In order to implement our application, we have to produce a relational database using MySQL. The following pages specify the steps we have taken to create the database, from the conceptual design to the logical scheme and the definitive implementation of the code. The database was designed by analysing the use cases and the class diagram, in order for the application to be as fast and efficient as possible.

**CONCEPTUAL DESIGN**

**Entities**

Considering the application requirements, we identify the following entities:

* **USER:** represent the application user
* **EMPLOYEE**: represents the different employees that work within the company.
* **TEAM:** the various employees work in teams.
* **PRODUCT:** the different products that are assembled in the company by the teams.
* **COMPONENT:** each product is made by different components
* **SUPPLIER:** represents the various suppliers from which the company buys the components to assemble the products.
* **CUSTOMER:** represents the company clients. Clients can buy products.

**Attributes and identifiers**

The following table shows the various attributes related to the entities. Each attribute is to be considered having (1,1) cardinality with the entity, unless otherwise specified.

|  |  |
| --- | --- |
| **ENTITY** | **ATTRIBUTES** |
| USER | IDuser, name, surname, username, password, mail |
| EMPLOYEE | IDemployee, team, role, salary |
| TEAM | IDteam, location |
| PRODUCT | IDproduct, productPrice, productName, productDescription, productAvailability |
| COMPONENT | IDcomponent, componentName, componenetAvailability, componentDescription |
| SUPPLIER | IDsupplier, companyName, supplierMail |
| CUSTOMER | IDcostumer, address |

The entity USER is uniquely identified by the IDuser attribute, which is unique for each user. We also report the user name, surname, username, password and mail. Username are not necessarily unique for the users.

The entity EMPLOYEE is uniquely identified by the iDemployee attribute, which is unique for each employee. The other attributes are the employee’s role in his team and the salary.

The entity TEAM is uniquely identified by the IDteam attribute, which is unique for each team. We also have an attribute location which represents the physical location where the team is currently working.

The entity PRODUCT is uniquely identified by the iDproduct attribute, which is unique for each product. The other attributes are the product price, the product name, a brief description of the product and its availability. The availability of a product is a Boolean value that represents if the product is available to be purchased by a customer.

The entity COMPONENT is uniquely identified by the iDcomponent attribute, which is unique for each component. The other attributes are the component name, its price and the availability, which represents if the component is ready to be used by a team to assemble a product. Availability is a Boolean value

The entity SUPPLIER is uniquely identified by the iDSupplier attribute. We also have an attribute for the company for which the supplier works and its mail address.

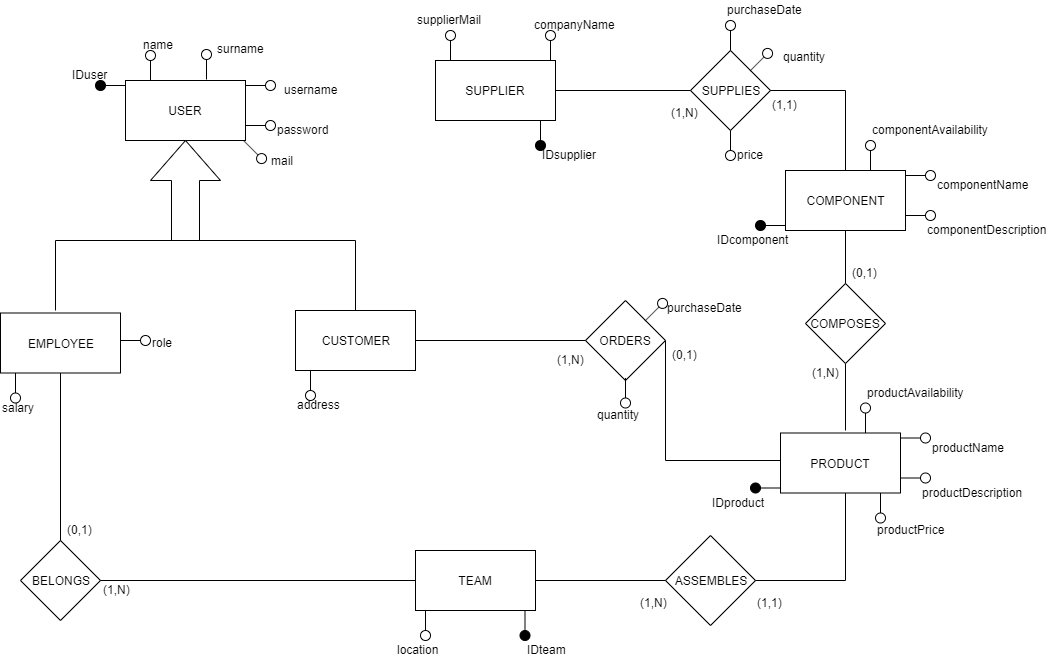
The entity COSTUMER is uniquely identified by the iDCostumer attribute. We also report the customer’s address.

