Università degli Studi di Salerno

.DIEM

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata

Corso di Laurea in Ingegneria Informatica

Basi di Dati 2023/2024
Canale A-H

Project Work

Traccia N. A16 – Fish2Home

Gruppo n. **31 – AH**

WP	Cognome e Nome	Matricola	e-mail	Responsabile
1-3	Cirillo Francesco Pio	0612705370	f.cirillo36@studenti.unisa.it	X
2-4	Fasolino Alessandra	0612705401	a.fasolino35@studenti.unisa.it	

Academic Year 2023-2024

Summary

1. Description of the reality of interest

Title: **Fish2Home**

Fish2Home is a catch-of-the-day delivery service that hires and manages riders who pick up the catch (fish bundle) from the customer's chosen fisherman, and deliver it directly to the customer's home. Fish2Home commissioned the design and implementation of a database to support this service.

In order to order fish, the customer must sign up and complete their user profile with personal details, contact information and usual address, as well as username and password to access the service. Storage of at least one debit/credit card is also required. In general, the customer may have more than one payment card associated with their profile.

There are various types of fishermen who can sell fish using this service. Obviously, the main information characterizing the fisherman is of interest (the vat number, tax code, name, a description, type, address, etc.). The individual fisherman obviously may or may not own a single boat (the type of which is to be considered an attribute) depending on the type of fishing he or she does: Bottom trawling, Longline fishing, Pole and line fishing, etc. In addition, with respect to the catch of the day, each fisherman offers an assortment (Bundle) of fish that also depends on the type of fishing he performs and that characterizes him.

In the case dealt with, the customer has to buy a bundle of fish indicating the weight of interest, but in any case the service does not offer the possibility of providing the customer with detailed information on the composition of the bundle of interest at the level of each individual fish.

Orders essentially consist of bundles assigned based on requests, but the system must allow tracking of orders placed by users and bundles assigned by reporting for each bundle delivered to the customer, payment information, quantities, total cost, and the destination address, which may be different from the user's home.

Each order must be delivered to its destination by a rider. To organize deliveries, the system divides the riders into zones, assigning each zone to one or more riders. The size of the zone depends on the number of inhabitants present. The order to be delivered will be assigned to a rider belonging to the zone in which to make the delivery. You want to track rider pickups of fish at the fisherman's place and deliveries to homes, specifically reporting pickup and delivery times. The service covers a limited geographical area in a single province.

1.1. Analysis of the reality of interest

The objective of the project is to build a database to create a service for smart sales of fish bundles directly from fishing boats to the customer via a rider.

It is taken into consideration that the service in question is active in the Amalfi Coast area.

Consider the presence of various types of fishermen who can sell different species of fish organized in bundles categorized by weight. Each fisherman can own a maximum of one boat and can specialize in more than one type of fishing (Bottom trawling, Longline fishing, Pole and line fishing, etc...). Customers who are subscribed to the service can buy bundles by accessing a description and weight. Deliveries are made by riders.

It was decided to characterize the project by orienting more accurately to the modeling of the delivery service; orders, customers, riders, and order contents, i.e., fish bundles offered by fishermen, will be modeled accurately. Each fisherman can offer bundles of his own catch, characterizing his proposals in the way he believes will best benefit his sales. Customers will not be able to create customized bundles but can only choose from those proposed to safeguard the interests of the fishermen.

The objective of the project is the proper modeling of the whole aspect related to selling and buying, describing less accurately the details related to the production of the raw material. Fishing techniques and vessels will still be modeled but only in order to provide more information to the end user, not to ensure adherence to certain business rules. The fisherman will have to ensure for himself that the information he provides about his fishing techniques is consistent with the products he offers.

2. Specification analysis

Workpackage	*Task*	*Responsible*
WP0	Specification Analysis	Entire Group

2.1. Glossary of terms

	Term	Description	Synonyms	Links
1	Bundle	Bundle of fish arranged by the fisherman. It is purchased as part of an order.	catch, assortment	fisherman, order, boat
2	Fisherman	He who uses the platform to sell his catch organized in bundles. Can carry out different types of fishing with the help of one only type of boat.	-	bundle
3	Customer	The one who places orders to the fisherman to buy bundle. Lives in an area of his City of residence.	user	order, zone
4	Boat	Boat of property of the Fisherman used to practice one or more fishing methods.	-	fisherman
5	Order	Purchase by a customer for one or more bundles. It is delivered by a rider.	-	customer, bundle, rider
6	Rider	Employee at delivery of fish orders to customers residing in an area.	-	zone, order
7	Zone	Geographical region to which one or more riders are assigned on the basis of population density.	-	rider, order, client

Table 1. Glossary of Terms

2.2. Structuring of requirements into phrases

2.2.1. General phrases

Fish2Home is a catch-of-the-day delivery service that hires and manages riders who pick up the catch from the customer's chosen fisherman, and deliver it directly to the customer's home.

2.2.2. Customer-related phrases

The customer to access the service must sign up and create a profile.

The customer to access the profile must provide username and password.

The customer's profile must contain username, password, biographical data, telephone contact information, usual residence, and one or more credit/debit cards.

The customer purchases a bundle of fish by indicating the weight of interest, being able to a c c e s s related information.

2.2.3. Phrases related to the fisherman

Of the fisherman, the vat number, tax identification number, name, a description, type, and address are known.

The individual fisherman may or may not own a single boat depending on the method of fishing he or she does: Bottom trawling, Longline fishing, Pole and line fishing, etc .

The individual fisherman may engage in one or more types of fishing.

2.2.4. Phrases related to 'ordering

Orders consist of Bundles assigned based on customer requests.

The system must allow tracking of orders placed by users and bundles assigned.

Each order contains information for each bundle delivered to the customer: payment information, quantities, total cost, and destination address.

The destination address may be different from the user's home address.

2.2.5. Rider-related phrases

Each rider is assigned to a zone; multiple riders can be assigned to each zone. The size of

the zone depends on the number of inhabitants present.

2.2.6. Phrases related to order delivery

Each order must be delivered to its destination by a rider.

The order to be delivered will be assigned to a rider belonging to the area where the delivery is to be made.

You want to track rider pickups of fish at the fisherman's place and deliveries to homes, specifically reporting pickup and delivery times.

2.2.7. Phrases related to the bundle

The bundle is entered by the fisherman.

The bundle should contain information such as description and weight.

2.3. Identification of main operations

Operation 1: Search for fishermen who sell fish at a temperature at the time of pickup on average greater than 5°C, therefore whose quality is compromised (1/month)

Operation 2: Selection of all customers who live in "Praiano1" but order for a different area (1/year)

Operation 3: Calculating a customer's monthly expense (900/month)

Operation 4: Creating a booked order (200/day)

Operation 5: Calculating quantity of bundles delivered by a rider (40/month)

Operation 6: Search for riders, and their work zones, who are the only riders assigned to the zone and deliver an above-average number of orders (1/month)

3. Conceptual Design

Workpackage	*Task*	*Responsible*
WP1	Conceptual Design	Cyril Francis Pius

3.1. Conceptual Outline

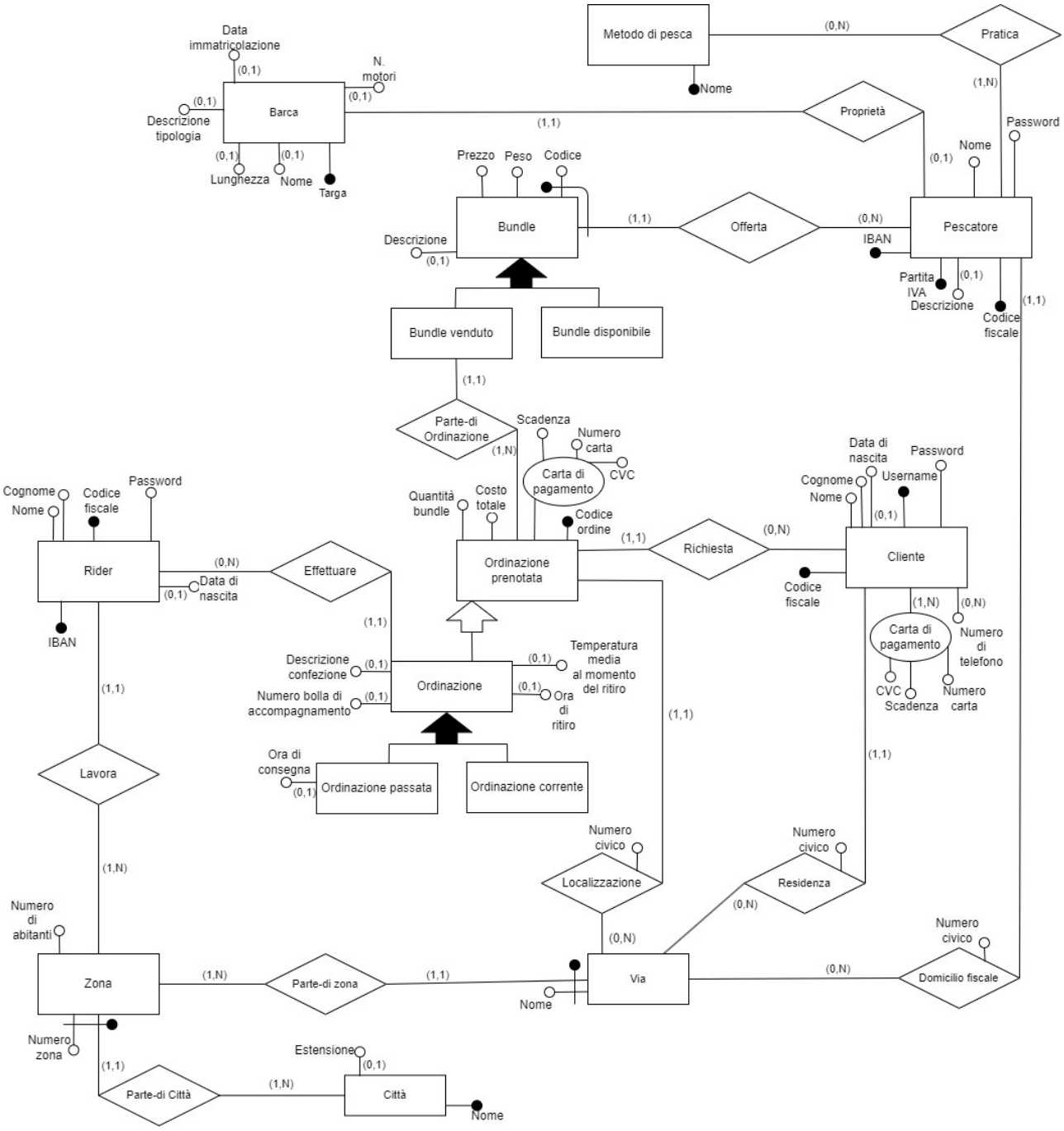

Figure 1. E-R Scheme

3.1.1. Notes on the E-R schema

In order to create the ER schema, a mixed strategy was chosen in that while deciding to divide the requirements into separate components, the definition of a skeleton schema was not waived in order to maintain an overall view.

For simplicity we model all the different types of streets with the entity "Street" ignoring the differences between squares, streets, roads, cross streets, etc.

We point out that the choice to identify the City entity by its name is justified and does not represent a critical issue since the service being modeled covers a geographic area contained within a province and, at least in Italy at present, there are no homonymous municipalities in the same province let alone in the same region.

Note the choice not to consider the Accompanying Bill Number as the Order identifier, this being because it is theoretically possible for multiple Orders to be part of the same shipment.

In order not to visually burden the illustrative content of the documentation, the figures showing fragments of the E-R diagram (*Figure2* through *Figure19* and *Figure22*) do not show the multiplicities of optional attributes.

One notices the use of two different types of "Part-of" conceptual patterns. All use cases of this pattern fall into the case where the component entity (e.g., "Street" between "Street" and "Zone") has no independent existence with respect to the compound entity. An exception is the association between the entities Bundle Sold and Order Booked, in fact the existence of a Bundle is not conditional on its belonging to an Ordering. The Bundle is perfectly identified by the Code is the association that binds it to the entity Fisherman.

Finally, the conceptual scheme obtained on the basis of the 4 quality parameters is evaluated:

- Correctness, attention has been paid to the non-misuse of ER model constructs;
- Completeness, all data of interest are modeled in the presented schema and all operations of interest are performable using concepts described in the schema;
- Readability, the schema appears understandable, care was taken in the choice of names so that they were meaningful, and all concepts were analyzed so that they were clarified in detail;
- Minimality, with regard to minimality, the presence of redundancies is acknowledged which therefore make the pattern non-minimal, however not all redundancies are necessarily bad, specifically the redundancies present are not the result of an error but rather of shrewd design choices; at the appropriate stage a decision will be made whether to maintain or eliminate the redundancies in question.

