UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Foundations of Databases A.Y. 2021-2022
Homework 2 – Conceptual and Logical Design

Master Degree in Computer Engineering
Master Degree in Cybersecurity
Master Degree in ICT for Internet and Multimedia

Deadline: November 26, 2021

Team acronym	TAGMS	
Last Name	**First Name**	**Student Number**
Arslan	Tunahan	2023640
Basso	Marco	2005796
Cimarosto	Pietro	2027173
Collado	Martin	2039907
Esposito	Vittorio	2005795
Giuliani	Amedeo	2005797
Quiroz	Giannina	2041427
Zanini	Samuele	2019038

Conceptual Design

Variations to the Requirement Analysis

An employee must necessarily have one and only one role. A role is typically associated with multiple employees but may not have any. For example, the "intern" role can be created but without having hired anyone in that position yet.

An employee must work in at least one department. Each department can have multiple workers, but none if the department is new and newly created.

A manager is an employee with role equals to "Manager". Only they are allowed to stipulate one or more contracts with many suppliers. A supplier can stipulate many contracts with the manager as well, but a new supplier can also not have stipulated a contract yet. Our system, therefore, is able to also store information relating to suppliers with whom the company has not yet entered into agreements.

Each contract specifies one or more items (e.g., ingredients, packaging materials, etc.) and the respective quantity. An item can be specified in at least one contract, so it can be provided by many different suppliers. Each item belongs to only one item category, and to an item category can belong zero or many items. The stock quantity of each item is tracked. It will therefore be increased upon receipt of the items, after purchase from one or more suppliers, and decreased upon the preparation of lots of products that are made up of the items. The delivery date of the items is specified in each contract, so that the respective quantity in stock is automatically updated only upon arrival of the goods. The company does not use periodic delivery contracts, so every time it becomes necessary to purchase certain ingredients or packaging material, a new contract is signed with the supplier.

A product, which is a finished good ready to be sold, is made up of one or more items (e.g., one glass bottle, a hundred grams of sugar, a hundred milliliters of water, etc.) with the respective quantities. For example, a product called "Coke J" can consist of one aluminum can, 50ml of water, 10g of sugar, etc. Another product, called "Coke B" for example, may have the same ingredients as the previous example but can be packaged with a glass bottle. The expiration date of a product is specified in the various lots (that include that particular product) and it may differ in each lot. The stock quantity of a product is not explicitly specified, but it can be obtained by checking the specified product quantities in each lot not yet sold or shipped. An item can be utilized in many products (also none if, for example, the item is brand new). Each product belongs to one and only one product category, that is used to distinguish them. To a product category can belong zero or many products. The company decides the value of the price increase according to company policies. This increase ("Price_increase") must take on a value greater than or equal to 1. The price of a product is calculated as a multiplication between the cost of production and the price increase. The expiration date of the ingredients (items) is not tracked as the company guarantees to keep them stored for a short period of time because the ingredients are used shortly after their purchase and a FIFO policy is being implemented. In addition, keeping a maintaining level of stock and having a "Minimum_quantity" attribute is not needed because products are being produced by demand which avoids excess stocks in the warehouse.

A package is composed of one or more packaging materials (i.e., an item used for packaging, such as a box, a meter of plastic tape, a kilogram of polystyrene, etc.) with the respective quantities. For example, a package named "PK1" can consist of 4 boxes of dimensions 30cm x 30cm x 10cm, 2 meters of plastic tape and 200 g of polystyrene. A "PK2" package can consist of 6 boxes of dimensions 30cm x 30cm x 10cm, 4 meters of plastic tape and 300 g of polystyrene. An item (e.g., packaging material in this case) can be utilized in many packages (also none if, for example, the package is brand new and not yet used). Each package belongs to one and only one package category, that is used to distinguish them. To a package category can belong zero or many packages. Also, there is no "Minimum_quantity" attribute because packages are received in bulk quantities

and their shelf life are longer compared to items and products which makes it manageable without the need of maintaining level.

In a lot there can be stocked a certain amount of only a product and a certain amount of only a package. The quantity of products in each lot depends both on the dimensions of the package of the individual product (e.g., bottle of glass) and on the features of the package (in particular, the size of the box and the number of boxes that make up the package). Each type of product can be stocked in many lots (in none if, for example, the product is brand new) and the same holds true for packages. The price of a specific lot ("Lot_price") is calculated as the multiplication of the price of the product (stored in the lot) multiplied by its quantity. It is important to underline that the "Lot_price" attribute cannot be derived through an online operation as the fields on which it depends ("Production_Cost" and "Price_increase") can be modified over time and therefore give discordant results at different times. Each lot is also characterized by an expiration date. As some lots may be produced in advance to reduce lead times, some of them may not sell on time and therefore expire. The data analyst will perform half-yearly analysis in this regard to reduce waste. When a lot is produced, the company specifies the current price and VAT. The cost of the packages is not explicitly charged to the customer. A discount can also be associated with each lot. The company decides the total discount to apply to a specific lot. This discount expresses a percentage and is a number between 0 and 100. Furthermore, the company is able to take into account changes in VAT.

