**07/05/2015**

**Mega.com**

**Project Report**

**ISE 305**

**Database Systems**

**Ahmet Cüneyd Tantuğ**

**ISE 313**

**WEB-Design & Management Tools**

**Tolga Ovatman**

**Group Mega.com**

**Mert Özsaydı 150110225**

**Efe Mert Yılmaz 150110231**

**Gülce Başar 150120201**

**Alper Özaydın 150110239**

May 2015

**1) Scope**

In 21th century the role of the cell phones can not be underestimated. We saw an opportunity to sell cell phone product in a growing market. That’s why we created Mega.com. In Mega.com, we designed a website for selling all the things about the phone such as batteries, phone covers, charging and connection cables, headphones, car accessories, screen protectors, phone stands, portable speakers, armbands etc. Mega.com comes from the first letters of the group member’s name which are Mert, Efe, Gulce and Alper. Mega.com users can order all the things about a phone in our e-store in a large scale. Users can see the products without logging in and they can sign up to the Mega.com for free. For giving an order to the site, users must have an account. Mega.com will send the products via their account information safely.

Mega.com’s database is used to store category information, city information, color information, manufacturer information, order information, product information, user information, type of telephone information and special information & stock about every product. For Web Design & Management part we use NetBeans IDE. Also for the database part we use Java DB. In our database we use integer, varchar, blob, double and date data types.

**1.1) Major Inputs**

– Product Issues

\*Information about every product is defined by Mega.com employees

– Financial Issues

\*Credit Card Number

\*Credit Card Security Number

\*Credit Card Expiration Date

– Security Issues

\*Usernames

\*Passwords (should be kept with cryptic version)

– Processing Functionalities:

\* Interface for ordering product

\*Showing the stock for every product and buy now option for if stock is valid

**1.2) Major Output**

-Mega.com order request

**1.3) Processing Functionalities**

|  |  |
| --- | --- |
| **Function** | **Priority** |
| User Login | Optional |
| User Checkout | Optional |
| User Information on Checkout | Essential |
| Credit Card Number | Essential |
| Credit Card CVC | Essential |
| Credit Card Expiration Date | Essential |

**Table 1: Processing Functionalities**

**1.4) User Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| **Req. No** | **Priority** | **Reference** | **Description** |
| R1 | Low | Costumer | Login the Mega.com |
| R2 | High | Costumer | User should login for ordering products |
| R3 | High | Costumer | User should give his/her credit card number |
| R4 | High | Costumer | User should give his/her credit card CVC |
| R5 | High | System | System take the order and send it |
| R6 | High | System | System automatically update the stock |

**Table 2: User Requirements**

**1.5) Milestones**

|  |  |  |
| --- | --- | --- |
| **Key Milestones** | **Start Date** | **End Date** |
| Project requirements planning | 15 March 2015 | 1 April 2015 |
| Planning and creating tables for database | 1 April 2015 | 15 April 2015 |
| Designing and coding website | 15 April 2015 | 20 April 2015 |
| Database and website connected | 20 April 2015 | 1 May 2015 |
| Test case & Bug fix | 1 May 2015 | 7 May 2015 |

**Table 3: Milestones**

**1.6) CRUD Operations**

SQL Insert Operations

* Adding user information to User Table while signing up of a user to website
* Adding purchasing product to Order Table in database

SQL Update Operations

* Editing Information of user
* Updating stock number after a successful purchase