3.2. Design Pattern

3.2.1. Pattern Evolution of a Concept

The Booked Order when picked up by the rider undergoes an evolution over time in Ordering due to the addition of new information such as the Time of Pickup attribute and the Temperature at Pickup attribute. This more accurately models the time at which the Make association comes into being, i.e., after the rider picks up; therefore, the association changes from being optional (minimum cardinality 0) to mandatory.

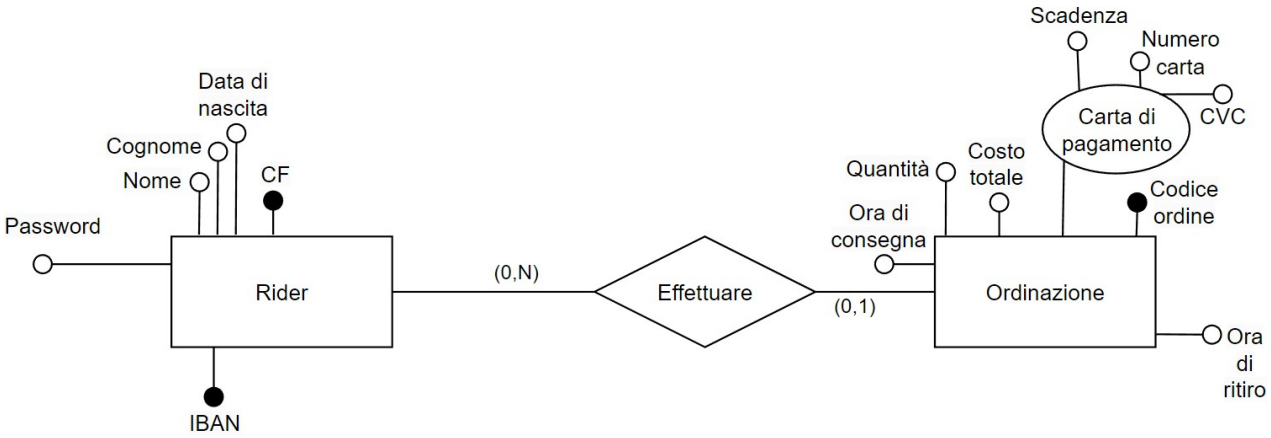

Figure 2 Pattern prior to the application of the EVOLUTION OF A CONCEPT Pattern.

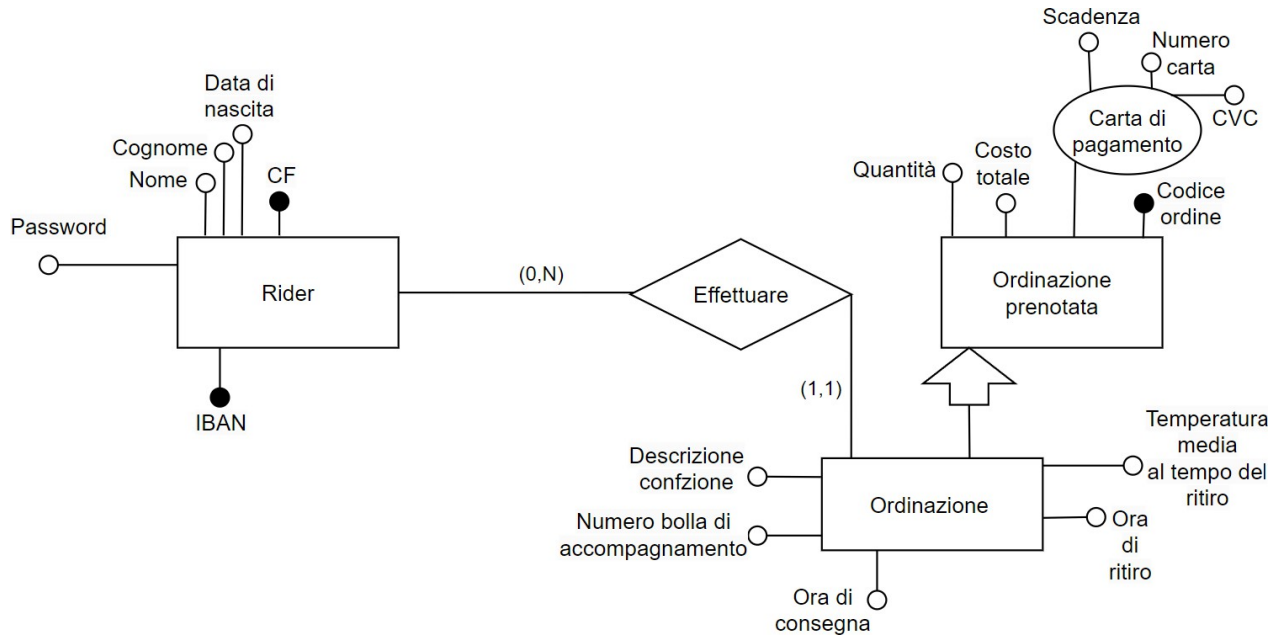

Figure 3 Pattern following the application of the EVOLUTION OF A CONCEPT Pattern.

3.2.2. Pattern Historization of entities.

In order to track past orders, the pattern of historicization was applied to distinguish between Current Ordering and Past Ordering, the latter entity being characterized by the attribute Time of Delivery, a piece of information of interest in the modeled reality.

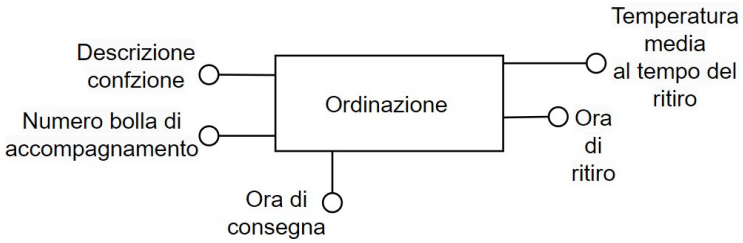

Figure 4. Pattern prior to the application of the Entity Historization Pattern.

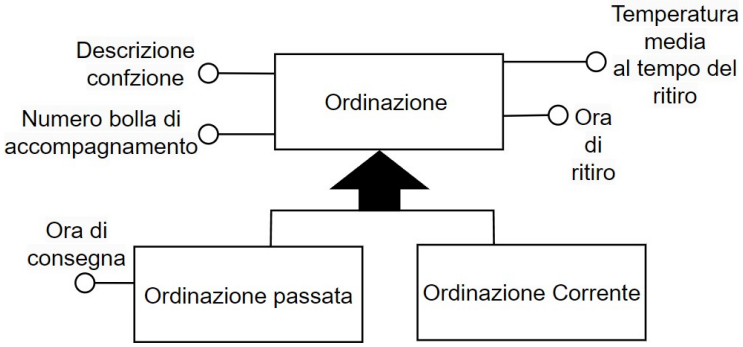

Figure 5. Diagram following the application of the ENTITY HISTORICITY Pattern.

It is considered to historicize orders for the last year.

3.3. Data Dictionary

Entity	Description	Attributes	Identifier
Boat	Boat with which the fisherman may carry out one or more fishing methods.	Date of registration, No. engines, Type description, Length, Name, License plate	License plate
Bundle	Package of fish composed by the fisherman.	Weight, Price, Code, Description	Code & Fisherman
Fisherman	Individual who practices certain fishing techniques. Offers clients his or her catch in a bundle.	VAT Number, Tax Code, Name, Description, Password, IBAN	VAT Number, Tax Code, IBAN
Ordering	Purchase of certain bundles by a customer, must be delivered to the customer by a rider.	Bundle quantity, Total cost, Payment card (Card number, Expiration date, CVC), Order code, Pickup time, Delivery time, Average temperature at the time of pickup, Package description, Bill number accompanying	Order Code
Rider	Individual who picks up bundles from fishermen to deliver to customers.	First name, Last name, Date of birth, Tax code, Password, IBAN	Tax Code, IBAN
Customer	Individual placing orders.	First name, Last name, Date of birth, Tax code, Username, Password, Payment card (Card number, Expiration, CVC), Number of phone	Tax Code, Username
Zone	Portion of a city that includes a number of streets.	Number of inhabitants, Zone Number	Zone & City Number
Street	Street in a city that is part of a zone.	Name	Name & Zone

Table 2. Data dictionary-Entity

Relationships	Description	Entities Involved	Attributes
Bid	Associates an Angler with the Bundles it offers for sale.	Fisherman (0,N), Bundle (1,1)	
Part-of-Order	Associates a sold Bundle to the Ordering of which it is a part.	Sold Bundle (1,1), Booked Ordering (1,N)	
Make	Associates the Rider with the Orders it is to deliver.	Rider (0,N), Ordering (1,1)	
Request	Associate a Customer with their Booked Orders.	Customer (0,N), Booked Order (1,1)	
Work	Associates a Zone with the Riders who work there.	Zone (1,N), Rider (1,1)	

Part-of-Zone	Associates a Zone with the Routes that make it up.	Zone (1,N), Street (1,1)	
Localized	Associates an Order with the Street at which it is to be delivered.	Order Booked (1,1), Street (0,N)	House Number
Ownership	Associates a Boat with the fisherman who owns it.	Fisherman (0,1), Boat(1,1)	

Table 3. Data dictionary - Relationships

It is intended to specify that to specify two different identifiers for the same entity the identifiers were listed by separating them with a comma; instead, to specify compound identifiers the various attributes that compose them were listed separated by &.

Workpackage	*Task*	*Responsible*
WP4	Business Rules	Fasolino Alessandra

3.4. Corporate Rules

Constraint Rules.
(RV1) A Zone has a maximum size of about 2500 inhabitants and in any case never more than 3000 inhabitants. **(RV2**) In an order the time of delivery cannot be prior then of pickup. **(RV3**) The rider may deliver only orders that are related to the zone in which he works. **(RV4**) Bundle and Order prices must be all positive. **(RV5**) The payment card related to an order must be owned by the customer who placed the order.

Table 4. constraint rules

Rules of derivation.
(RD1) The total cost of a Booked Order is obtained by summing the Price of the Bundles that comprise it. **(RD2)** The quantity of bundles in an Order is obtained by counting the bundles sold that are linked to the order in question by the Part-of-Order association.

Table 5. Rules of derivation

4. Logical Design

Workpackage	Task	Responsible
WP2	Logical Design	Fasolino Alessandra

4.1. Conceptual Scheme Restructuring

4.1.1. Performance Analysis

4.1.1.1. Volume Table

Concept	Type	Volume
Fisherman	E	50
Bundle available	E	150
Bundle sold	E	90.075
Fishing method	E	5
Boat	E	100
Customer	E	900
Order booked	E	25
Order passed	E	60.000
Current ordering	E	25
Rider	E	40
Zone	E	20
City	E	15
Street	E	450
Practice	R	150
Properties	R	100
Bid	R	90.225
Part-Of-Order	R	90.075
Make	R	60.025
Request	R	60.050
Work	R	40
Localization	R	60.050
Residence	R	900
Fiscal domicile	R	50
Part-area	R	450
Part-of-town	R	20

Table 6. table of volumes

Notes on volumes

To accurately estimate the amount of **customers** on average registered for the service, the following reasoning was followed:

- Total inhabitants of the area covered (Amalfi Coast): approximately 37,500
- Estimated number of households: approximately 9374

Considering a single enrollee per household and assuming a number of service enrollees equal to 10 percent of potential purchasers yields the estimate made.

4.1.1.2. Table of transactions

Transaction	Type	Frequency
Operation 1: Search for fishermen selling poorly preserved fish	B	1/month
Operation 2: Searching for customers who order for a different area of residence	B	1/year
Operation 3: Calculate a customer's monthly expense	B	900/month
Transaction 4: Creating a booked order	I	200/day
Operation 5: Calculating quantity o f Bundles delivered by a Rider	B	40/month
Operation 6: Riders working alone alone in the area and deliver more orders than average	B	1/semester

Table 7. Table of operations

4.2. Analysis of redundancies

- Redundancy 1: Total Cost (RESERVED ORDER). The total cost of the Order is obtained by summing the value of the Price attribute of the sold Bundle entities related to the Order in question by means of the Part-of-Order association.
  TYPE: Attribute that can be derived from other entities.
- Redundancy 2: Bundle Quantity (ORDER RESERVED). The bundle quantity of the Ordering is obtained by counting the number of occurrences of the Part-of-Bundle association.
  TYPE: Attribute Derivable from count of occurrences.

