



DEPARTMENT OF COMPUTER  
ENGINEERING AND IT



AMIRKABIR UNIVERSITY  
OF TECHNOLOGY

**In the name of Allah**  
**Amirkabir University of Technology**  
**Department of Computer Engineering and IT**

**Database Design Project**  
**Online Store**

**Programmer: Amir.M Pirhosseinloo**

**Email: [amirphl@aut.ac.ir](mailto:amirphl@aut.ac.ir)**

**Github: amirphl**

**Linkedin: amirmohammad pirhosseinloo**

**Professor: Dr H Shahriyari**

Endurance again has four aspects: eagerness, fear, abstention (from the world) and anticipation (of death). So, whoever is eager for Paradise will ignore the passions; whoever fears the Fire (of Hell) will refrain from prohibited acts; whoever abstains from the world takes hardships lightly; and whoever anticipates death will hasten towards good deeds.

**Amir Al-Mu'minin 'Ali Ibn Abi Talib (Peace Be Upon Him)**

<b>Contents</b>	
<b>Abstract</b> .....	4
<b>Project Definition</b> .....	4
<b>ERD</b> .....	6
<b>Reduction to Schemas</b> .....	6
<b>Other Constraints</b> .....	7
<b>Procedurs</b> .....	7
<b>Triggers</b> .....	8
<b>Fake Data</b> .....	8
<b>Testing</b> .....	8
<b>Query Results</b> .....	8
<b>Conclusion</b> .....	11

## 1. Abstract

In this paper, I describe my activities for creating the database of an online store.

Store contains below entities:

- Customers
- Temporary Customers(unregistered)
- Shops
- Products
- Transmitters
- Supporters
- Operators

project parts:

- I. Extracting relationships between entities according to requirement specification to create Entity Relationship Diagram(ERD) (here project definition is requirement specification)
- II. Converting relationships to schemas
- III. Extracting constraints between entities and relationships
- IV. Converting schemas and relationships to executable code
- V. Converting constraints to triggers and procedures (I didn't use any Function while it is better to use Functions instead of trigger due to performance issues.)
- VI. Preparing fake data for database
- VII. Testing
  - a. verification testing
  - b. validation testing
- I. Performing queries according to project definition
- II. Conclusion

Codes and other files are available at:

[https://github.com/amirphl/Database\\_Design\\_Project/](https://github.com/amirphl/Database_Design_Project/)

## 2. Project Definition

Entities attributes are present in the code on

[https://github.com/amirphl/Database\\_Design\\_Project/tree/master/src](https://github.com/amirphl/Database_Design_Project/tree/master/src)

So I describe only conditions and constraints:

- **Customer**
  - Each customer may have many addresses and many phone numbers.
  - Customer can charge his account credit.
  - Customer password is stored in hashed format.
  - Each customer can edit only his account and can't change his username.
- **Temporary Customer**
  - No need to register for purchase, just provides a valid email.
  - Each one has exactly one address and one phone number.
  - The payment is online.

- There is no credit.
- There is no account charging because there is no credit.
- **Shop**
  - Each one is opened between start\_time and end\_time.
  - If an order is recorded out of time between start\_time and end\_time, it is being stored in database but the status of order is 'rejected'. The credit will not be decreased and no item will be sent to customer.
- **Supporter and Operator**
  - According to definition, Supporters and Operators do nothing in the Shop just like the Ostrich algorithm.
- **Transmitter**
  - Transmits the items purchased with someone to his home.
  - When transmitter delivers order (items) to customer, two occurrences happen:
    - status of the order changes to 'done'.
    - status of the transmitter changes to 'free'.
  - After delivering order to customer, 5 percent of price of order is granted to the transmitter.
- **Order**
  - Address of order is one of the addresses of Customer or the only address of Temporary Customer.
  - Phone number of order is one of the phone numbers of Customer or the only phone number of Temporary Customer.
  - Customers can pay for orders online or offline while Temporary Customers can only pay for order online.
  - After accepting order (in other words, status = 'accepted' or 'sending'), number of products in shop decreases.
- **Product**
  - Everything is visible in the code.

