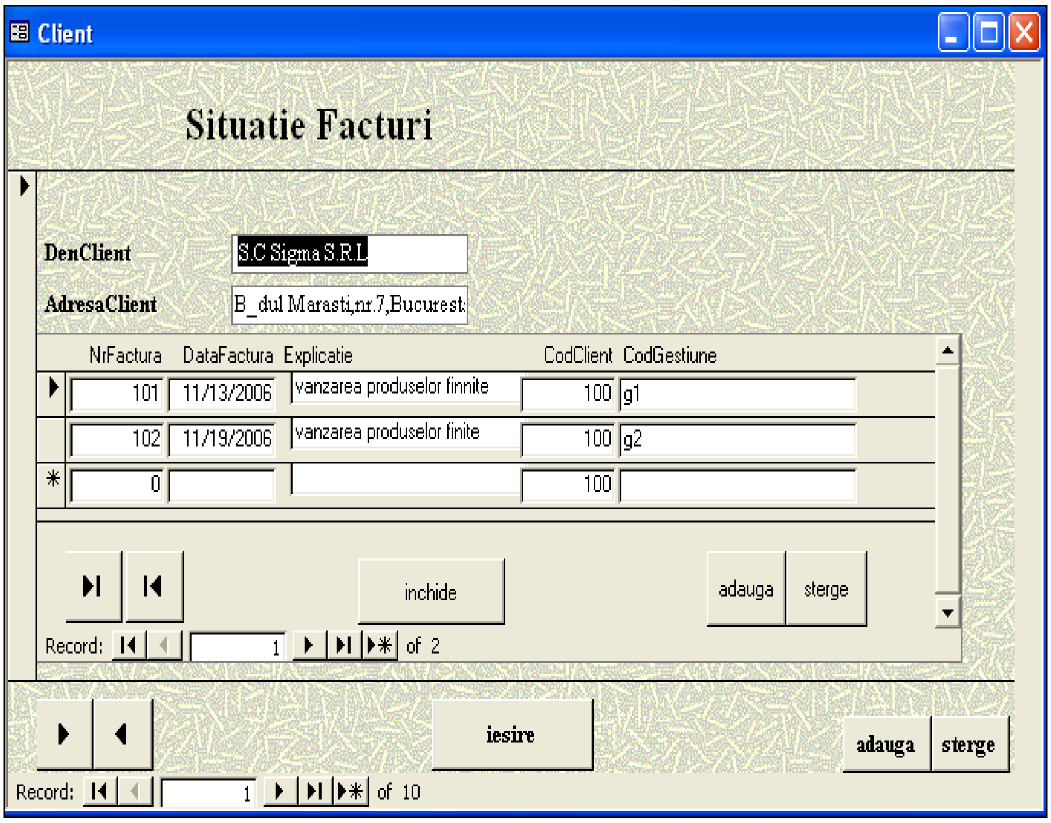
Client

CLIENTI

Cauta Client S.C Sigma S.R.L

| | |
|---|---|
| Codul | 100 |
| Denumire | S.C Sigma S.R.I |
| Adresa | B_dul Marasti,nr.7 |