4.2.1. Redundancy analysis 1: Cost of an order.

- **Operation 3: Calculation of a customer's monthly expense.**

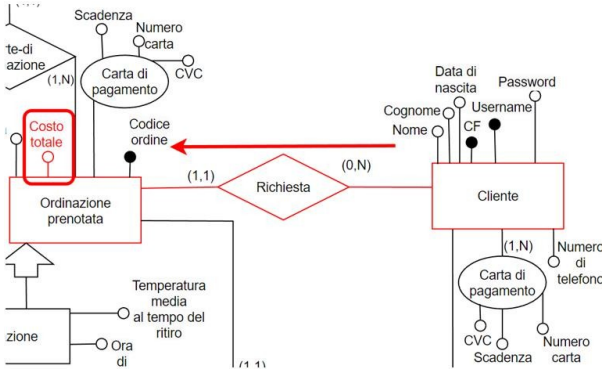

Figure 6. operation 3 path with redundancy 1

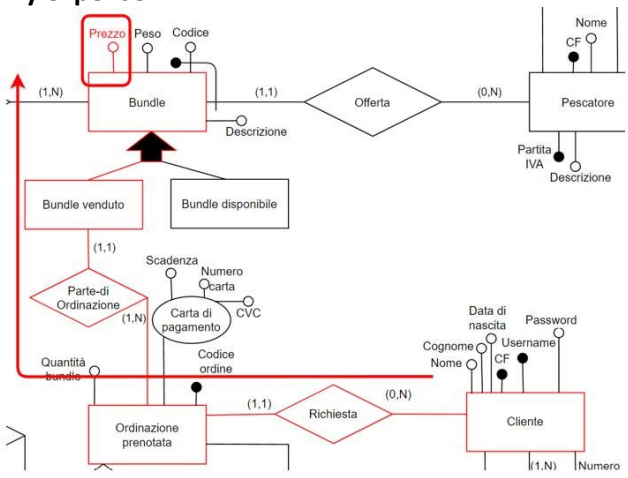

Figure 7. operation 3 path without redundancy 1

With Redundancy

CONCEPT	BUILT	ACCESS	TYPE
CLIENT	E	1	L
REQUEST	R	7	L

ORDER PASSED	E	7	L

Table 8. Access table for operation 3 in the presence of redundancy 1

Without Redundancy

CONCEPT	CONSTRUCT	ACCESSES	TYPE
CLIENT	E	1	L
REQUEST	R	7	L
ORDER PASSED	E	7	L
PART-ORDER	R	10	L
BUNDLE SOLD	E	10	L

Table 9. Access table for operation 3 in the absence of redundancy 1

- **Operation 4: creation of a booked order.**

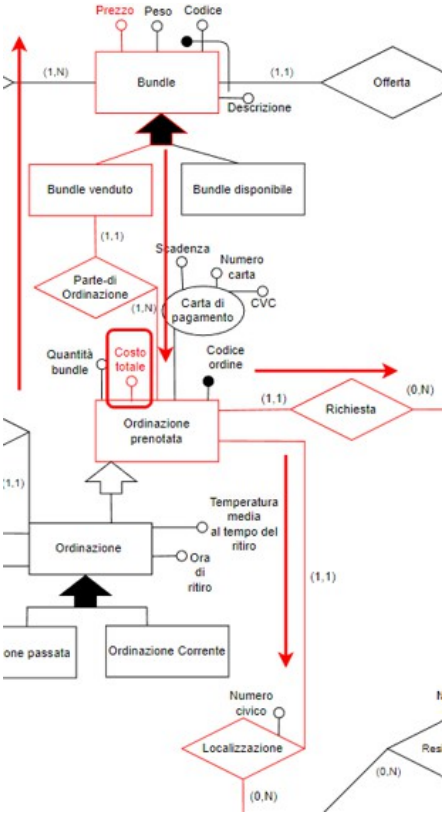

Figure 8. operation path 4 with redundancy 1

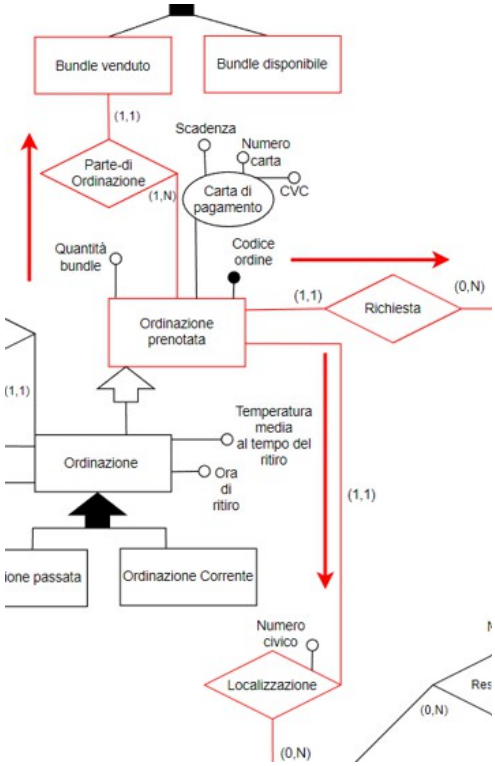

Figure 9. operation 4 path without redundancy 1

With Redundancy

CONCEPT	CONSTRUCTO	ACCESS	TYPE
ORDER BOOKED	E	1	S
LOCATION	R	1	S
REQUEST	R	1	S
PART-OF-ORDER	R	2	S
BUNDLE AVAILABLE	E	2	L
BUNDLE AVAILABLE	E	2	S
BUNDLE SOLD	E	2	L
ORDER BOOKED	E	1	L
ORDER BOOKED	E	1	S

Table 10. Access table for operation 4 in the presence of redundancy 1

Without Redundancy

CONCEPT	CONSTRUCT	ACCESSES	TYPE
ORDER BOOKED	E	1	S
LOCATION	R	1	S
REQUEST	R	1	S
PART-OF-ORDER	R	2	S
BUNDLE AVAILABLE	E	2	L
BUNDLE AVAILABLE	E	2	S

Table 11. Access table for operation 4 with no redundancy 1

Note: For both cases (operation 4 with and without redundancy) the Available Bundle Read and Write is considered necessary for conversion to Sold Bundle

4.2.1.1. Evaluation of redundancy 1

4.2.1.1.1. Memory Occupancy Calculation.

Assuming that the total cost of the booked order requires 5 bytes, sufficient to store a numeric(5,2) type, it turns out that the redundant data involves

$$5\ byte \times\ 60050\ ordinazioni = 300250\ byte - circa\ 293.21\ kilobyte$$

4.2.1.1.2. Cost of single operation

With Redundancy

OPERATION	L	S
Operation 3	15	0
Operation 4	5	8

Table 12. Access report table in the presence of redundancy 1

Without Redundancy

OPERATION	L	S
Operation 3	35	0
Operation 4	2	7

Table 13. Table account of accesses in the absence of redundancy 1

4.2.1.1.3. Cost of operations given their frequencies

With Redundancy

OPERATION	L	S
Operation 3	13500	0
Operation 4	25000	40000
TOTAL	**38500**	**40000**

Table 14. Table summary of accesses as a function of frequencies in the presence of redundancy 1

Without Redundancy

OPERATION	L	S
Operation 3	31500	0
Operation 4	10000	35000
TOTAL	**41500**	**35000**

Table 15. Table report of accesses as a function of frequencies in the absence of redundancy 1

The cost of a write access twice as much as a read access is considered. The quantities of reads and writes shown in the table are relative to monthly frequencies of operations.

4.2.1.1.4. Evaluation Conclusions.

After analyzing the operations involving redundancy, it is observed that with the load considered:

- In the presence of redundancy, the cost of operations is about 118500 monthly accesses
- The memory occupancy is about 293.21 kilobytes (unimportant size f o r modern computers)
- In the absence of redundancy, the cost of operations is 111500 daily accesses Therefore, it is

decided not to maintain redundancy as it increases the number of accesses .

4.2.2. Redundancy analysis 2: Amount of bundles in an order.

- **Step 4: Create a booked order.**

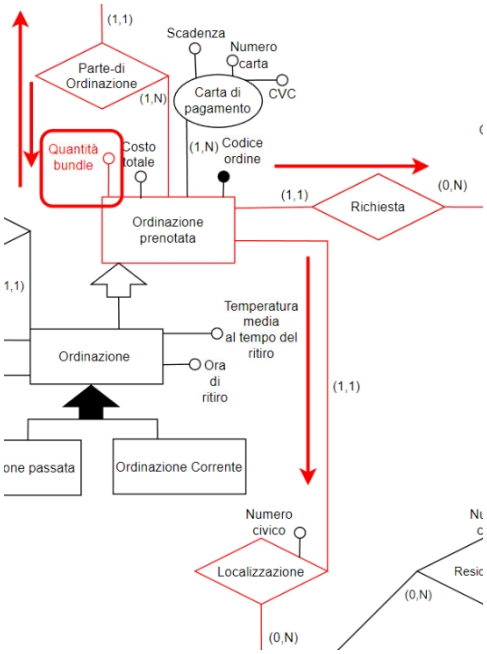

Figure 10. Operation 4 path with redundancy 2

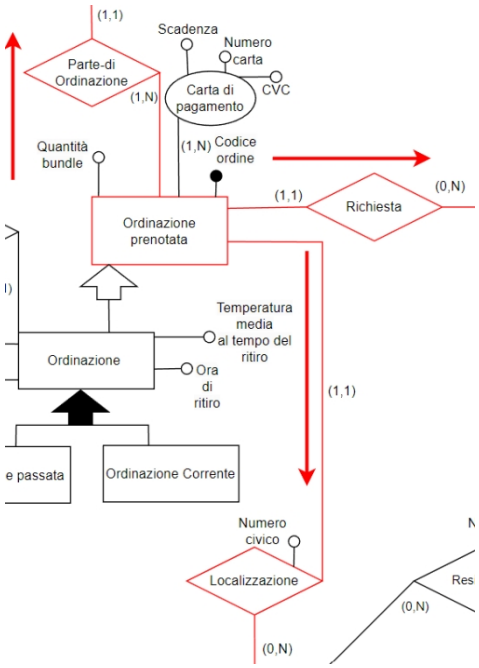

Figure 11. operation 4 path without redundancy 2

With Redundancy

CONCEPT	CONSTRUCTO	ACCESS	TYPE
ORDER BOOKED	E	1	S
REQUEST	R	1	S
LOCALIZATION	R	1	S
PART-OF-ORDER	R	2	S
BUNDLE AVAILABLE	E	2	L
BUNDLE AVAILABLE	E	2	S
PART-ORDER	R	2	L
ORDER-BOOKED	E	1	L

ORDER BOOKED	E	1	S

Table 16. Access table for operation 4 in the presence of redundancy 2

Without Redundancy

CONCEPT	CONSTRUCT O	ACCESSES	TYPE
ORDER BOOKED	E	1	S
REQUEST	R	1	S
LOCALIZATION	R	1	S
PART-OF-ORDER	R	2	S
BUNDLE AVAILABLE	E	2	L
BUNDLE AVAILABLE	E	2	S

Table 17. Access table for operation 4 in the absence of redundancy 2

- **Operation 5: calculation of quantity of bundles delivered by a rider**

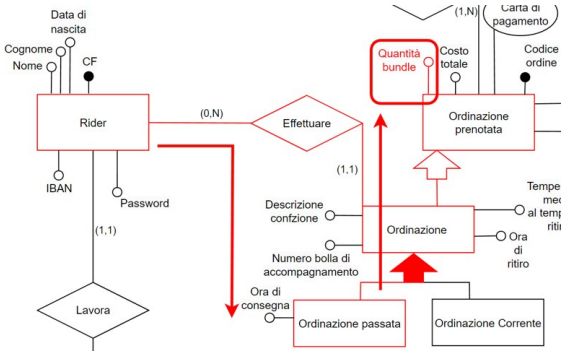

Figure 12. Operation 5 path with redundancy 2

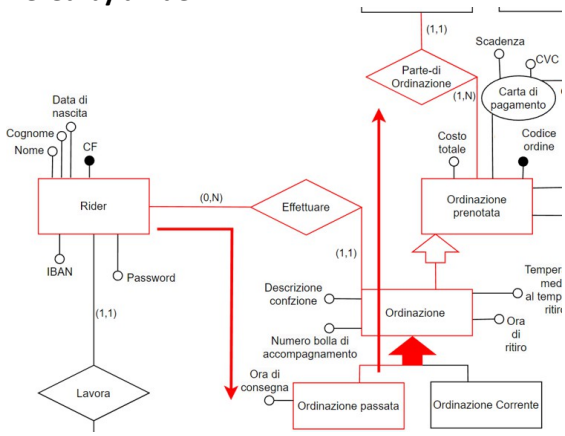

Figure 13. operation 5 path without redundancy 2

With Redundancy

CONCEPT	CONSTRUCT	ACCESS	TYPE
RIDER	E	1	L
EFFECT	R	1500	L
ORDER PASSED	E	1500	L

Table 18. Access table for operation 5 in the presence of redundancy 2

Without Redundancy

CONCEPT	CONSTRUCT	ACCESSES	TYPE
RIDER	E	1	L
EFFECT	R	1500	L
ORDER PASSED	E	1500	L
PART-OF-ORDER	R	2250	L

Table 19. Access table for operation 5 in the absence of redundancy 2.