The "Quantity" attribute of the "Item" entity is derived. When a new contract is signed, in the delivery date (specified by the attribute "Delivery_date" of "Contract") the quantity of each item in the contract is incremented accordingly based on the "Purchased_Quantity" attribute in the relationship "Specify". When a new lot is inserted in the system, each quantity of each item, that constitute the product in the lot, is decremented by a number equal to the multiplications between "Product_quantity" in "Stocked" and "Quantity" in "Made_up_of(1)". The same applies to "Package" entity where the decreasing factor is calculated from the product between "Package_quantity" and "Quantity" in "Made_up_of (2). Immediately after this, for each item involved, the "Quantity" attribute is compared with the "Minimum_quantity" attribute so a manager will get notified to avoid shortages. Furthermore, the worker, before processing the order, verifies that there are sufficient quantities of items to satisfy the request.

The customer decides with the seller regarding the products to be bought. The salesman, then, after communicating the products (with respective quantity) to the warehouse worker, will place the order only when all the lots included in the order are ready. A seller can place zero or many orders for a customer, so a customer can make many orders (none if, for example, the customer is new). An order can be placed by only one salesman for only one customer (i.e., an order for a customer cannot be placed by two or more sellers, but by only one of them). An order includes one or more lots. Each lot can be included by only one order (none if the lot is produced in advanced and waiting to be ordered). The invoice will be automatically generated by the application linked to the system as soon as the order is placed. The total net amount of the order ("Net_price") must be calculated as the sum, for each lot i included, of the Lot_price_i * (1 - Lot_discount_i / 100). The total taxes are calculated as the sum, for each lot i, of the Lot_price_i * VAT_i / 100. When the order is ready, a worker will ship it: a worker can ship zero or many orders, and an order can be shipped by at most one worker (none if the order is waiting to be shipped). The cancellation and modification of the order is not accepted since the goods can be produced on commission and the company wants to minimize the waste caused by the expiry of the products. The customer, which is a business, must necessarily pay within 60 days and can be informed about the status of the order either by contacting the seller. Furthermore, through the tracking number received by email, the customer can monitor the shipping.