SQL Delete Operations

* Deleting a user

SQL Read Operations

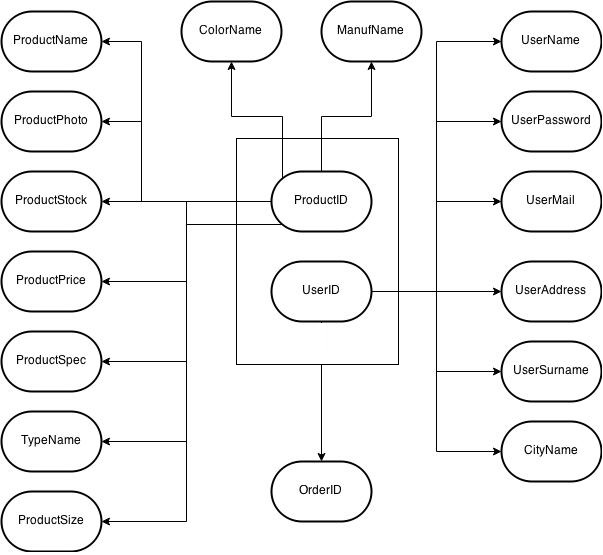
* Reading all product information (photos, price, name, size etc.) from database
* Reading all user information from database

**2) Functional Dependencies**

Functional dependency is a relationship that exists when one attribute uniquely determines another attribute. If R is a relation with attributes A and B, a functional dependency between the attributes is represented as A🡪B, which specifies B is functionally dependent on A.

Functional dependency in a database serves as a constraint between two sets of attributes. Defining functional dependency is a significant part of relational database design and contributes to aspect normalization.

* ProductID 🡪 ProductName
* ProductID 🡪 ProductPrice
* ProductID 🡪 ProductPhoto
* ProductID 🡪 ProductSize
* ProductID 🡪 ProductSpec
* ProductID 🡪 ProductStock
* ProductID 🡪 ColorName
* ProductID 🡪 ManufName
* ProductID 🡪 TypeName
* UserID 🡪 UserMail
* UserID 🡪 UserPassword
* UserID 🡪 UserName
* UserID 🡪 UserSurname
* UserID 🡪 UserAddress
* UserID 🡪 CityName
* {ProductID, UserID} 🡪 OrderID



**Table 4: Functional Dependencies**

**3) Normalization Steps**

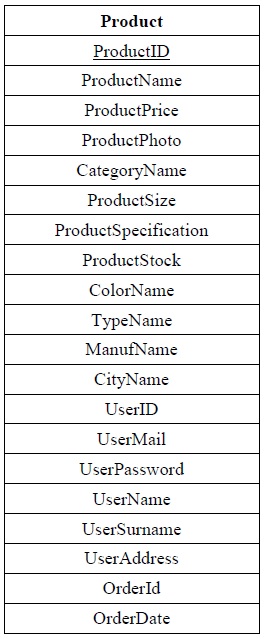
Normalization is the process of efficiently organizing data in a database. The purpose of normalization is to reduce data storage and to reduce data redundancy by making sure any given piece of data is stored only once.

For instance, if there are a lot of same city values in a table, programmers do not want to write them all. They create another table for cities and connect two tables with keys. This process makes effective database and programmer can save a lot of space after normalization.

**3.1) 1NF**

The main purpose of the first normalization form is eliminate repeating groups. The first rule dictates that we must not duplicate data within the same row of a table.

* There is no duplicate rows in the table.
* There is no duplicate columns in the table.
* No multi-valued attributes, like storing multiple phone number in a single column.
* All entries within a single column must be of the same data type.



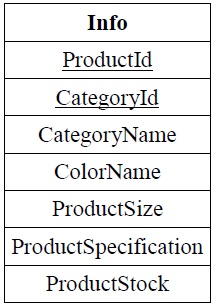
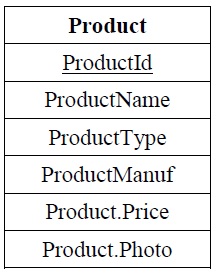
**Table 5: 1NF**

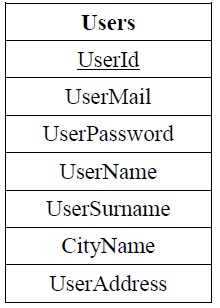
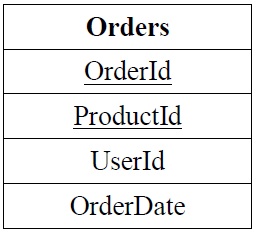
In the project, we use normalization to define and organize entities and attributes. In first normal form, we define all rows and collect attributes in a single table without any repetition.