4.2.2.1. Evaluation of redundancy 2

4.2.2.1.1. Memory occupancy calculation

Assuming that the amount of bundle in ordering requires 4 bytes, sufficient to store integers, it turns out that the redundant data involves

$$4\ byte \times 90225\ bundle = 360900\ byte - circa\ 352.44\ kilobyte$$

4.2.2.1.2. Cost of single operation

With Redundancy

OPERATION	L	S
Operation 4	5	8
Operation 5	3001	0

Table 20. access report table in the presence of redundancy 2

Without Redundancy

OPERATION	L	S
Operation 4	2	7
Operation 5	5251	0

Table 21. Table account of accesses in the absence of redundancy 2.

4.2.2.1.3. Cost of operations given their frequencies

With Redundancy

OPERATION	L	S
Operation 4	25000	40000
Operation 5	120040	0
TOTAL	**145040**	**40000**

Table 22. Table summary of accesses as a function of frequencies in the presence of redundancy 2

Without Redundancy

OPERATION	L	S
Operation 4	10000	35000
Operation 5	210040	0
TOTAL	**220040**	**35000**

Table 23. Table account of accesses as a function of frequencies in the absence of redundancy 2

The cost of a write access double that of a read access is considered.

4.2.2.1.4. Evaluation Conclusions.

After analyzing the operations involving redundancy, it is observed that with the load considered:

- In the presence of redundancy, the cost of operations is about 225040 monthly accesses
- The memory occupancy is about 352.44 kilobytes (unimportant size f o r modern computers)
- In the absence of redundancy, the cost of operations is 290040 monthly accesses

Therefore, it is decided to keep redundancy as it decreases the number of accesses.

4.3. Elimination of generalization

4.3.1. Bundle generalization

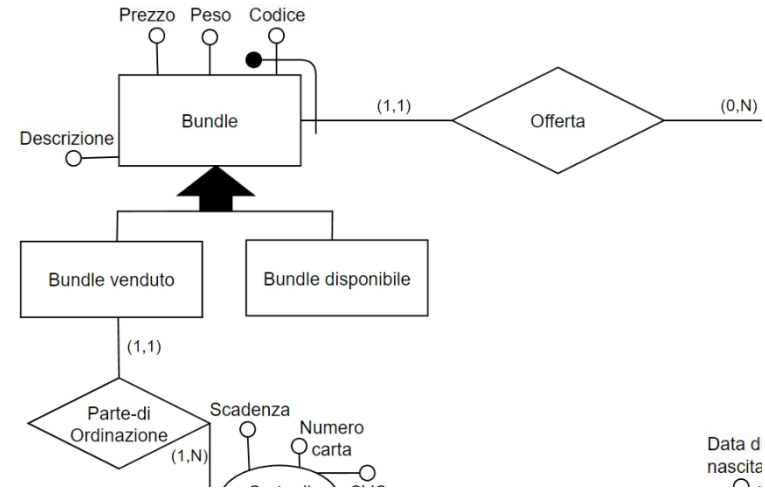

Figure 14. Bundle generalization before elimination

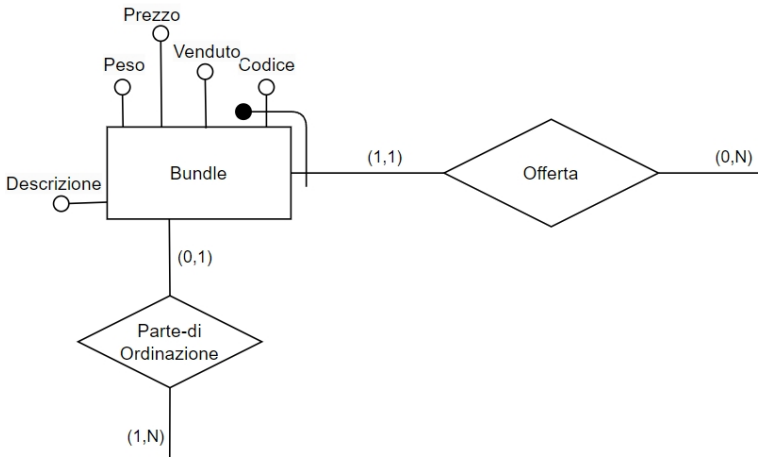

Figure 15. Bundle entities after elimination.

It is considered appropriate to apply the strategy of merging daughters into the parent (strategy 1) because although some operations differentiate between sold Bundle and available Bundle the two entities are characterized by the same attributes so there will be no memory waste by merging them due to null attributes.

4.3.2. Generalization Ordering

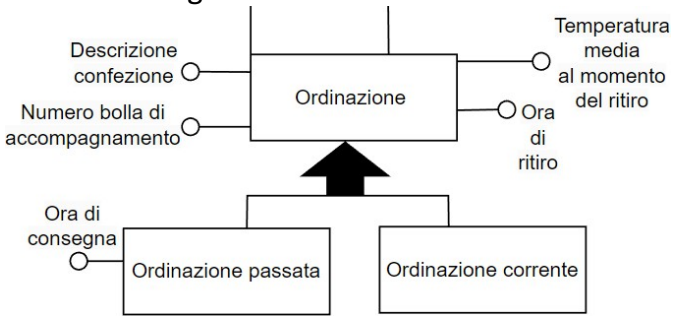

Figure 16. Generalization Ordering before deletion

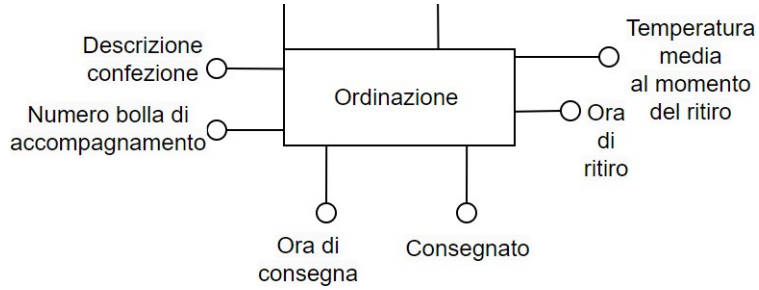

Figure 17. Entity Sorting after elimination

Taking into consideration that most of the operations related to the entities in question do not differentiate between the occurrences of the two daughters of the generalization, it is considered appropriate to adopt the strategy of merging the daughters into the parent (strategy 1).

Therefore, it is deemed necessary to add an attribute (Delivered) to distinguish between the two types of occurrences. The NULL value, if any, of the Delivery Time attribute is then contextualized by the Delivered attribute; the latter allows us to know whether the delivery has not yet been made (and therefore the time of delivery is unknown) or whether the delivery has completed but the Delivery Time information is absent.

4.3.3. Generalization Order Booked

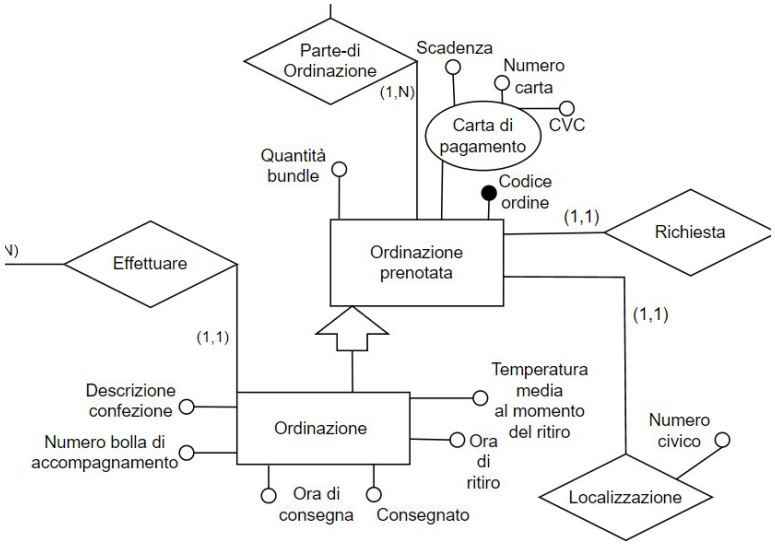

Figure 18. Generalization Ordering booked before deletion

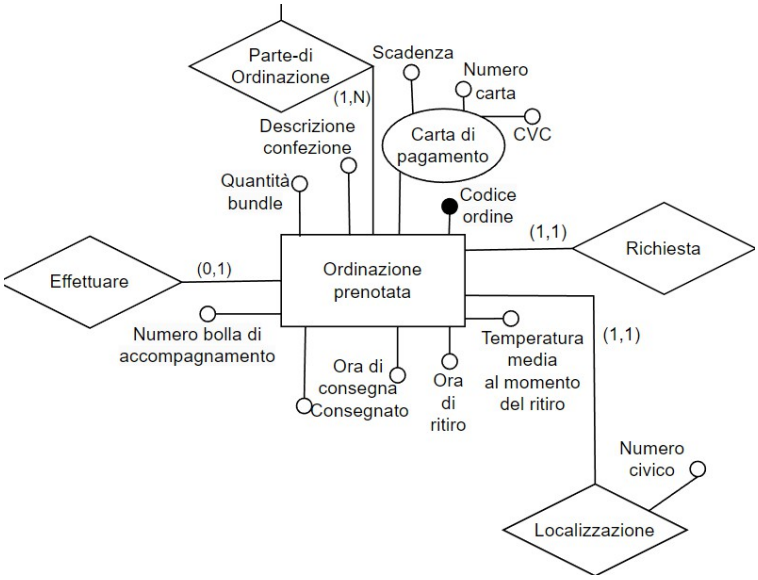

Figure 19. Entity Ordering after elimination

It is considered appropriate to apply the strategy of merging daughters into the parent of the generalization (strategy 1). Indeed, we note the absence of operations that differentiate occurrences between the parent entity and the daughter entity. Even the operation of counting the orders made by a rider effectively does not require access to the ordering entity since it is sufficient to count the occurrences of the association make related to the rider of interest.

The chosen strategy is also in line with the concept behind this generalization i.e., the evolution of the concept of ordering that when picked up by the rider acquires new attributes and associations.

4.4. Partitioning/Bundling Entities and Associations.

4.4.1. Removal of multivalue attribute Phone Number in Customer

Removal of the multivalue attribute Phone Number in Customer entity was performed by reification of the Phone Number in Entity attribute. The created entity was bound to Customer by means of an association whose cardinality reflected the initial situation.

4.4.2. Removal of compound and multivalue attribute Customer's Credit Card and Ordering.

Removing a multivalue attribute requires reifying the attribute to an entity; consequently, since Customer Entity Payment Card is a compound multivalue attribute, its component attributes could not simply be added to the Customer entity, as with a traditional compound attribute, but instead had to be reified.

Given the presence of the Payment Card Entity instead of choosing the traditional route.

Also for the elimination of the Payment Card compound attribute of the Ordering entity, it was decided not to use the traditional method of merging the component attributes into the parent entity because, given the presence of the Payment Card Entity reified by the eponymous attribute

of Customer, it was deemed more appropriate to link the Payment Card Entity not only to Customer but also to Ordering. The two different associations have different cardinalities to reflect different initial situations.

4.5. Choice of primary identifiers

4.5.1. Introducing new identifier in Client

The Customer entity has two different candidate identifiers: Tax Code and Username.

The choice for the primary identifier falls on Username because the requirements call for the Customer to be able to access the platform via username and password. The presence of an operation that distinguishes customer occurrences based on the Username attribute makes it more than suitable for the role of main identifier.

4.5.2. Principal identifier choice in Rider

The Rider entity has two different candidate identifiers: Tax Code, IBAN.