Entity-Relationship Schema

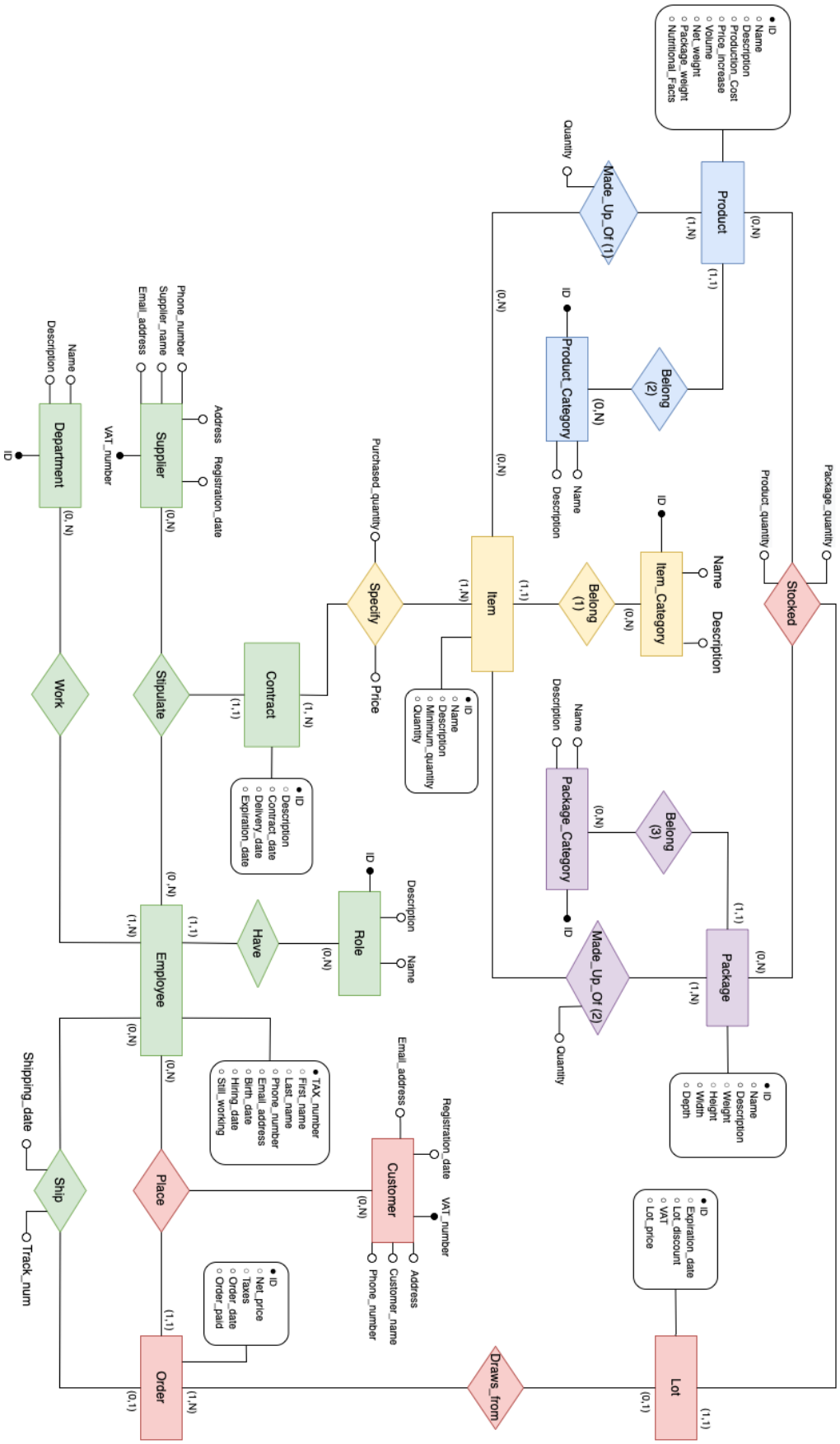

Figure 1: Entity-Relationship schema.

Data Dictionary

Entities Table

Entity	Description	Attributes	Identifier
Employee	Represents data of an employee who works in the company and needs access to the system	• TAX_number: TAX code of the employee, text • First_name: name of the employee, text • Last_name: surname of the employee, text • Phone_number: phone number (prefix included) of the employee, text • Email_address: email address of the employee, text • Birth_date: birthdate of the employee, Datetime • Hiring_date: hiring date of the employee, Datetime • Still_working: flag used to know if employee is still working for the company, boolean	TAX_number
Role	Represents data on the role of employees who work in the company	• ID: role identifier, serial • Name: name of the role, text • Description: technical description of the role, text	ID
Department	Represents data on the departments in which employees work	• ID: department identifier of the company, serial • Name: name of the department, text • Description: description of the department's function, text	ID

Customer	Represents data about a customer of the company	• VAT_number: VAT number of the customer, int • Customer_name: name of the customer, text • Phone_number: phone number (prefix included) of the customer, text • Address: billing address of the customer, text • Email_address: email address of the customer, text • Registration_date: customer registration date in the database, Datetime	VAT_number
Contract	Represents data about a contract stipulated between a supplier and a manager for the supply of items	• ID: contract identifier, serial • Description: description of the contract, text • Contract_date: date of signature of the contract with the supplier, Datetime • Delivery_date: expected date of delivery of the goods, Datetime • Expiration_date: expiration date of the contract with the supplier, Datetime	ID

Supplier	Represents data about a supplier of the company	• VAT_number: VAT number of the supplier company, int • Supplier_name: name of the supplier company, text • Phone_number: phone number (prefix included) of the supplier company, text • Email_address: email address of the supplier company, text • Address: address of the supplier company, text • Registration_date: recording date of the supplier company, Datetime	VAT_number
Order	Represents the order placed by a salesman for a customer	• ID: order identifier, serial • Order_date: date in which the order has been processed, Datetime • Order_paid: status of the payment, boolean • Net_price: total net amount including discount (VAT excluded), float • Taxes: total amount of taxes to be payed, float	ID

Lot	Represents a lot in the inventory of the company, containing final products and packaging	• ID: lot identifier, serial • Expiration_date: expiration date of the included products, Datetime • Lot_price: lot price (without the discount) at the time of sale, float • Lot_discount: percentage of discount of a lot, int • VAT: value added tax (percentage) at the time of sale, int	ID
Product	Represents the final product that is marketed	• ID: product identifier, serial • Name: name of the product, text • Description: description of the product, text • Nutritional_Facts: description of nutritional facts of the specific product, text • Volume: volume of the product in milliliters, int • Net_weight: net weight of the product in grams, int • Package_weight: package weight of the product in grams, int • Production_cost: cost of the production for a product, float • Price_increase: price increase factor, float	ID

Item	Represents materials provided by suppliers	• ID: identifier of the item, serial • Name: name of the item, text • Description: description of the item, text • Quantity: stock quantity of the item (automatically updates), int • Minimum_quantity: minimum stock quantity of the item, int	ID
Package	Represents packaging of finished products which are made up of boxes, tapes, and other packaging materials	• ID: identifier of the package, serial • Name: name of the package, text • Description: description of the package, text • Weight: weight dimension of the package in grams, int • Height: height dimension of the package in centimeters, int • Width: width dimension of the package in centimeters, int • Depth: depth dimension of the package in centimeters, int	ID
Product_Category	Represents the category of a product	• ID: identifier of the product category, serial • Name: name of the product category, text • Description: description of the product category, text	ID

Item_Category	Represents the category of an item	• ID: identifier of the item category, serial • Name: name of the item category, text • Description: description of the item category, text	ID
Package_Category	Represents the category of a package	• ID: identifier of the package category, serial • Name: name of the package category, text • Description: description of the package category, text	ID