**3.2) 2NF**

The main purpose of the second normalization form is eliminate redundant data. 2NF attempts to reduce the amount of redundant data in a table by extracting it, placing it in new table or tables and creating relationships between those tables.

* Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
* Create relationships between these new tables and their predecessors through the use of foreign keys.

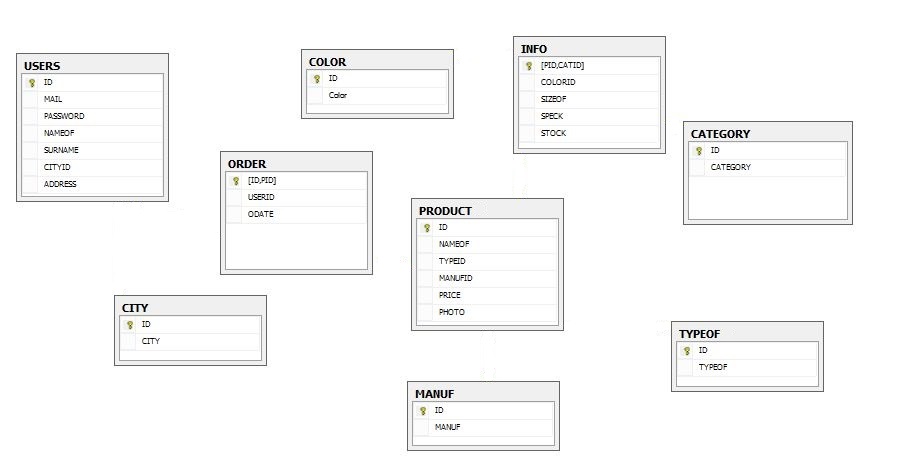
**Table 6: 2NF**

In second normal form, we split one table to four main tables and define a primary key /keys for each table. In Info table, there are two primary keys because any product can be in different categories at the same time. Also Order table has 2 primary keys because users can buy a lot of products in the same order number. This situation gives the chance of creating shopping carts in this project. The rest of them have only one primary key.

**3.3) 3NF**

The main purpose of the third normalization form is eliminate non-dependent columns. Third normal form (3NF) is a database principle that allows you to cleanly organize your tables by building upon the database normalization principles provided by 1NF and 2NF.

* Eliminate transient dependencies.
* Remove columns that are not fully dependent upon the primary key.



**Table 7: 3NF**

In third normal form, we separate 4 tables into 9 different tables. We define that color, city, manuf, typeof, and category tables should be in different table. We determine like that because of efficient memory management and organizing information easily in database.

Every color name has a unique integer id number because integer is 4 byte, using id in color table provide us more efficient memory. In addition, because color, city, manuf, typeof, and category have unique id, nobody can add meaningless color, manufacture name to database and organizing them will be easy.

**4) Entity Relationship Diagram**

ER models are represented by ER diagram. In project, we have 9 different entities (USERS, ORDER, CATEGORY, INFO, CITY, PRODUCT, COLOR, MANUF, and TYPEOF). Each entities have different attributes and all of these are in some relationship each other. In ER diagram, there are no many to many (M2M) because controlling and defining entities are not easy in entities and relationships diagram. There are 5 ‘one to many’ relationships and 3 ‘one to one’ relationships in ER diagram.