Despite the presence of the candidate identifiers Tax Code and IBAN, it is preferred to introduce another attribute, Rider Code, as it is considered more fitting to the identification of a Rider type user for a service such as the one modeled.

4.5.3. Principal identifier choice in Fisherman

The Fisherman entity has three different candidate identifiers: Tax ID and VAT number, IBAN.

The three are equally preferable from the point of view of the requirements for the main identifier (be non-null, consist of a single attribute, be internal identifiers). The VAT ID number attribute is preferred as it is considered to be more fitting to the identification of a business activity.

4.6. Final restructured schema

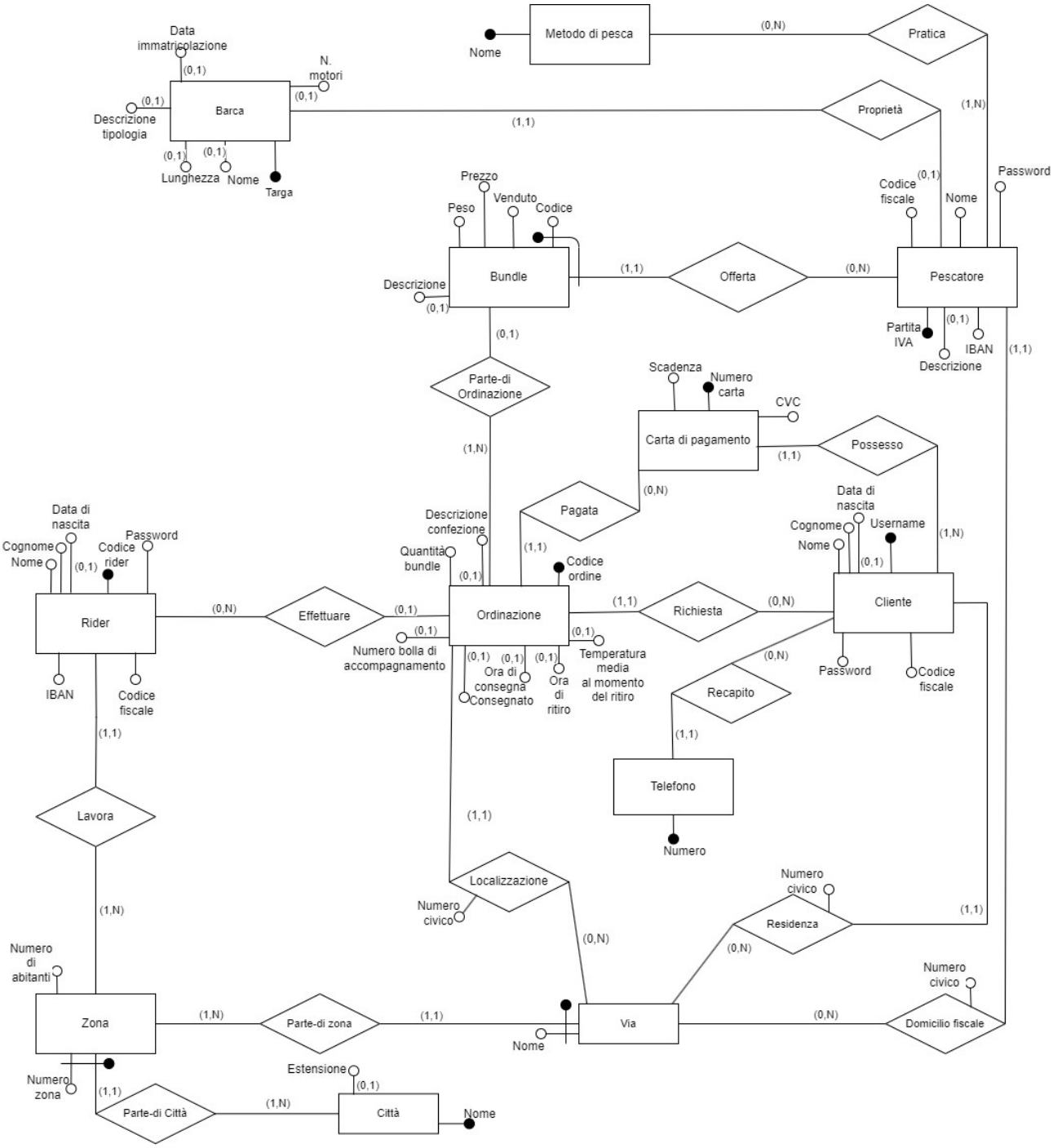

Figure 20. ER Schema Restructured

4.7. Logical Scheme

The following conventions are observed in the schema:

- Underlined solid line used for attributes that are part of the primary identifier;
- Underlined with dashed line used for attributes that represent referential integrity constraints;
- Underlined with double continuous line for attributes that perform both primary key and referential integrity constraint functions; and
- With the symbol * (asterisk) used to indicate attributes that can take null value .

Fisherman(VAT#, Street, Area, City, NumberCity, Tax Code, IBAN, Name, Password, Description*),
MethodOfFishing(Name**)**
Practice(MethodOfFishing, Fisherman)
Boat(License plate, Owner, DateRegistration*, NMotors*, DescriptionTypology*, Length*, Name*)
Bundle(Code, Fisherman, Ordering*, Weight, Price, Sold, Description*) **Ordering**(OrderCode, DescriptionPackage*, Delivered*, NumberBulletDelivery*, TemperatureAverageAtDelivery*, PickupTime*, DeliveryTime*, Customer, Street, Area, City, NumberCivic, CardPayment, Rider*, QuantityBundle)
Customer(Username, BirthDate*, LastName, FirstName, Password, Tax Code, Street, Area, City, NumberCivic)
CardPayment(Card Number, CVC, Expiration, Customer)
Phone(Number, Customer)
Rider(CodeRider, Area, City, Password, IBAN, LastName, FirstName, DateBirth*, TaxNumber)
City(Name, Extension) **Zone**(NumberZone, City,
NumberAbiters) **Street**(Name, Zone, City)

4.8. Documentation of the logic schema

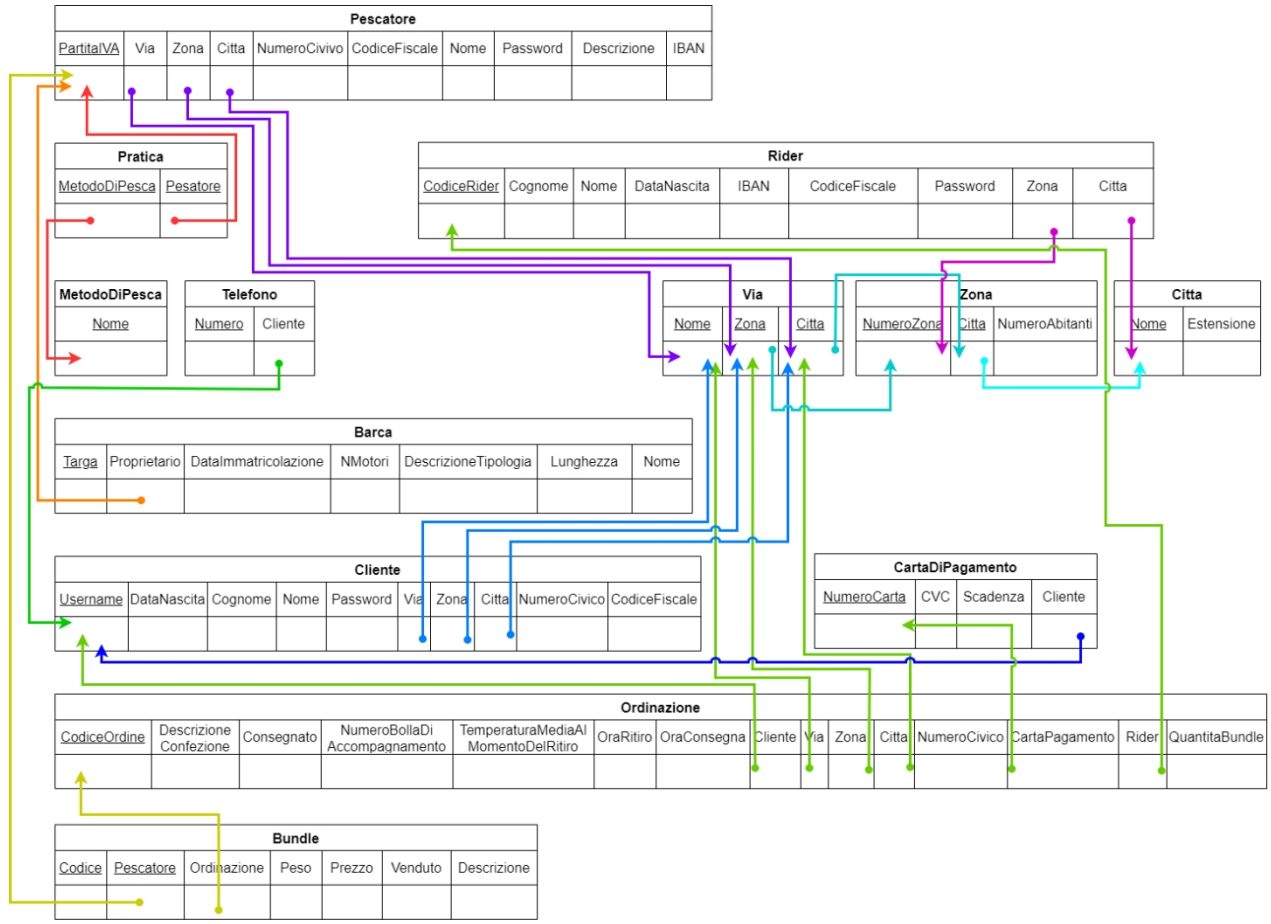

Figure 21. Logic schema

5. Normalization

Workpackage	Task	Responsible
WP3	Normalization	Cirillo Francesco Pio

5.1. Conceptual Scheme Considerations

Before approaching the verifications of Normal Forms based on the logical schema, some considerations on the conceptual schema are presented in order to verify the quality of the schema. It is pointed out that we are talking at this time about the post-restructured ER schema; in fact, it would not make sense to talk about normal forms for the original ER, which, presenting compound and multivalued attributes, certainly does not respect the first normal form. The restructured ER is therefore definitely in 1NF.

Assuming preliminarily that it is possible to consider each entity and each association as a relation we present the following considerations.

5.1.1. Normalization checks on entities.

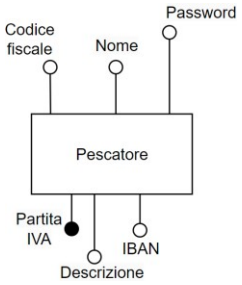

Figure 22. Entity Fisherman

Let us analyze the entity Fisherman, consider specifically its attributes.

It turns out to be trivial to observe that the entity is in Second Normal Form since its identifier consists of a single attribute.

We note that all attributes are functionally dependent on the entity's identifier: VAT ID. There are also other functional dependencies: all attributes depend on IBAN (candidate key) and all attributes depend on Tax Code (candidate key).

From the identified dependencies, it can be seen both that *there are no non-key attributes that transitively depend on the key* (3NF) and that *every determinant of the functional dependencies present is a candidate key* (BCNF).

By means of similar reasoning, it is possible to analyze all entities in the ER and prove that the ER as a whole is in BCNF.

5.1.2. Normalization checks on associations

The absence of nonbinary associations is shown. Every binary association is, surely, in BCNF, consequently all associations in the ER are in BCNF.

5.2. First Normal Form

The Database appears to meet the First Normal Form criteria in that it consists of Relationships that meet the First Normal Form criteria because they *are devoid of compound attributes and multivalued attributes*.

In fact, during the Restructuring of the Conceptual Schema, compound attributes and multivalued attributes were removed which resulted in a Logical Schema devoid of them.

5.3. Second Normal Form

In the following, the Second Normal Form of the Database is verified by analyzing the individual relationships that comprise it.

Second Normal Form is verified for a relation if it is in First Normal Form and if every non-primal attribute (attributes that do not belong to the primary key) has a full functional dependence on the key of the relation, that is, there are no nonkey attributes that partially depend on the key. It follows that relationships in First Normal Form that have the primary key composed of only one attribute are in Second Normal Form. This is the case for the following relationships:

- Fisherman;
- Rider;
- Phone;
- City;
- Boat;
- Client;
- CardPayment;
- Ordering.

It is also inferred from the definition of Second Normal Form that relations, in First Normal Form, lacking non-prime attributes meet the criteria for Second Normal Form. This is the case for the following relations:

- Practice;
- Street.