Relationships Table

Relationship	Description	Component Entities	Attributes
Have	Relates each employee to a role	• Employee (1,1) • Role (0,N)	None
Work	Assigns each employee to a department	• Employee (1,N) • Department (0,N)	None
Stipulate	Links the supplier with the company and the contract stipulated	• Supplier (0,N) • Employee (0,N) • Contract (1,1)	None
Place	Links the order made by the employee	• Employee (0,N) • Order (1,1) • Customer (0,N)	None

Ships	Relates the employee shipping the order with the order itself and the shipment details	• Employee (0,N) • Order (0,1)	• Track_num: tracking code of the shipment provided by the external shipment service, int • Shipping_date: date on which the company ships the goods, Datetime
Specify	Describes which items are provided by a contract	• Contract (1,N) • Item (1,N)	• Purchased_quantity: the quantity of items which are purchased, int • Price: the price of the amount of items which are purchased, float
Belong (1)	Links items to the category	• Item (1,1) • Item_Category (0,N)	None
Belong (2)	Links products to the category	• Product (1,1) • Product_Category (0,N)	None
Belong (3)	Links packages to the category	• Package (1,1) • Package_Category (0,N)	None
Made up of (1)	Describes what items are involved in creating the product	• Item (0,N) • Product (1,N)	• Quantity: the quantity of items composing a specific product
Made up of (2)	Describes what items are involved in creating the package	• Item (0,N) • Package (1,N)	• Quantity: the quantity of items composing a specific package

Stocked	Specifies the products and packages stocked in the lots	• Package (0,N) • Product (0,N) • Lot (1,1)	• Product_quantity: quantity of the included product, int • Package_quantity: quantity of the included package, int
Draws from	Associates the lots to an order	• Order (1,N) • Lot (0,1)	None

External Constraints

- The units of measure used by the company are specified in detail in the entity table.
- Only Employees that are sill working (i.e. those who have the "Still_working" attribute set to "True") in the company can access the system.
- Only the Manager can insert new contracts, so the Employee who takes part in the "stipulate" relationship must have the role equal to "Manager".
- Only the Salesman can insert new orders, so the Employee who takes part in the "place" relationship must have the role equal to "Salesman". Only the Salesman can update the order status by changing the attribute "Order_paid" which by default is set to false.
- Only the Worker can ship orders, so the Employee who takes part in the "ship" relationship must have the role equal to "Worker".
- A Product is made up of one or more items taken from a given domain, i.e. having a certain Item_Category (e.g., "ingredient" or "glass bottle"). A Package is made up of one or more items taken from a given domain, i.e. having a certain Item_Category (e.g., "box", "plastic tape" or "polystyrene").
- The quantities of products contained in a lot belong to a finite set (e.g. 25, 50, 100).
- An expired lot cannot be sold.

Functional Requirements Satisfaction Check

The DBMS has to be able to:

- **store all the details of the employees, customers and suppliers in the organization:** Employee entity stores data related to the employees. Customer entity has details about the customers and Supplier entity has data related to suppliers.
- **allow the employees to update their personal information:** Employee entity has some attributes as Email_address, Password or Phone_number that can be changed. Employees can access the system using their credentials and change this data.
- **store details of all on-hand products in the inventory such as item code, item description, quantity and expiration date:** The inventory is represented by the entities Item, Product, Package and Lot. The Item entity contains information relating to materials (e.g., ingredients, packaging materials, ...) with their respective descriptions and quantities. Likewise, the Product entity contains information relating to finished products and the Package entity contains information relating to packaging. The products are packed in lots. Each lot is also characterized by an expiry date.

- **allow the employees to log into the system and enter the inbound items they received with information item code, item description, quantity, expiration date and supplier:** Employees can log in the database and insert data about new items in the system. An employee can also update an existing Item and its respective quantity.
- **show and generate the list of inbound and outbound transactions:** The inbound transactions can be generated by inspecting the instances of the Contract entity, while the outbound transactions can be obtained by inspecting the instances of the Order entity.
- **allow the employees to log into the system and enter the outbound transaction needed for the issuance of the products in the production and shipment to the customers; inventory stocks will be automatically updated whenever there are inbound and outbound transactions; show and generate the current inventory balance or stock inquiries:** Regarding items, the update is executed automatically when an inbound transaction occurs by inspecting the new Contract: for each Item the quantity "Item_Quantity" is increased accordingly. Regarding the outbound transaction, the value of "Item_Quantity" is decreased when a new lot (which stocks a product that is made up of that Item) is prepared. Regarding products, the stock quantity can be obtained by inspecting the lots produced but not yet ordered or expired. The same holds true for packages.
- **receive and process the Customers order, specifying which products they want and respective quantity:** Salesmen are able to access the database and enter an instance of the Order entity reporting the lots containing the desired products only when all lots are ready.
- **allow users to view order and shipment status of finished products; create tracking code for orders:** With the unique tracking number (attribute "Track_num" of the Ship relationship), and the unique ID attribute of the Order entity, the users can get information about the order and shipment.
- **generate invoice whenever payment has been made:** When an order is placed, the invoice is automatically generated by the application connected to the system. The data is extracted from the entities Order and Lot, and from the "Draws from" relationship. The total amount, net price, taxes and the list of ordered Lots are specified.
- **grant Cycle Counting in order to validate the accuracy of inventory:** Cycle counting is a periodic check done by a warehouse worker on the items in the physical inventory. After acquiring the real quantities for each item, a check will be made on the system. In case of mismatch, the "Quantity" attribute of the Item is updated.
- **re-ordering the previous orders is allowed:** The system allows salesmen to access past orders and lots using the ID attribute and retrieve information about the lots, the products, and their quantities. In this way, the customer can order the same goods.