**RELATIONS BETWEEN ENTITIES**

The following table shows the different relations between the entities in the database. Each relation is matched with the appropriate cardinality. We propose a brief explanation for each match.

|  |  |  |  |
| --- | --- | --- | --- |
| **RELATIONS** | **ENTITY A**  **(Cardinality with A)** | **ENTITY B**  **(Cardinality with B)** | **MOTIVATION** |
| BELONGS | EMPLOYEE  (0,1) | TEAM  (1,N) | An employee can work in up to one team, and a team needs at least one member |
| ASSEMBLES | TEAM  (1,N) | PRODUCT  (1,1) | Each team can work at same time on one or more products. A product can be assembled by only one team |
| ORDERS | CUSTOMER  (1,N) | PRODUCT  (0,1) | Each product can be purchased by only one client. A customer can of course by more than one product |
| COMPOSES | PRODUCT  (1,N) | COMPONENT  (0,1) | A product can be made using one or more components. A component can be used in only one product. |
| SUPPLIES | COMPONENT  (1,1) | SUPPLIER  (1,N) | A supplier can sell more than one component |

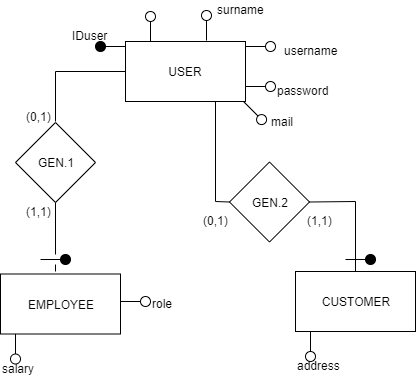
The ER-Diagram appears as follows:



**Generalization analysis**

As we can see, the entity USER is actually a generalization of the entities EMPLOYEE and CUSTOMER. We decided to solve using the “no aggregation” approach. We maintain in the scheme both the father and the children, so that we still have three entities: USER, EMPLOYEE and CUSTOMER. This approach may result in an increase in the number of access, but doesn’t introduce NULL values in the database. The “no aggregation” approach is preferable when we want to distinguish the occurrences on the father entity (USER) and the children entities (EMPLOYEE and CUSTOMER).

The following picture show the resolved generalization



**Operations on the Database**

Considering the application requirements we defined before, the following operations are expected:

1. ADD USER

This operation allows the administrator to add a user to the database

Expected frequency: 3 times a day

1. REMOVE USER

This operation allows the administrator to remove a user from the database

Expected frequency: 1 times a day

1. UPDATE SALARY

This operation allows the administrator to modify an employee’ salary

Expected frequency: 1 times a day

1. VISUALIZE ORDER STATUS

This operation allows a customer to see the status of his orders

Expected frequency: 100 times a day

1. VISUALIZE PRODUCTS

This operation allows a customer to see the list of the available products

Expected frequency: 200 times a day

1. ADD ORDER

This operation allows a customer to buy a product from the ones available

Expected frequency: 20 times a day

1. VISUALIZE TEAM PRODUCTS

This operation allows a team leader to see the different product his team it’s working on.