It then remains to check whether the Zone and Bundle relationships are in Second Normal Form.

Zone(NumberZone, City, NumberInhabitants)

NumberZona City→ NumberAbiters

The only non-first attribute of the Zone relation is NumberAbiters, this is bound by a full functional dependency to the primary key of the relation. This is because removing any primary key attribute from the dependency loses its validity. Thus we infer that Zone is a relation in Second Normal Form.

Bundle(Code, Fisherman, Ordering, Weight, Price, Sold)

Code Fisherman→ Ordering Code
Fisherman→ Weight Code
Fisherman→ Price Code
Fisherman→ Sold

We note that there are no functional dependencies other than trivial ones in fact the non-primitive attributes of the Bundle relation are bound by full functional dependencies to the primary key of the relation. Thus we infer that Bundle is a relation in Second Normal Form.

All relationships are in Second Normal Form so the Database is in Second Normal Form.

5.4. Third Normal Form

In the following, the Third Normal Form of the Database is verified by analyzing the individual relationships that make up the Database.

Third Normal Form is verified for a relation if it is in Second Normal Form and if for each nontrivial functional dependency X→ A defined on it at least one of the following conditions is verified:

- *X contains a candidate key K of the relation;*
- *A belongs to at least one candidate key in the relation.*

Customer(Username, DateBorn, LastName, FirstName, Password, TaxCode, Street, Area, City, NumberCivic)

It is highlighted that all attributes are functionally dependent on the Username attribute, which is in fact the primary key of the relationship. It is also shown that all attributes are functionally dependent on the attribute TaxCode which is in fact candidate key of the relationship.

It can be seen that the definition of Third Normal Form is valid for the Customer relation in fact all determinants of the identified functional dependencies are keys of the relation. It also turns out to be true that *there are no transitive dependencies of the form K→ A, where K is the key and there is another set of attributes X, not key, with dependencies K→ X and X→ A.*

Using a similar procedure, adherence to the Third Normal Form criteria was verified for all other relationships in the Database, however, it is decided not to attach the procedures performed in order not to burden the documentation given the similarity to what has already been proposed with the Customer relationship.

5.5. Normal Form of Boyce and Code

In the following we verify the Boyce and Code Normal Form of the Database by analyzing the individual relationships that comprise it.

The Boyce and Code Normal Form is verified for a relation if it is in First Normal Form and if for every nontrivial functional dependency X→ A defined on it, X contains a candidate key K of the relation, i.e., X is superkey of the relation.

It can be shown that if a relation has only one key then the Third Normal Form and the Boyce and Code Normal Form coincide. All relations in the database except Fisherman, Customer and Rider have only one candidate key, so we can conclude that, having already shown that they are in Third Normal Form, these are in Boyce and Code Normal Form.

With reference to the Customer relation we note that, observing the dependencies described in the previous paragraph, all determinants are candidate keys of the relation; it follows that Customer is in Normal Form of Boyce and Code.

Reasoning similar to that carried out for Customer allows us to show that Rider and Angler are also in Normal Form of Boyce and Code.

It is concluded that the database is in Normal Form of Boyce and Code since all component relationships are in Normal Form of Boyce and Code.

6. Script Creating and Populating Database

Workpackage	Task	Responsible
WP2	SQL: Script creation and population	Fasolino Alessandra

In the Query Tool in the postgres database:

```
1  CREATE DATABASE fish2home;
```

Creating Relationships and Populating Relationships there is the creation of triggers, but this has been omitted here as it is presented in the appropriate section.
Emphasis is placed on some relevant points such as:

- row16 - Creation of a domain to facilitate entry of the Password attribute;
- line 18/37 - Three different methodologies for establishing a primary key;
- line 58/65 - Two different ways to establish a referential integrity constraint, including inserting a reaction policy to the second;
- line73 - Use of default values;
- line118/148 - Use of constructs to specify constraints .

In the Query Tool in the fish2home database:

```
1  DROP TABLE IF EXISTS Pratica CASCADE;
2  DROP TABLE IF EXISTS Barca CASCADE;
3  DROP TABLE IF EXISTS Pescatore CASCADE;
4  DROP TABLE IF EXISTS MetodoDiPesca CASCADE;
5  DROP TABLE IF EXISTS Via CASCADE;
6  DROP TABLE IF EXISTS Zona CASCADE;
7  DROP TABLE IF EXISTS Citta CASCADE;
8  DROP TABLE IF EXISTS Bundle CASCADE;
9  DROP TABLE IF EXISTS Ordinazione CASCADE;
10 DROP TABLE IF EXISTS Cliente CASCADE;
11 DROP TABLE IF EXISTS Telefono CASCADE;
12 DROP TABLE IF EXISTS CartaPagamento CASCADE;
13 DROP TABLE IF EXISTS Rider CASCADE;
14 DROP DOMAIN IF EXISTS PWD CASCADE;
15
16 CREATE DOMAIN PWD VARCHAR(70) NOT NULL;
17
18 CREATE TABLE Citta (
19     Nome VARCHAR(30) PRIMARY KEY,
20     Estensione NUMERIC(5, 2)
21 );
22
23 CREATE TABLE Zona (
24     NumeroZona INT,
25     Citta VARCHAR(30),
26     NumeroAbitanti INT NOT NULL,
27     PRIMARY KEY(NumeroZona, Citta),
28     FOREIGN KEY(Citta) REFERENCES Citta(Nome)
29 );
30
31 CREATE TABLE Via (
32     Nome VARCHAR(30),
33     Zona INT,
34     Citta VARCHAR(30),
35     CONSTRAINT localizzazione PRIMARY KEY(Nome, Zona, Citta),
36     FOREIGN KEY(Zona, Citta) REFERENCES Zona(NumeroZona, Citta)
37 );
```

```
38
39  CREATE TABLE Pescatore (
40      PartitaIVA CHAR(11),
41      Via VARCHAR(30) NOT NULL,
42      Zona INT NOT NULL,
43      Citta VARCHAR(30) NOT NULL,
44      NumeroCivico INT NOT NULL,
45      CodiceFiscale CHAR(16) NOT NULL,
46      Nome VARCHAR(30) NOT NULL,
47      pw PWD,
48      Descrizione VARCHAR(300),
49      IBAN CHAR(27) UNIQUE NOT NULL,
50      PRIMARY KEY(PartitaIVA),
51      FOREIGN KEY(Via, Zona, Citta) REFERENCES localizzazione
52  );
53
54  CREATE TABLE MetodoDiPesca (
55      Nome VARCHAR(30) PRIMARY KEY
56  );
57
57
58  CREATE TABLE Pratica (
59      MetodoDiPesca VARCHAR(30)
60          REFERENCES MetodoDiPesca(Nome),
61      Pescatore CHAR(11),
62      PRIMARY KEY(MetodoDiPesca, Pescatore),
63      FOREIGN KEY(Pescatore) REFERENCES Pescatore(PartitaIVA)
64          ON DELETE CASCADE ON UPDATE CASCADE
65          DEFERRABLE INITIALLY DEFERRED
66  );
67
68  CREATE TABLE Barca (
69      Targa CHAR(8),
70      Proprietario CHAR(11) UNIQUE NOT NULL,/*un pescatore può possedere una sola barca*/
71      DataImmatricolazione DATE,
72      NMotori INT,
73      DescrizioneTipologia VARCHAR(300) DEFAULT'peschereccio',
74      Lunghezza NUMERIC(4, 2),
75      Nome VARCHAR(30),
76      PRIMARY KEY(Targa),
77      FOREIGN KEY(Proprietario) REFERENCES Pescatore(PartitaIVA)
78  );
79
80  CREATE TABLE Rider (
81      CodiceRider CHAR(3),
82      Cognome VARCHAR(30) NOT NULL,
83      Nome VARCHAR(30) NOT NULL,
84      Zona INT NOT NULL,
85      Citta VARCHAR(30) NOT NULL,
86      DataNascita DATE,
87      pw PWD,
88      IBAN CHAR(27) UNIQUE NOT NULL,
89      CodiceFiscale CHAR(16) UNIQUE NOT NULL,
90      PRIMARY KEY(CodiceRider),
91      FOREIGN KEY(Zona, Citta) REFERENCES Zona(NumeroZona, Citta)
92  );
```

```
93
94  CREATE TABLE Cliente (
95      Username VARCHAR(30),
96      DataNascita DATE,
97      Cognome VARCHAR(30) NOT NULL,
98      Nome VARCHAR(30) NOT NULL,
99      pw PWD,
100     CodiceFiscale CHAR(16) UNIQUE NOT NULL,
101     Via VARCHAR(30) NOT NULL,
102     Zona INT NOT NULL,
103     Citta VARCHAR(30) NOT NULL,
104     NumeroCivico INT NOT NULL,
105     PRIMARY KEY(Username),
106     FOREIGN KEY(Via, Zona, Citta) REFERENCES Via(Nome, Zona, Citta)
107 );
108
109 CREATE TABLE CartaPagamento (
110     NumeroCarta VARCHAR(16),
111     CVC CHAR(3) NOT NULL,
112     Scadenza DATE NOT NULL,
113     Cliente VARCHAR(30) NOT NULL,
114     PRIMARY KEY(NumeroCarta),
115     FOREIGN KEY(Cliente) REFERENCES Cliente(Username)
116 );
117
118 CREATE TABLE Ordinazione (
119     CodiceOrdine CHAR(5),
120     DescrizioneConfezione VARCHAR(300) CHECK
121         (rider IS NOT NULL OR DescrizioneConfezione IS NULL),
122     Consegnato BOOLEAN  CHECK
123         (rider IS NOT NULL OR consegnato = 'false'),
124     NumeroBollaDiAccompagnament INT  CHECK
125         (rider IS NOT NULL OR NumeroBollaDiAccompagnament IS NULL),
126     TemperaturaMediaAlMomentoDelRitiro NUMERIC(4, 2)  CHECK
127         (rider IS NOT NULL OR TemperaturaMediaAlMomentoDelRitiro IS NULL),
128     OraRitiro TIMESTAMP CHECK
129         (rider IS NOT NULL OR OraRitiro IS NULL),
130     OraConsegna TIMESTAMP CHECK /*se l'ordine non è stato consegnato non può essere conosciuto l'orario di consegna*/
131         ((consegnato = 'true' OR OraConsegna IS NULL)
132         AND (rider IS NOT NULL OR OraRitiro IS NULL)
133         AND (OraConsegna IS NULL OR OraConsegna > OraRitiro)),
134     Cliente VARCHAR(30) NOT NULL,
135     Via VARCHAR(30) NOT NULL,
136     Zona INT NOT NULL,
137     Citta VARCHAR(30) NOT NULL,
138     NumeroCivico INT NOT NULL,
139     CartaPagamento VARCHAR(16) NOT NULL,
140     Rider VARCHAR(30), /*se il l'ordine non è stato affidato a un rider allora l'orario di ritiro e le altre
141                         informazioni associate non possono essere conosciute*/
142     QuantitaBundle INT NOT NULL DEFAULT 0 CHECK (QuantitaBundle >= 0),
143     PRIMARY KEY(CodiceOrdine),
144     FOREIGN KEY(CartaPagamento) REFERENCES CartaPagamento(NumeroCarta),
145     FOREIGN KEY(Via, Zona, Citta) REFERENCES Via(Nome, Zona, Citta),
146     FOREIGN KEY(Rider) REFERENCES Rider(CodiceRider),
147     FOREIGN KEY(Cliente) REFERENCES Cliente(Username)
148 );
149
```

```sql
CREATE TABLE Bundle (
    Codice CHAR(5),
    Pescatore CHAR(11) NOT NULL,
    Ordinazione CHAR(50) CHECK (
        (Venduto = true AND Ordinazione IS NOT NULL) OR
        (Venduto = false AND Ordinazione IS NULL)),
    Peso NUMERIC(4, 2) NOT NULL CHECK (Peso > 0),
    Prezzo NUMERIC(5, 2) NOT NULL CHECK (Prezzo > 0),
    Venduto BOOLEAN NOT NULL, /*se l'ordine non è stato venduto non si può sapere
    l'ordinazione associata ma se è venduto deve essere conosciuta l'ordinazione nel quale è inserito*/
    Descrizione VARCHAR(50),
    PRIMARY KEY(Codice, Pescatore),
    FOREIGN KEY(Pescatore) REFERENCES Pescatore(PartitaIVA),
    FOREIGN KEY(Ordinazione) REFERENCES Ordinazione(CodiceOrdine)
);

CREATE TABLE Telefono (
    Numero VARCHAR(10) PRIMARY KEY,
    Cliente VARCHAR(30) NOT NULL,
    FOREIGN KEY(Cliente) REFERENCES Cliente(Username)
);
/*INIZIA LA FASE DI POPOLAMENTO*/
/*segue la creazione di tuple della relazione pescatore*/
INSERT INTO citta(nome, estensione)
    VALUES( 'Positano', 8.65);
INSERT INTO citta(nome, estensione)
    VALUES( 'Praiano', 2.67);
INSERT INTO citta(nome, estensione)
    VALUES( 'Furore', 1.88);
INSERT INTO citta(nome, estensione)
    VALUES( 'Conca dei Marini', 1.13);

/*segue la creazione di tuple della relazione zona*/
INSERT INTO zona(numeroZona, citta, numeroAbitanti)
    VALUES( 1, 'Positano', 1860);
INSERT INTO zona(numeroZona, citta, numeroAbitanti)
    VALUES( 2, 'Positano', 1860);
INSERT INTO zona(numeroZona, citta, numeroAbitanti)
    VALUES( 1, 'Praiano', 1977);
INSERT INTO zona(numeroZona, citta, numeroAbitanti)
    VALUES( 1, 'Furore', 681);
INSERT INTO zona(numeroZona, citta, numeroAbitanti)
    VALUES( 1, 'Conca dei Marini', 677);

/*segue la creazione di tuple della relazione via*/
INSERT INTO via(nome, zona, citta)
    VALUES ('Via Cristoforo Colombo', 2, 'Positano');
INSERT INTO via(nome, zona, citta)
    VALUES ('Via Corvo', 1, 'Positano');
INSERT INTO via(nome, zona, citta)
    VALUES ('Via Gavitella', 1, 'Praiano');
INSERT INTO via(nome, zona, citta)
    VALUES ('Via Antico Seggio', 1, 'Praiano');
INSERT INTO via(nome, zona, citta)
    VALUES ('Via Aldo Moro', 1, 'Furore');
INSERT INTO via(nome, zona, citta)
    VALUES ('Via Croce', 1, 'Furore');
INSERT INTO via(nome, zona, citta)
    VALUES ('Via delle Querce', 1, 'Conca dei Marini');
INSERT INTO via(nome, zona, citta )
    VALUES ('Via Liscia', 1, 'Conca dei Marini');

/*segue la creazione di tuple della relazione metodo di pesca*/
INSERT INTO MetodoDiPesca(nome)
    VALUES('Reti a strascico');
INSERT INTO MetodoDiPesca(nome)
    VALUES('Pesca con palangari');
INSERT INTO MetodoDiPesca(nome)
    VALUES('Pesca con reti a strascico');
INSERT INTO MetodoDiPesca(nome)
    VALUES('Pesca con la canna');
INSERT INTO MetodoDiPesca(nome)
    VALUES('Pesca con nasse');
```

```sql
/*segue la creazione di tuple della relazione pratica*/
begin;
INSERT INTO pratica(metododipesca, pescatore)
    VALUES('Reti a strascico', '01234567890');
INSERT INTO pratica(metododipesca, pescatore)
    VALUES('Pesca con nasse', '01234567890');
INSERT INTO pratica(metododipesca, pescatore)
    VALUES('Pesca con palangari', '98765432109');
INSERT INTO pratica(metododipesca, pescatore)
    VALUES('Pesca con nasse', '98765432109');
INSERT INTO pratica(metododipesca, pescatore)
    VALUES('Pesca con reti a strascico', '11223344556');
INSERT INTO pratica(metododipesca, pescatore)
    VALUES('Pesca con la canna', '11223344556');
INSERT INTO pratica(metododipesca, pescatore)
    VALUES('Pesca con la canna', '01107027890');
INSERT INTO pratica(metododipesca, pescatore)
    VALUES('Pesca con la canna', '98765200602');
INSERT INTO pratica(metododipesca, pescatore)
    VALUES('Pesca con la canna', '11223342024');
/*segue la creazione di tuple della relazione pescatore*/
INSERT INTO pescatore(partitaIVA, via, zona, citta, numeroCivico, codiceFiscale, nome, pw, descrizione, iban)
    VALUES( '01234567890', 'Via Cristoforo Colombo', 2, 'Positano', 31, 'RSSMRA80A01H501Q',
           'F.lli Rossi', 'mariorossiP2',
           'Famiglia di pescatori della costiera amalfitana che offre solo prodotti ittici di primissima qualità',
           'IT12X3456789012345678901234');
INSERT INTO pescatore(partitaIVA, via, zona, citta, numeroCivico, codiceFiscale, nome, pw, descrizione, iban)
    VALUES( '98765432109', 'Via Antico Seggio', 1, 'Praiano', 25, 'FRRGNN85A01H123X',
           'Mare Chiaro', 'giovanniferraroP1',
           'pescatore amalfitano appassionato di barche', 'IT98Y34567890123456789012345');
INSERT INTO pescatore(partitaIVA, via, zona, citta, numeroCivico, codiceFiscale, nome, pw, descrizione, iban)
    VALUES( '11223344556', 'Via Aldo Moro', 1, 'Furore', 14, 'MSSNNA81F01H331Q', 'Non solo pesce',
           'annamossaF1', 'premio miglior pescatrice donna in Italia 2022',
           'IT76Z3456789012345678901234');
INSERT INTO pescatore(partitaIVA, via, zona, citta, numeroCivico, codiceFiscale, nome, pw, iban)
    VALUES( '01107027890', 'Via Corvo', 1, 'Positano', 11, 'CNTMTE86A12F205W',
           'Pescheria marini', 'matteoContiPescheria', 'IT12X3456789012345678902024');
INSERT INTO pescatore(partitaIVA, via, zona, citta, numeroCivico, codiceFiscale, nome, pw, iban)
    VALUES( '98765200602', 'Via delle Querce', 1, 'Conca dei Marini', 2, 'DLCSRA93C13F205M',
           'Mare', 'SaraDeLucaaa8', 'IT98Y34567890123456789902023');
INSERT INTO pescatore(partitaIVA, via, zona, citta, numeroCivico, codiceFiscale, nome, pw, iban)
    VALUES( '11223342024', 'Via Liscia', 1, 'Conca dei Marini', 41, 'MRTDVD85R15F205N', 'Pescheria felice',
           'DavideMaretto', 'IT76Z3456789012345678902022');
commit;

/*segue la creazione di tuple della relazione barca*/
INSERT INTO Barca(targa, proprietario, dataImmatricolazione, NMotori, descrizioneTipologia, lunghezza, nome)
    VALUES('BA123AB', '01234567890', '2020-08-05', 2, 'Barca da traina', 9.00, 'Santa Maria');
INSERT INTO Barca(targa, proprietario, dataImmatricolazione, NMotori, descrizioneTipologia, lunghezza, nome)
    VALUES('MI345GH', '11223344556', '2022-07-15', 2, 'Barca da traina', 10.00, 'Sofia Benedetta');
INSERT INTO Barca(targa, proprietario, dataImmatricolazione, NMotori, descrizioneTipologia, lunghezza, nome)
    VALUES('TO567OP', '98765432109', '2019-12-20', 1, 'Barca da pesca cabinata', 8.00, 'Santa Marianna');
INSERT INTO Barca(targa, proprietario, dataImmatricolazione, NMotori, lunghezza, nome)
    VALUES('TO567AF', '01107027890', '2019-12-20', 1,  7.50, 'Luisella');