Logical Design

Transformation of the Entity-Relationship Schema

Redundancy Analysis

The Employee and Order entities, related to each other through the relationships Place and Ship, do not form a cycle because:

- Only the Employee with the Salesman role can place the order;
- Only the Employee with the Worker role can ship the order;
- Eliminating the Place or Ship relationship implies a violation of functional requirements and a loss of information (e.g., who is the seller who places the order or who is the worker who ships it).

The Lot, Product, Package, and Item entities do not form a loop because:

- Eliminating a "Made up of" relationship implies a violation of functional requirements and a loss of information (eg, from which items, and in what quantity, a product or package is composed);
- Eliminating the "Stocked" relationship implies a violation of functional requirements and a loss of information (e.g., it is not possible to know which products and packages are contained in a lot).

The ER schema presents the derivate attributes:

- "Quantity" (of Item): used to keep in memory the current quantity of items;
- "Net_price" (of Order): used to keep in memory the total net price of an order;
- "Taxes" (of Order): used to keep in memory the total taxes of an order.

We report below the analysis of the database load to check whether keeping these attributes or not.

Choice of Principal Identifiers

The schema does not contain external identification cycles and the main identifiers comply with the selection criteria.

Analysis of Database Load

The load analysis is divided in two parts: the first, to decide whether to store "lot_price" into the "Lot" entity, or computing it when necessary via the relationship "Stocked"; the second, to decide whether to store "quantity" into the "Made up of (2)" relationship, or computing it when necessary via the relationship "Made up of (2)". The second part of the analysis is done also for the other relationship "Made up of (1)" and leads to the same results. We can see that O_1, with or without redundancy, necessitate the same amount of operations, while

Operation	Description	Frequency	Type
O_1 : Insert new lot	Store data about a newly packaged lot.	25/week	Online
O_2 : Compute order price	Compute order price from lot price	25/week	Online
O_3 : Create new order	Create new package from items	100/week	Online
O_4 : Compute order price	Compute the price of the order, which is composed of several lots	100/week	Online
O_5 : Create new product	Create new product after checking sufficient item quantity	500/week	Online

Table 4: O_1 Without redundancy

Concept	Construct	Access	Type	Average Access
Product	Entity	1	R	1 x 25 x 1 = 25
Lot	Entity	1	W	1 x 25 x 2 = 25
Stocked	Relationship	1	W	1 x 25 x 2 = 25
Total Access				**75**

Table 5: O_1 With redundancy

Concept	Construct	Access	Type	Average Access
Product	Entity	1	R	1 x 25 x 1 = 25
Lot	Entity	1	W	1 x 25 x 2 = 25
Stocked	Relationship	1	W	1 x 25 x 2 = 25
Total Access				**75**

Table 6: O_2 Without redundancy

Concept	Construct	Access	Type	Average Access
Product	Entity	1	R	1 x 25 x 1 = 25
Lot	Entity	1	W	1 x 25 x 2 = 25
Stocked	Relationship	1	W	1 x 25 x 2 = 25
Total Access				**75**

from O_2 we can assert that the number of operations without redundancy is tripled with respect to the case with redundancy. Hence, the attribute "lot_price" of the entity "Lot" should to be kept.

Table 7: O_2 With redundancy

Concept	Construct	Access	Type	Average Access
Lot	Entity	1	W	1 x 25 x 2 = 25
Total Access				**25**

Table 8: O_3 Without redundancy

Concept	Construct	Access	Type	Average Access
Lot	Entity	1	R	1 x 100 x 1 = 100
Draws from	Relationship	1	W	1 x 100 x 2 = 200
Order	Entity	1	W	1 x 100 x 2 = 200
Total Access				**500**

Table 9: O_3 With redundancy

Concept	Construct	Access	Type	Average Access
Lot	Entity	1	R	1 x 100 x 1 = 100
Draws from	Relationship	1	W	1 x 100 x 2 = 200
Order	Entity	1	W	1 x 100 x 2 = 200
Total Access				**500**

Table 10: O_4 Without redundancy

Concept	Construct	Access	Type	Average Access
Order	Entity	1	R	1 x 100 x 1 = 100
Draws from	Relationship	1	R	1 x 100 x 1 = 100
Lot	Entity	1	R	1 x 100 x 1 = 100
Total Access				**300**

Table 11: O_4 With redundancy

Concept	Construct	Access	Type	Average Access
Order	Entity	1	R	1 x 100 x 1 = 100
Total Access				**100**

We can see that O_3, with or without redundancy, necessitate the same amount of operations, while from O_4 we can assert that the number of operations without redundancy is tripled with respect to the case with redundancy, as well. Hence, the attribute "Net_price" of the entity "Order" should to be kept.