Expected frequency: 60 times a day

1. VISUALIZE COMPONENTS

This operation allows a team leader to see the components of his team

Expected frequency: 6 times a day.

1. CHANGE AVAILABILITY

This operation allow a team leader to change the availability of a product

Expected frequency: 60 times a day

**Volume table**

In the following table we report the number of instances from the various entities and relations in the E-R diagram.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **E/R** | **Number of instances** | **Motivation** |
| USER | E | 100 | Initial hypothesis |
| GEN.1 | R | 30 | 30% of the users are employees |
| EMPLOYEE | E | 30 | (1,1) cardinality with Gen.1 |
| GEN.2 | R | 70 | 70% of the users are customers |
| CUSTOMER | E | 70 | (1,1) cardinality with Gen.2 |
| BELONGS | R | 70 | (0,1) cardinality with Employee |
| TEAM | E | 6 | There are 6 teams in the company |
| ASSEMBLES | R | 100 | Each team assembles 100 products on average |
| PRODUCT | E | 600 | (1,1) cardinality with Assembles |
| ORDERS | R | 350 | On average each customers make 5 orders (5x70=350) |
| COMPOSES | R | 3000 | On average 5 components are needed to assemble a product |
| COMPONENT | E | 3000 | (0,1) cardinality with Composes |
| SUPPLIES | R | 3000 | (1,1) cardinality with Component |
| SUPPLIER | E | 10 | Each supplier can supply 300 components (3000/300 = 10) |

**Operations analysis and redundancy introduction**

Let’s now analyse operation 8 (VISUALIZE COMPONENTS) in terms of the number of accesses required each time the operation is called. We would like to introduce a redundant attribute called “team” for the entity EMPLOYEE. The “team” represents the team where the employee is currently working.

Operation 8: VISUALIZE COMPONENTS

This operation allows a team leader to see the components of his team

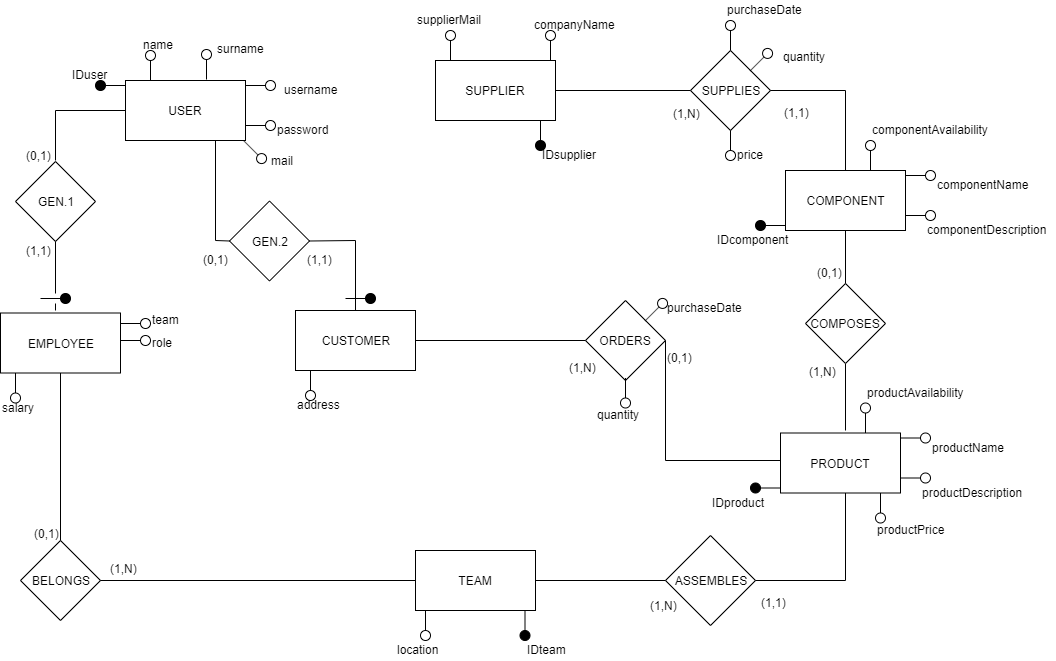
|  |  |  |
| --- | --- | --- |
| **Without redundancy “team”** | | |
| **Elementary operations number** | **Type of operation (read/write)** | **Motivation** |
| 1 | R | Access to Employee |
| 1 | R | Access to Team |
| 70x6 = 420 | R | Join of the entities Employee and Team |
| 422 | Total number of operations | |
| 422x6 = 2532 | Total number of operations in a day | |