```

```
/*segue la creazione di tuple della relazione cliente*/
INSERT INTO cliente(username, dataNascita, cognome, nome, pw, codiceFiscale, via, zona, citta, numeroCivico)
    VALUES('FrancescoDeAngelis', '1984-07-02', 'De Angelis', 'Francesco',
           '$2y$10$3Bzws0AvwI8goVnid6hsCe7sTws5aED8p385.BcXpYXMvsfKwA0/q', 'DNGFNC84B07H412K',
           'Via Corvo', 1, 'Positano', 80);
INSERT INTO cliente(username, dataNascita, cognome, nome, pw, codiceFiscale, via, zona, citta, numeroCivico)
    VALUES('AlessiaGiuliani', '1976-10-23', 'Giuliani', 'Alessia',
           '$2y$10$weWgG.Rk8ouv2TlJ2pBADuXikicf.JyFYqiI9M4yETNDeXbyRoh8C',
           'GLNLSS76R23H804L', 'Via Croce', 1, 'Furore', 12);
INSERT INTO cliente(username, dataNascita, cognome, nome, pw, codiceFiscale, via, zona, citta, numeroCivico)
    VALUES('Conti1992', '1992-12-08', 'Conti', 'Laura',
           '$2y$10$OUEOMFhcD1ZjWC587PxKCeKsjL9XUPkG7zqHGKRcwF1K1QmE8tIxW',
           'CNTLRA92M12H315J', 'Via Gavitella', 1, 'Praiano', 1);
INSERT INTO cliente(username, dataNascita, cognome, nome, pw, codiceFiscale, via, zona, citta, numeroCivico)
    VALUES('GiovanniNovembre', '1987-11-29', 'Moretti', 'Giovanni',
           '$2y$10$EraILy1Nc3MDbaqy5GNmbexZjlY6omaTVtNaoaspSdjmTdxaqW8X6',
           'MRTGNN87S29H627G', 'Via Liscia', 1, 'Conca dei Marini', 27);
INSERT INTO cliente(username, dataNascita, cognome, nome, pw,codiceFiscale, via, zona, citta, numeroCivico)
    VALUES('Ferrari02', '1988-09-03', 'Ferrari', 'Simone',
           '$2y$10$xrwwvZohBYHNdVsyt.RKxOLV.IUZc.mpmV3haq6ybAV7A6s99CGL.',
           'FRRSMN89T08F205Q', 'Via Liscia', 1, 'Conca dei Marini', 2);
INSERT INTO cliente(username, dataNascita, cognome, nome, pw, codiceFiscale, via, zona, citta, numeroCivico)
    VALUES('MartinaRusso23', '1966-11-24', 'Russo', 'Martina',
           '$2y$10$Jw3WmG4UpSCKCn975/9QnOjnk39CCJYMFI5dEiRoWNj85QOW9e5RS',
           'RSSMRT96D47F205J', 'Via Gavitella', 1, 'Praiano', 30);
INSERT INTO cliente(username, dataNascita, cognome, nome, pw, codiceFiscale, via, zona, citta, numeroCivico)
    VALUES('Curso08', '1962-11-18', 'Caruso', 'Francesco',
           '$2y$10$4goGb8lIk96oyF9IH0LqnO1Ysni73OKc/0Nh84qKF0pIzmHWZ5vPq',
           'CRSFNC87R14F205T', 'Via Cristoforo Colombo', 2, 'Positano', 11);
INSERT INTO cliente(username, dataNascita, cognome, nome, pw, codiceFiscale, via, zona, citta, numeroCivico)
    VALUES('GiugliGalli60', '1960-01-09', 'Galli', 'Giulia',
           '$2y$10$O.6FyaGqh7ccWXmIMJfmnu4MrlCq6/UryT7nMzanECtOezIllB16a',
           'GLLGLI94M41F205L', 'Via Cristoforo Colombo', 2, 'Positano', 22);

/*segue la creazione di tuple della relazione rider*/
INSERT INTO rider(codiceRider, cognome, nome, zona, citta, dataNascita, pw, iban, codiceFiscale)
    VALUES('001', 'Romano', 'Annamaria', 1, 'Positano', '1988-08-05',
          '$2y$10$98oITI2DGNJfHaSy2MGgp.FtmxcZSYLgU7c8pXIloOpdSph2lIjEG',
          'IT34R3456789012345678901234', 'RMNNM73D03H736F');
INSERT INTO rider(codiceRider, cognome, nome, zona, citta, dataNascita, pw, iban, codiceFiscale)
    VALUES('002', 'Esposito', 'Marco', 2, 'Positano', '1998-05-12',
          '$2y$10$gFHVU3Io3B/1yY1SFX.bUOWJRmhNkzdSaD7TtdPiJNlyM.qxd.6Ba',
          'IT45T3456789012345678901234', 'SPTMRC80P18H929E');
INSERT INTO rider(codiceRider, cognome, nome, zona, citta, dataNascita, pw, iban, codiceFiscale)
    VALUES('003', 'Ferrara', 'Sofia', 1, 'Praiano', '1999-07-03',
          '$2y$10$XMytryYlNs.4PjuRCk5RdOSBNTwjoHxtyJkBXJoCqz0BBkU4diaqq',
          'IT56G3456789012345678901234', 'FRRSFA95M05H208D');
INSERT INTO rider(codiceRider, cognome, nome, zona, citta, pw, iban, codiceFiscale)
    VALUES('004', 'Costa', 'Matteo', 1, 'Furore', '$2y$10$xtPUoQIzh5fRthgEfn6GSueqE83IuksDmR5YymvBpequMp3/5SwHO',
          'IT12X3456789012345678901235', 'CSTMTT82H14H919B');
INSERT INTO rider(codiceRider, cognome, nome, zona, citta, dataNascita, pw, iban, codiceFiscale)
    VALUES('005', 'Bianchi', 'Luca', 1, 'Conca dei Marini', '2000-12-12',
          '$2y$10$2BnSl2he4rGypUUTHlB0fe9OG.chxhO9L5RQogP3/S3zIGsZH0Kde',
          'IT78F3456789012345678901234', 'BNCLCA90L21H123B');
INSERT INTO rider(codiceRider, cognome, nome, zona, citta, dataNascita, pw, iban, codiceFiscale)
    VALUES('006', 'Pioggia', 'Annarita', 1, 'Positano', '1982-08-05',
          '$2y$10$98oITI2DGNJfHaSy2MGgp.FtmxcZSYLgU7c8pXIloOpdSph2lhfg',
          'IT34R3456789012345678900711', 'PGNNR73D03H736F');