Table 12: O_5 Without redundancy

Concept	Construct	Access	Type	Average Access
Product	Entity	1	W	1 x 500 x 2 = 1000
Belong (2)	Relationship	1	W	1 x 500 x 2 = 1000
Product Category	Entity	1	R	1x 500 x 1 = 500
Made_up_of (1)	Relationship	1	W	1 x 500 x 2 = 1000
Item	Entity	10	R	10 x 500 x 2 = 10000
Total Access				**13500**

Table 13: O_5 With redundancy

Concept	Construct	Access	Type	Average Access
Item	Entity	1	R	1 x 500 x 1 = 500
Product	Entity	1	W	1 x 500 x 2 = 1000
Belong (2)	Relationship	1	W	1 x 500 x 2 = 1000
Product Category	Entity	1	R	1x 500 x 1 = 500
Made_up_of (1)	Relationship	1	W	1 x 500 x 1 = 1000
Total Access				**4000**

With redundancy, to know if there are enough items to create the requested product, we just have to read the "quantity" attribute of the "Item" entity to know if there are enough of them for the requested product; then, we create a new entry in "Product", "Belong (2)", and "Made_up_of (1). Without redundancy, we would have computed the quantity of each item, necessary to make the requested product, in advance. We have assumed that a product is made up of ten items on average, hence the average reads would have been 10 instead of 1, with redundancy. So, the attribute "quantity" should be kept.

Relational Schema

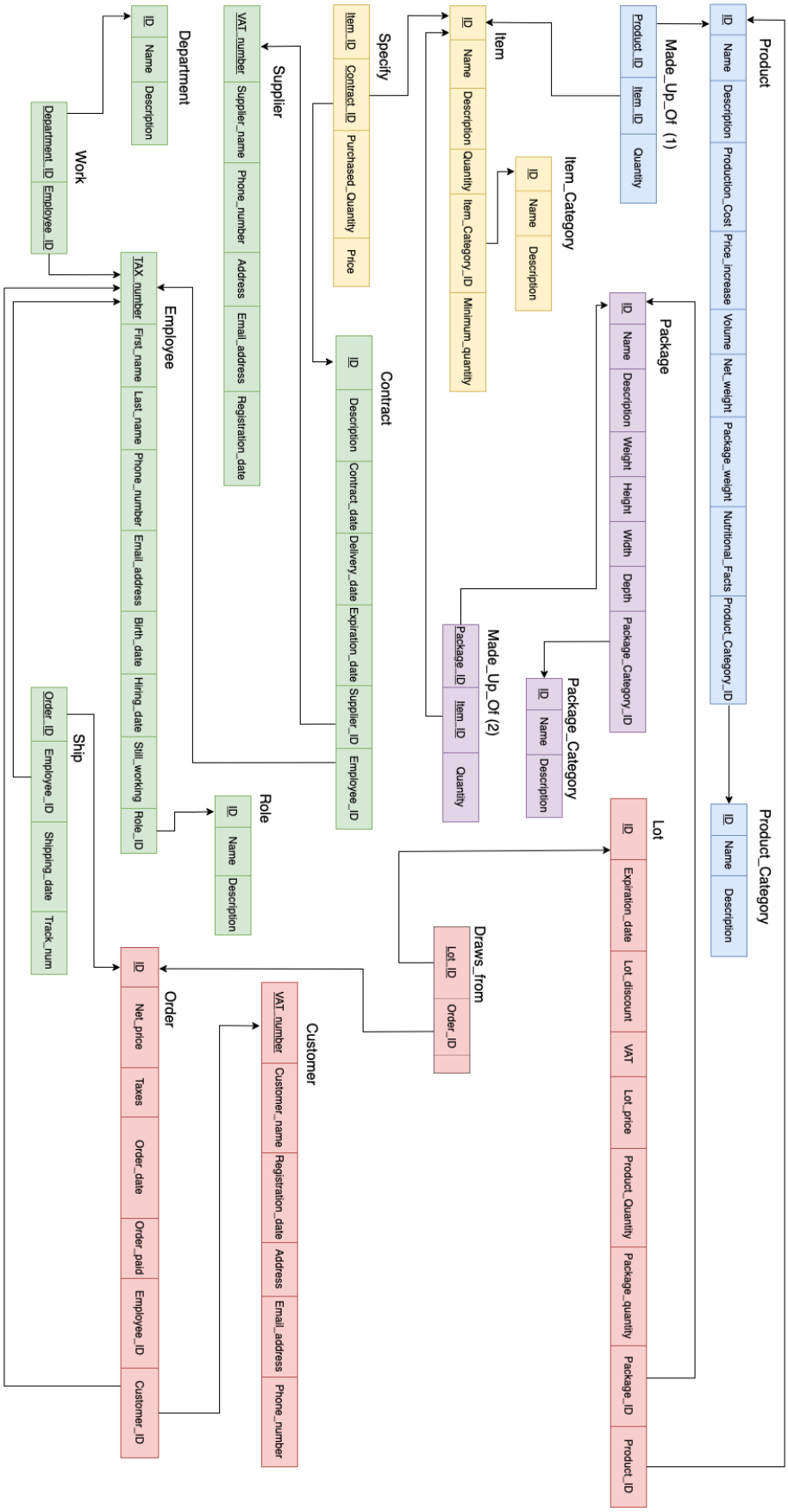

Figure 2: Relational schema.