### some points:

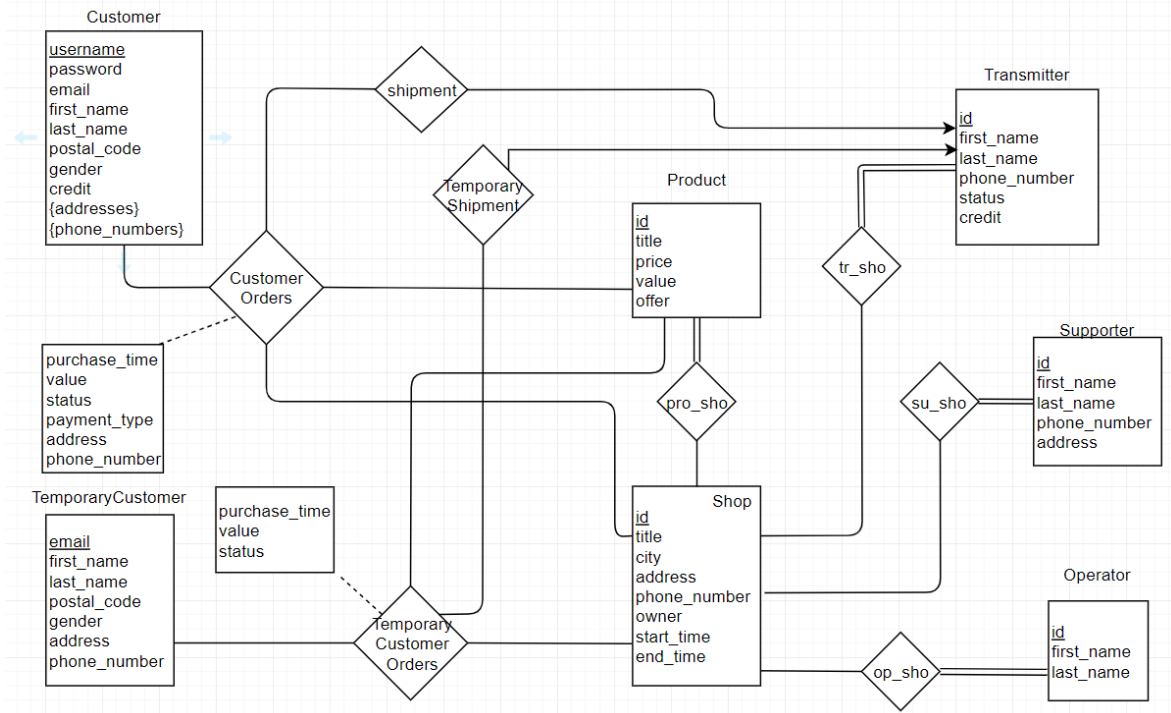
- Create tables for logging changes being made on tables.
- Avoid redundancy in data.
- Write optimum queries.

### queries:

- Find 5 top most products of each shop.
- Find phone number of customers (Customers + Temporary Customers) who their orders are rejected.
- Find difference between average price of Customers orders and average price of Temporary Customers orders.
- Find transmitter who average price of his shipments are more than others.

- Find shop which has most hours of work.

### 3. ERD



[tool for drawing ERD: [www.draw.io](http://www.draw.io) .thanks to draw.io]

This is the ERD of Online Store. Log tables are not shown here. There are 4 log tables:

- UpdateCustomerLog
- UpdateTransmitterLog
- UpdateCustomerOrderLog
- UpdateTemporaryCustomerOrderLog

### 4. Reduction to Schemas

#### a. Customer

- {addresses} is reduced to schema 'CustomerAddresses'.
- {phone\_numbers} is reduced to schema 'CustomerPhoneNumbers'.

#### b. CustomerOrders

'CustomerOrders' relationship is reduced to schema 'CustomerOrders'. It contains primary keys of Customer, Shop, Product in addition to purchase\_time as primary key because time is periodic. Other descriptive attributes are added according to ERD.

#### c. TemporaryCustomers

Like 'CustomerOrders' relationship.

#### d. pro\_sho

This relationship must be reduced to a schema with primary keys of Product and Shop, but because of total participation of Product in relation, just primary key of Shop is added to the Product entity. There is no null value in shopId attribute in Product because of total participation, so no redundancy exists.

**e. shipment**

This relationship is many to one from CustomerOrders to transmitters, so it must be initiated as a schema with primary key of CustomerOrders, but this is not a good design, not all orders are connected to transmitters, for example, orders with status = 'rejected', so number of fields with null values increases. To solving this problem, a new schema is created called Shipment with primary keys of CustomerOrders. This design solves redundancy problem.

**f. TemporaryShipment**

Same as Shipment relationship.

**g. op\_sho**

This relationship is many to many between Operator and Shop, so it must be initiated as a schema with primary keys of Operator and Shop, but because the participation is total from Operator to Shop, the relation can be deleted and primary key of Shop added to Operator.

**h. Supporter**

Same as Operator relationship.

**i. Transmitter**

Same as Operator relationship.