**Figure 3. Form Clienti for entering new data about customers**



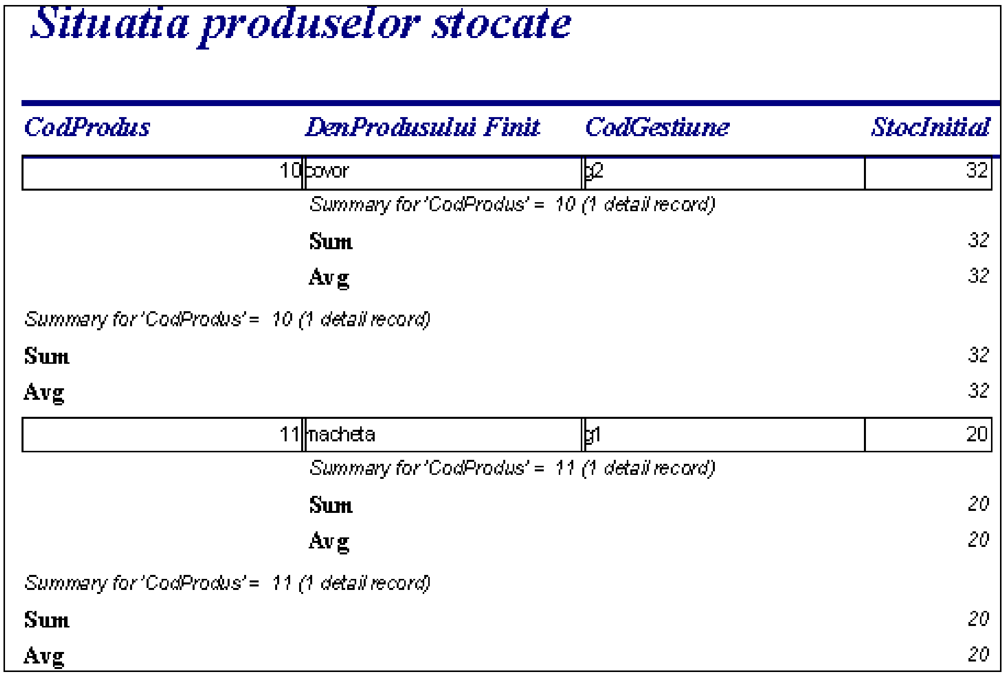
Client

Situatie Facturi

| | |
|---|---|
| DenClient | S.C Sigma S.R.L |
| AdresaClient | B_dul Marasti,nr.7,Bucurest |

| NrFactura | DataFactura | Explicatie | CodClient | CodGestiune |
|---|---|---|---|---|
| 101 | 11/13/2006 | vanzarea produselor finnite | 100 | g1 |
| 102 | 11/19/2006 | vanzarea produselor finite | 100 | g2 |
| 0 | | | 100 | |

**Figure 4. Detailed image for a customer's invoices**



**Situatia produselor stocate**

| CodProdus | DenProdusului Finit | CodGestiune | StocInitial |
|---|---|---|---|
| 10 | covor | g2 | 32 |

*Summary for 'CodProdus' = 10 (1 detail record)*

**Sum** 32

**Avg** 32

*Summary for 'CodProdus' = 10 (1 detail record)*

**Sum** 32

**Avg** 32

| CodProdus | DenProdusului Finit | CodGestiune | StocInitial |
|---|---|---|---|
| 11 | macheta | g1 | 20 |

*Summary for 'CodProdus' = 11 (1 detail record)*

**Sum** 20

**Avg** 20

*Summary for 'CodProdus' = 11 (1 detail record)*

**Sum** 20

**Avg** 20

**Figure 5. Different end situations displayed with a report**

## 4. CONCLUSIONS

A database management system such as Access, provides us with the software tools we need to organize that data in a flexible manner. It includes facilities to add, modify or delete data from the database, queries about the data stored in the database and produce reports summarizing selected contents.

In enterprise environment, Microsoft Access is particularly appropriate for meeting end-user database needs and for rapid application development. Microsoft Access is easy enough for end users to create their own queries, forms and reports, laying out fields and groupings, setting formats, etc. This capability allows professional developers, as well as end users, to develop a wide range of applications to fulfill the needs of an organization or commercial purpose.

In conclusion, effective database design can help the development team reduce overall development time and costs. Undertaking the process of database design and creating a data model helps the team better understand the user's requirements and thus enables them to build a system that is more reflective of the user's requirements and business rules. The act of performing database design is platform-independent so persons who use database systems other than SQL Server should also be able to benefit from these concepts.

## REFERENCES:

**[1]. Benaroch, M.; Kauffman, R. J**. (2000) *Justifying electronic banking network expansion using real option analysis*, MIS Quaterly; 24(2): pp.197-225

**[2]. Chen, P.P-S.** (1976) *The Entity Relationship Model – Towards a unified view of data*, ACM Transactions on Database Systems, 1(1):9-36

**[3]. Codd, E.F.** (1970) *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM, 13(6)

**[4]. Năstase, P., et al.** (2001) *Tehnologia bazelor de date. Access 2000*, Editura Economică, București

**[5]. Muntean, C.** (2003) *Crearea de aplicații Visual Basic 6.0 cu baze de date Access*, Editura Mirton, Timişoara

**[6]. Microsoft Office** http://en.wikipedia.org/wiki/Microsoft_Office [Accessed 16 April 2011]

**[7]. 2007 Office Tutorials** http://www.officetutorials.com/ [Accessed 16 April 2011]

**[8]. Classifying Relationships** http://www.referatele.com/referate/noi/informatica/classifying-relation14212441821.php [Accessed 25April 2011]