Ship is a one to many relationship with optional participation either of Order and of Employee, due to the fact that an order can be ready to ship but not been shipped yet and a Worker might not have shipped an order yet. We have created the "Ship" relation, instead of incorporating its attributes into "Order", because if an order is ready but not yet shipped, the "Shipping_date" and "Track_num" attributes (as well as the "Employee_ID" foreing key) would be NULL, and there could be many orders in this situation. Draws_from is a one to many relationship with optional participation of Lot, due to the fact that a Lot can be produced in advance (and so it wouldn't be assigned to any order yet). We have created the "Draws_from" relation, instead of corporating the "Order_ID" foreing key into Lot because if a Lot is not assigned to any order yet, the "Order_ID" foreing key would be null, and there could be many lots in this situation.

Data Dictionary

Relation	Attribute	Description	Domain	Constraints
Product	ID	Identifier of a product	Serial	PRIMARY KEY
	Name	Name of a product	Text	NOT NULL
	Description	Description of a product	Text	
	Production_Cost	Production cost of a product	Float	NOT NULL
	Price_increase	Price increase factor of a product	Float	NOT NULL
	Volume	Volume of a product in milliliters	Int	NOT NULL
	Net_Weight	Net weight of a product in grams	Int	NOT NULL
	Package_weight	Package weight of a product in grams	Int	NOT NULL
	Nutritional_Facts	Description of nutritional facts of a product	Text	NOT NULL
	Product_Category_ID	Identifier of a Product Category	Serial	NOT NULL, Foreign Key to Product_Category
Product_Category	ID	Identifier of a Product Category	Serial	PRIMARY KEY
	Name	Name of a Product Category	Text	NOT NULL
	Description	Description of a Product Category	Text	
Item	ID	Identifier of an item	Serial	PRIMARY KEY
	Name	Name of an item	Text	NOT NULL
	Description	Description of an item	Text	
	Quantity	Stock quantity of an item	Int	NOT NULL
	Minimum_quantity	Minimum stock quantity of the item	Int	NOT NULL
	Item_Category_ID	Identifier of an Item Category	Serial	NOT NULL, Foreign Key to Item_Category
Item_Category	ID	Identifier of an Item Category	Serial	PRIMARY KEY
	Name	Name of an Item Category	Text	NOT NULL
	Description	Description of an Item Category	Text	
Specify	Item_ID	Identifier of an item	Serial	Foreign Key to Item, Primary key with Contract_ID
	Contract_ID	Identifier of a contract	Serial	Foreign Key to Contract, Primary key with Item_ID
	Price	The price of the amount of items which are purchased	Float	NOT NULL

	Purchased_Quantity	Amount of each item which is purchased	Int	NOT NULL
Contract	ID	Identifier of a contract	Serial	PRIMARY KEY
	Description	Description of a contract	Text	
	Contract_date	Date of signature of a contract with the supplier	Datetime	NOT NULL
	Delivery_date	Expected date of delivery of the goods	Datetime	NOT NULL
	Expiration_date	Expiration date of a contract	Datetime	NOT NULL
	Supplier_ID	Identifier of a supplier	Int	NOT NULL, Foreign Key to Supplier
	Employee_ID	Identifier of an employee	Text	NOT NULL, Foreign Key to Employee
Package	ID	Identifier of a package	Serial	PRIMARY KEY
	Name	Name of a package	Text	NOT NULL
	Description	Description of a package	Text	
	Weight	Weight of a package in grams	Int	NOT NULL
	Height	Height of a package in centimeters	Int	NOT NULL
	Width	Width of a package in centimeters	Int	NOT NULL
	Depth	Depth of a package in centimeters	Int	NOT NULL
	Package_Category_ID	Identifier of a Package Category	Serial	NOT NULL, Foreign Key to Package_Category
Package_Category	ID	Identifier of a Package Category	Serial	PRIMARY KEY
	Name	Name of a Package Category	Text	NOT NULL
	Description	Description of a Package Category	Text	
Lot	ID	Identifier of a lot	Serial	PRIMARY KEY
	Expiration_date	Expiration date of the included products	Datetime	NOT NULL
	Product_Quantity	Amount of a product in each lot	Int	NOT NULL
	Package_Quantity	Amount of a package in each lot	Int	NOT NULL
	Lot_Discount	Percentage of discount of a lot	Int	NOT NULL
	VAT	Value added tax (percentage)	Int	NOT NULL
	Lot_price	Lot price without discount	Float	NOT NULL
	Package_ID	Identifier of a package in each lot	Serial	NOT NULL, Foreign Key to Package
	Product_ID	Identifier of a product in each lot	Serial	NOT NULL, Foreign Key to Product
Order	ID	Identifier of an order	Serial	PRIMARY KEY
	Net_price	Total net amount including discount (VAT excluded)	Float	NOT NULL
	Taxes	Amount of taxes to be paid	Float	NOT NULL
	Order_date	Date in which the order has been processed	Datetime	NOT NULL
	Order_paid	Status of the order payment	Boolean	NOT NULL
	Employee_ID	Identifier of the employee that places the order	Text	NOT NULL, Foreign key to Employee
	Costumer_ID	Identifier of the customer that places the order	Int	NOT NULL, Foreign key to Customer
Customer	VAT_number	VAT_number of the customer	Text	PRIMARY KEY