|  |  |  |
| --- | --- | --- |
| **With redundancy “team”** | | |
| **Elementary operations number** | **Type of operation (read/write)** | **Motivation** |
| 70 | R | Access to Employee. I can directly read in which team each employee is working |
| 70 | Total number of operations | |
| 70x6 = 420 | Total number of operations in a day | |

Since the attribute “team” makes the operations more than 6 times faster, we decide to add the redundancy in the database.

**Final ER diagram**

The definitive version of the ER diagram appears as follow:



**LOGICAL DESIGN**

**Logical scheme**

Since we have the final ER diagram, we can now translate it into tables obtaining the following logical scheme:

* USER(IDuser, name, surname, username, password, mail)
* EMPLOYEE(IDemployee, salary, role, team)
* CUSTOMER(IDcustomer, address)
* TEAM(IDteam, location)
* ASSEMBLES(team, product)
* PRODUCT(IDproduct, productName, productDescription, productPrice, productAvailability)
* COMPONENT(IDcomponent, componentName, componentDescription, componentAvailability)
* COMPOSES(component,product)
* ORDERS(customer, product, purchaseDate, quantity, status)
* SUPPLIER(IDsupplier, companyName, supplierMail)
* SUPPLIES(component, supplier, quantity, purchaseDate, price)

Referential integrity constraint:

* There is a referential integrity constraint between the attribute IDuser of the table USER and the attribute IDemployee of the table EMPLOYEE
* There is a referential integrity constraint between the attribute IDuser of the table USER and the attribute IDcustomer of the table CUSTOMER
* There is a referential integrity constraint between the attribute IDteam of the table EMPLOYEE and the attribute IDteam of the table TEAM
* There is a referential integrity constraint between the attribute IDteam of the table TEAM and the attribute team of the table ASSEMBLES
* There is a referential integrity constraint between the attribute IDproduct of the table PRODUCT and the attribute product of the table ASSEMBLES
* There is a referential integrity constraint between the attribute IDcomponent of the table COMPONENT and the attribute component of the table COMPOSES
* There is a referential integrity constraint between the attribute IDproduct of the table PRODUCT and the attribute product of the table COMPOSES
* There is a referential integrity constraint between the attribute IDcustomer of the table CUSTOMER and the attribute customer of the table ORDERS
* There is a referential integrity constraint between the attribute IDproduct of the table PRODUCT and the attribute product of the table ORDERS
* There is a referential integrity constraint between the attribute IDcomponent of the table COMPONENT and the attribute component of the table SUPPLIES
* There is a referential integrity constraint between the attribute IDsupplier of the table SUPPLIER and the attribute supplier of the table SUPPLIES

**Normalization**

All the tables in our design respect the Boyce-Codd normal form because for every one of the dependencies X🡪Y in a table it’s true that X is a superkey for the table. There are not redundancy based on functional dependency.

**Database implementation**

Terminated our theoretical study for the database, we implemented the correspond tables on MySQL.