## 5. Other Constraints

- a. When a customer wants to purchase  $x$  number of product  $y$  from shop  $z$ , there must be exists more than  $x$  number of  $y$  in  $z$ , otherwise, order will be stored in database as 'rejected', no money will be reduced from customer credit and no transmitter will move anything to customer.
- b. Customer must have enough credit  $c$  to buy  $x$  number of product  $y$  with price  $p$  and offer  $f$  in other words,  $c \geq x * p * (1 - f)$

## 6. Procedures

- a. **add\_customer**: Before adding customer, hashes the password with sha1 built in method.
- b. **add\_order\_by\_customer**: Acts like trigger 'add\_order\_by\_customer'.
- c. **add\_order\_by\_temporary\_customer**: Acts like trigger 'add\_order\_by\_temporary\_customer'.
- d. **deliver\_to\_customer**: Changes status of order to 'done' and status of transmitter to 'free', then increases credit of transmitter by 5 percent of price of order.
- e. **deliver\_to\_temporary\_customer**: Same as deliver\_to\_customer, only one difference is that owner of product is TemporaryCustomer not Customer.
- f. **charge\_account**: Increases credit of customer.

## 7. Triggers

- a. **add\_order\_by\_customer**: Checks customer credit, existence of enough number of products to be delivered to customer, check hours of work of shop, rejecting or accepting order.
- b. **deliver\_to\_transmitter**: Checks existence of a transmitter in shop to carrying items to customer address. Sending items if some transmitter exists, else just accept order (someone like Operator or Supporter must check database for such situations each 24 hours (for example) and do appropriate actions to solve these issues).
- c. **add\_order\_by\_temporary\_customer**: Like 'add\_order\_by\_customer' trigger, except that there is no credit here and user must pay for product online.
- d. **deliver\_temporary\_customer\_order\_to\_transmitter**: Same as 'deliver\_to\_transmitter trigger'.
- e. **hash\_password**: Hash customer's password with [sha1](#) algorithm.
- f. **log\_update\_on\_customers**: Log changes made in Customer table. For example, changing password of a customer.
- g. **log\_update\_on\_transmitters**: Log changes made in Transmitters table. For example, change status to 'sending' or 'free'.
- h. **log\_update\_on\_customer\_orders**; Log changes made in CustomerOrders table. For example, change status from 'sending' to 'done' or 'accepted' to 'sending'.
- i. **log\_update\_on\_temporary\_customer\_orders** Same as log\_update\_on\_customer\_orders trigger.

## 8. Fake Data

Fake data is generated with help of <http://filldb.info>. but because lack of support for some types of foreign key, some changes made manually. Fake data is available at 'fake\_data' folder.

## 9. Testing

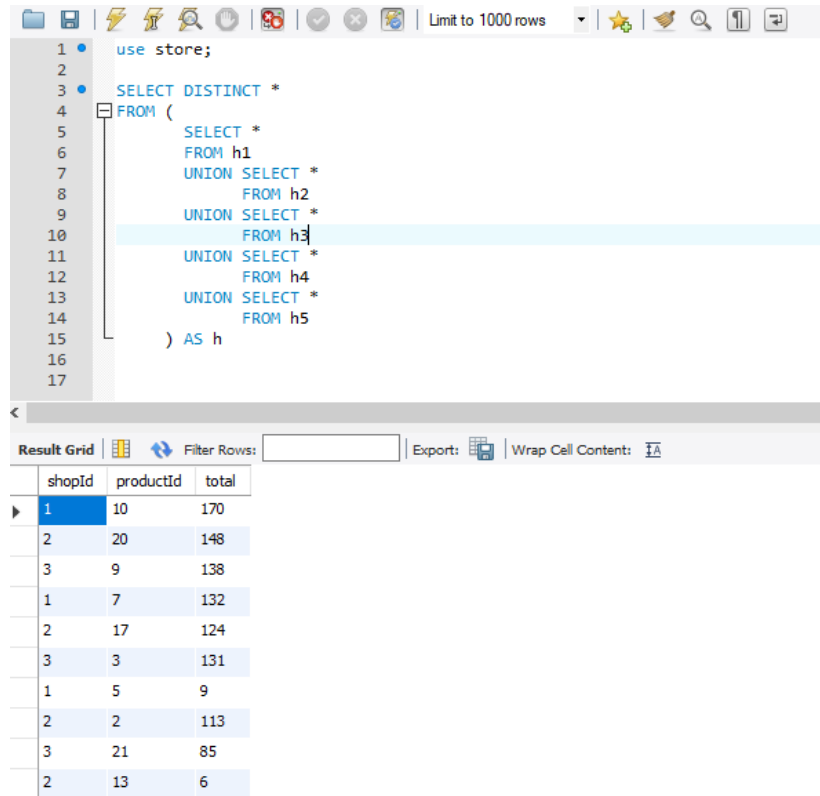
Testing is done with iteratively reviewing project definition. For example, one of bugs was making transmitter ID as primary key for table Shipment which was an incorrect decision. The bug fixed by removing it in one of the last commits.