	Customer_name	Name of the customer	Text	NOT NULL
	Phone_number	Phone number (prefix included) of the customer	Text	NOT NULL
	Address	Billing address of the customer	Text	NOT NULL
	Email_address	Email address of the customer	Text	NOT NULL
	Registration_date_ID	Customer registration date	Datetime	NOT NULL
Draws_from	Lot_ID	Identifier of a lot	Serial	Foreign Key to Lot, Primary key to Lot
	Order_ID	Identifier of an order	Serial	NOT NULL, Foreign Key to Order
Made_Up_Of (1)	Product_ID	Identifier of a product	Serial	Foreign key to Product, Primary key with Item_Id
	Item_ID	Identifier of an item	Serial	Foreign key to Item, Primary key with Product_Id
	Quantity	Amount of each item in a product	Int	NOT NULL
Made_Up_Of (2)	Package_ID	Identifier of a package	Serial	Foreign key to Package, Primary key with Item_Id
	Item_ID	Identifier of an item	Serial	Foreign key to Item, Primary key with Package_Id
	Quantity	Amount of each item in a package	Int	NOT NULL
Ship	Order_ID	Identifier of the shipped order	Serial	Foreign key to Order, Primary key
	Employee_ID	Identifier of employee that ships the order	Serial	NOT NULL, Foreign key to Employee
	Shipping_date	Date of when the order leaves to be shipped	Datetime	NOT NULL
	Track_num	Shipment code given by the third-party shipping company	Text	
Employee	TAX_number	TAX code of the employee	Text	PRIMARY KEY
	First_name	Name of the employee	Text	NOT NULL
	Last_name	Surname of the employee	Text	NOT NULL
	Phone_number	Phone number (prefix included) of the employee	Text	NOT NULL
	Email_address	Email address of the employee	Text	NOT NULL
	Birth_date	Birthdate of the employee	Datetime	
	Hiring_date	Hiring date of the employee	Datetime	NOT NULL
	Still_working	Flag used to know if employee is still working for the company	Boolean	NOT NULL
	Role_ID	Identifier of the role that an employee has	Serial	NOT NULL, Foreign Key to Role
Role	ID	Role identifier	Serial	PRIMARY KEY
	Name	Name of the role	Text	NOT NULL
	Description	Technical description of the role	Text	
Work	Department_ID	Identifier of a department	Serial	Foreign key to Department, Primary key with Employee_ID

	Employee_ID	Identifier of an employee	Text	Foreign key to Employee, Primary key with Department_ID
Department	ID	Department identifier of the company	Serial	PRIMARY KEY
	Name	Name of the department	Text	NOT NULL
	Description	Description of the department's function	Text	
Supplier	VAT_number	VAT number of the supplier company	Int	PRIMARY KEY
	Supplier_name	Name of the supplier company	Text	NOT NULL
	Phone_number	Phone number (prefix included) of the supplier company	Text	NOT NULL
	Email_address	Email address of the supplier company	Text	NOT NULL
	Address	Address of the supplier company	Text	NOT NULL
	Registration_date	Recording date of the supplier company	Datetime	NOT NULL

External Constraints

- Only employees that are sill working (i.e. those who have the "Still_working" attribute set to "True") in the company can access the system. So Employees that take part in any operation of insertion or modification must have True as value of attribute Still_working in its own Employee Relation.
- Only the Salesman can insert new orders, so the Employee_ID in the Order relation must have the role equal to "Salesman". Only the Salesman can update the order status.
- Only the Manager can insert new contracts, so the Employee_ID of the Contract relation must have the role equal to "Manager".
- Only the Worker can ship orders, so the Employee_ID in the Ship relation must have the role equal to "Worker".

Group Members Contributions

Conceptual Design

- **Variations to the Requirement Analysis:** Esposito, Basso
- **Entity-Relationship Schema:** Esposito, Basso, Zanini, Collado, Giuliani
- **Entities Table:** Esposito, Basso
- **Relationships Table:** Zanini e Giuliani
- **External Constraints:** Quiroz, Collado, Esposito, Basso, Cimarosto
- **Functional Requirements Satisfaction Check:** Cimarosto, Collado, Arslan, Esposito, Basso

Logical Design

- **Transformation of the Entity-Relationship Schema:** Esposito, Basso
- **Analysis of Database Load:** Giuliani e Zanini

- **Relational Schema:** Esposito, Basso, Quiroz, Collado
- **Data Dictionary:** Collado, Arslan, Cimarosto, Basso, Esposito
- **External Constraints:** Collado, Basso, Esposito