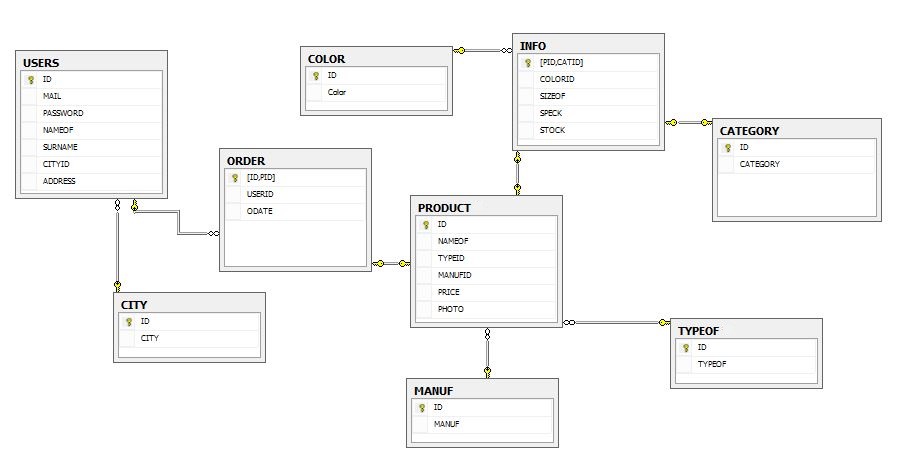
**One to One**

* Category 🡪 Info
* Info 🡪 Product
* Order 🡪 Product

**One to Many**

* Order 🡪 Users
* Product 🡪 Typeof
* Product 🡪 Manuf
* Info 🡪 Color
* Users 🡪 City

In User, category, city, product, orders, manuf, typeof, and color tables, ID is a primary key. In Info table, there are two different primary keys (PID and CATID) because any product can be in different categories at the same time. Also in Order table, there are two different primary keys (ID and PID) because in shopping cart, all products which is purchased need to be the same order number.



**Table 8: ER Diagram**

**5) Application Information**

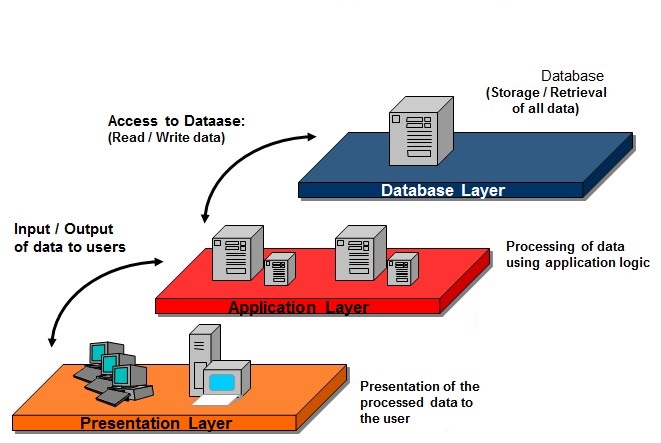
**5.1) Platform**

* Java : Our programming language and computing platform for Mega.com project
* Glassfish: Open-source application server
* Html: HTML stands for Hypertext Markup Language and it is used for creating websites. HTML is not a programming language but it is a markup language. It needs to be interpreted by web browser.
* CSS: CSS stands for Cascading Style Sheet. It is used for editing the look and format of a page written by a markup language.
* JDBC: It is Java database connectivity technology by Oracle. This is an API for Java to set connection between database and program.
* Derby: It is a relational database management system and it is open source. We choose Derby because it has small footprint and it has embedded JDBC driver that we can use in Java.
* NetBeans IDE: It is a software development platform developed by Oracle. In Mega.com we preferred to use NetBeans because it is free, open source and it has support for Java platforms.

**5.2) Architecture**

Mega.com has 3 layer application architecture. First layer is user interface in this layer we have a website for users to buy their products from our website. Second layer is business logic layer. Business logic essential to be shared through channels. Also we have a database layer in our Mega.com project. The whole thing need to be stored in our database.

First of all we designed our database for Mega.com project. While we are designing our database, we aim to store the clean data. Also, we aim to use low memory while we are designing Mega.com database. We create our tables for our future needs in this project. After designing and coding our database, we created our website. We used database first approach in this project.



**Table 9: Tier Client/Server Environment**