## 10. Query Results

Results images are available at 'query\_results' folder. Also results are shown by ascending order below:



### a) query 1



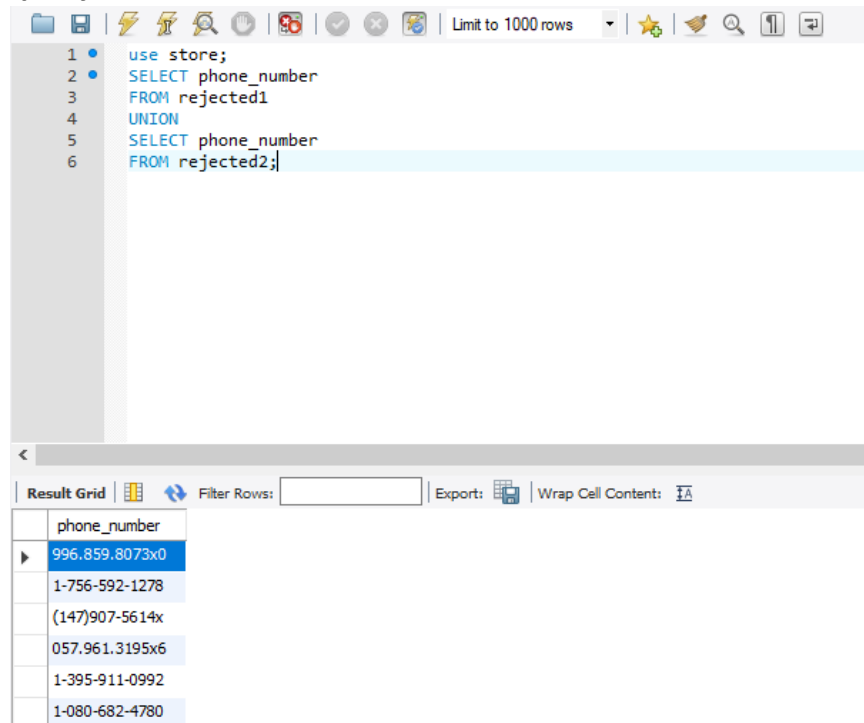
```

1 use store;
2
3 SELECT DISTINCT *
4 FROM (
5     SELECT *
6     FROM h1
7     UNION SELECT *
8     FROM h2
9     UNION SELECT *
10    FROM h3
11    UNION SELECT *
12    FROM h4
13    UNION SELECT *
14    FROM h5
15 ) AS h

```

	shopId	productId	total
1	10	170	
2	20	148	
3	9	138	
1	7	132	
2	17	124	
3	3	131	
1	5	9	
2	2	113	
3	21	85	
2	13	6	

### b) query 2



```

1 use store;
2 SELECT phone_number
3 FROM rejected1
4 UNION
5 SELECT phone_number
6 FROM rejected2;

```

phone_number
996.859.8073x0
1-756-592-1278
(147)907-5614x
057.961.3195x6
1-395-911-0992
1-080-682-4780

### c) query 3

```

1 • use store;
2 • SELECT T.average - C.average
3 FROM temporarycustomersaverage AS T, customersaverage AS C

```

Result Grid

T.average - C.average
4377.3956

Filter Rows:  Export: Wrap Cell Content:

#### d) query 4

```

1 • USE store;
2 • SELECT
3     id,
4     average
5 FROM transmitteraverageall
6 WHERE average >= ALL (SELECT R.average
7                       FROM transmitteraverageall AS R)

```

Limit to 1000 rows

Result Grid

id	average
10	76077.0000

Filter Rows:  Export: Wrap Cell Content:

### e) query 5

```
1 use store;  
2 SELECT *  
3 FROM shop  
4 WHERE end_time - start_time >= ALL (SELECT SH.end_time - SH.start_time  
5                                     FROM shop AS SH)
```

Result Grid								
Filter Rows:								
	id	title	city	address	phone_number	owner	start_time	end_time
▶	2	quia	East Ottilie	6099 Annamarie Underpass Apt. 29...	340-004-9668x3	sunt	01:58:10	17:51:09
*		NULL	NULL	NULL	NULL	NULL	NULL	NULL

## 11. Conclusion

In this project an ERD for an Online Store implemented, constraints and conditions satisfied as triggers and procedures. Fake data generated by <http://filldb.info> and ERD drawn with [www.draw.io](http://www.draw.io) but still one thing remained: normalization, the design is in BCNF but I have not checked is it in 4NF or not.