```

```sql
/*segue la creazione di tuple della relazione carta pagamneto*/
INSERT INTO CartaPagamento(NumeroCarta, CVC, Scadenza, Cliente)
    VALUES( '4556123456789012', '789', '2027-05-01', 'FrancescoDeAngelis');
INSERT INTO CartaPagamento(NumeroCarta, CVC, Scadenza, Cliente)
    VALUES( '4556123456789013', '781', '2027-08-01', 'AlessiaGiuliani');
INSERT INTO CartaPagamento(NumeroCarta, CVC, Scadenza, Cliente)
    VALUES( '4556123456780014', '731', '2026-08-01', 'Conti1992');
INSERT INTO CartaPagamento(NumeroCarta, CVC, Scadenza, Cliente)
    VALUES( '4550003456780015', '111', '2028-01-21', 'GiovanniNovembre');
INSERT INTO CartaPagamento(NumeroCarta, CVC, Scadenza, Cliente)
    VALUES( '2002110756789013', '331', '2027-02-01', 'Ferrari02');
INSERT INTO CartaPagamento(NumeroCarta, CVC, Scadenza, Cliente)
    VALUES( '2002200656789013', '331', '2027-02-20', 'MartinaRusso23');
INSERT INTO CartaPagamento(NumeroCarta, CVC, Scadenza, Cliente)
    VALUES( '4556123407110014', '349', '2026-03-01', 'Curso08');
INSERT INTO CartaPagamento(NumeroCarta, CVC, Scadenza, Cliente)
    VALUES( '4550003456780620', '433', '2028-04-21', 'GiugliGalli60');

/*segue la creazione di tuple della relazione ordinazione*/
INSERT INTO Ordinazione(CodiceOrdine, DescrizioneConfezione, Consegnato, NumeroBollaDiAccompagnament,
    TemperaturaMediaAlMomentoDelRitiro, OraRitiro, OraConsegna, Cliente, Via, Zona, Citta,
    NumeroCivico, CartaPagamento, Rider)
    VALUES('ORD00', 'Borse termiche', 'true', 40505001, 3.50,   '2024-05-05 10:00:00', '2024-05-05 12:30:00',
        'FrancescoDeAngelis', 'Via Antico Seggio', 1, 'Praiano', 36, '4556123456789012', '003');
INSERT INTO Ordinazione(CodiceOrdine, DescrizioneConfezione, Consegnato, NumeroBollaDiAccompagnament,
    TemperaturaMediaAlMomentoDelRitiro, OraRitiro, OraConsegna, Cliente, Via, Zona, Citta,
    NumeroCivico, CartaPagamento, Rider, QuantitaBundle)
    VALUES('ORD01', 'Contenitori refrigeranti','true', 40505002, 3, '2024-05-05 11:15:00', '2024-05-05 13:45:00',
        'AlessiaGiuliani', 'Via Croce', 1, 'Furore', 12, '4556123456789013', '004', 0);
INSERT INTO Ordinazione(CodiceOrdine, Consegnato, Cliente, Via, Zona, Citta, NumeroCivico,
    CartaPagamento, QuantitaBundle)
    VALUES('ORD02', 'false', 'Conti1992', 'Via delle Querce', 1, 'Conca dei Marini', 20, '4556123456780014',0);
INSERT INTO Ordinazione(CodiceOrdine, DescrizioneConfezione, Consegnato, NumeroBollaDiAccompagnament,
    TemperaturaMediaAlMomentoDelRitiro, OraRitiro, Cliente, Via, Zona, Citta,
    NumeroCivico, CartaPagamento, Rider, QuantitaBundle)
    VALUES('ORD03', 'Imballaggio sottovuoto','false', 40505112, 6, '2024-06-12 11:15:00',
           'GiovanniNovembre', 'Via Gavitella', 1, 'Praiano', 22, '4550003456780015', '003', 0);
INSERT INTO Ordinazione(CodiceOrdine, DescrizioneConfezione, Consegnato, NumeroBollaDiAccompagnament,
    TemperaturaMediaAlMomentoDelRitiro, OraRitiro, OraConsegna, Cliente, Via, Zona, Citta,
    NumeroCivico, CartaPagamento, Rider, QuantitaBundle)
    VALUES('ORD04', 'Sacchetti con ghiaccio', 'true', 40505333, 3,  '2024-05-10 11:15:00', '2024-05-10 13:45:00',
           'AlessiaGiuliani', 'Via Cristoforo Colombo', 2, 'Positano', 32, '4556123456789013', '002', 0);

/*segue la creazione di tuple della relazione bundle*/
INSERT INTO Bundle(Codice , Pescatore , Ordinazione , Peso , Prezzo , Venduto, Descrizione)
    VALUES ('bn000', '01234567890', 'ORD00', 3.00, 40.00, 'true', 'adatto ai bambini');
INSERT INTO Bundle(Codice , Pescatore , Ordinazione , Peso , Prezzo , Venduto)
    VALUES ('bn001', '01234567890', 'ORD01', 1.00, 20.00, 'true');
INSERT INTO Bundle(Codice , Pescatore , Ordinazione , Peso , Prezzo , Venduto)
    VALUES ('bn002', '01234567890', 'ORD01', 3.00, 60.00, 'true');
INSERT INTO Bundle(Codice , Pescatore , Ordinazione , Peso , Prezzo , Venduto, Descrizione)
    VALUES ('bn003', '01234567890', 'ORD02', 5.00, 100.00, 'true', 'paranza');
INSERT INTO Bundle(Codice , Pescatore, Peso , Prezzo , Venduto)
    VALUES ('bn004', '01234567890', 5.00, 100.00, 'false');
INSERT INTO Bundle(Codice , Pescatore , Ordinazione , Peso , Prezzo , Venduto, Descrizione)
    VALUES ('bn000', '11223344556', 'ORD03', 1.00, 10.00, 'true', 'ottima per la pasta allo scoglio');
INSERT INTO Bundle(Codice , Pescatore , Ordinazione , Peso , Prezzo , Venduto, Descrizione)
    VALUES ('bn001', '11223344556', 'ORD03', 1.00, 10.00, 'true', 'ottima per la pasta allo scoglio');
INSERT INTO Bundle(Codice , Pescatore , Ordinazione , Peso , Prezzo , Venduto, Descrizione)
    VALUES ('bn002', '11223344556', 'ORD03', 1.50, 16.00, 'true', 'polipo e alici');
INSERT INTO Bundle(Codice , Pescatore , Ordinazione , Peso , Prezzo , Venduto, Descrizione)
    VALUES ('bn003', '11223344556', 'ORD04', 1.50, 16.00, 'true', 'polipo e alici');
INSERT INTO Bundle(Codice , Pescatore, Peso, Prezzo , Venduto, Descrizione)
    VALUES ('bn001', '98765432109', 1.50, 20.00, 'false', 'bistecche di pesce');

/*segue la creazione di tuple della relazione telefono*/
INSERT INTO Telefono( Numero, Cliente)
    VALUES('0891234567', 'FrancescoDeAngelis');
INSERT INTO Telefono( Numero, Cliente)
    VALUES('3459876543', 'Conti1992');
INSERT INTO Telefono( Numero, Cliente)
    VALUES('3398765432', 'FrancescoDeAngelis');
INSERT INTO Telefono( Numero, Cliente)
    VALUES('3317654321', 'GiovanniNovembre');
```

7. SQL Query

Workpackage	Task	Responsible
WP3	SQL: Query	Cirillo Francesco Pio

7.1. Query with aggregation and join operator: sales quality of fishermen

Select the name and VAT ID of fishermen who sell Orders averaged at a temperature above 5 degrees.

```
SELECT Pescatore.Nome, Pescatore.PartitaIVA
    FROM Pescatore
        JOIN Bundle ON Pescatore.PartitaIVA = Bundle.Pescatore
        JOIN Ordinazione ON Bundle.Ordinazione = Ordinazione.CodiceOrdine
    WHERE Bundle.Venduto
    GROUP BY Pescatore.PartitaIVA
    HAVING AVG(Ordinazione.TemperaturaMediaAlMomentoDelRitiro) > 5;
```

7.2. Complex nested query: Stressed rider search.

Selection of riders and associated zone and city that do not share the zone with any other riders and have delivered more than average. Designed to identify where to place new riders.
Note the presence of 3 different nested queries in the WHERE clause of the main query.
Complex interpretation is present in the first and third as Binding passing occurs for these.

```
SELECT nome, zona, citta
    FROM Rider AS r1
    WHERE NOT EXISTS (SELECT *
                      FROM Rider AS r2
                      WHERE r1.CodiceRider <> r2.CodiceRider AND
                          r1.Zona = r2.Zona AND
                          r1.Citta = r2.Citta)
        AND
        (SELECT AVG(ordinazioni_consegnate) as media_consegne
            FROM (SELECT rider, count(*) as ordinazioni_consegnate
                    FROM Ordinazione
                    WHERE Rider is not null
                    GROUP BY Rider))
        <
        (SELECT COUNT(*)
            FROM Ordinazione
            WHERE r1.CodiceRider = Ordinazione.Rider);
```

7.3. Set Query: Search for Customers ordering at addresses other than residence

Selects Customers living in the "Praiano1" zone who placed orders at another zone.

```
SELECT Username, Cliente.Nome, Cognome
    FROM Cliente, Via
    WHERE Cliente.Via = Via.Nome AND
        Cliente.Zona = Via.Zona AND
        Cliente.Citta = Via.Citta AND
        Via.Zona = 1 AND
        Via.Citta = 'Praiano'
INTERSECT
SELECT Username, Nome, Cognome
    FROM Cliente, Ordinazione
    WHERE Ordinazione.Cliente = Cliente.Username AND
        (Ordinazione.Zona <> 1 OR
        Ordinazione.Citta <> 'Praiano');
```

7.4. Possible Other Queries.

7.4.1. Search for worst fishermen in the most populous areas.

Extracts cities consisting of at least one populous area (NAbitants > 1000) for which the average Number of Bundles in each Order is greater than 2. Tuples are sorted in descending order based on average temperature at the time of pickup.

```
SELECT Ordinazione.Citta, AVG(Ordinazione.TemperaturaMediaAlMomentoDelRitiro)
        AS temperatura_media, SUM(Ordinazione.QuantitaBundle) AS numero_tot_bundle
    FROM Ordinazione, Zona
    WHERE (Zona.NumeroAbitanti > 1000 AND
        Ordinazione.Citta = Zona.Citta AND
        Ordinazione.Zona = Zona.NumeroZona)
    GROUP BY Ordinazione.Citta
    HAVING AVG(Ordinazione.QuantitaBundle) > 2
    ORDER BY AVG(Ordinazione.TemperaturaMediaAlMomentoDelRitiro) desc
```

8. Views

Workpackage	Task	Responsible
WP4	Views	Fasolino Alessandra

8.1. *CostOrders* View

This view allows easy visualization of the composition of orders. In fact, the following are displayed for each order: number of component bundles, cost and total weight.

```
CREATE VIEW CostoOrdinazioni(ordinazione, numero_bundle, costo_ordine, peso) as
    SELECT ordinazione, quantitaBundle as numero_bundle, sum(prezzo), sum(peso)
        FROM Bundle, Ordinazione
        WHERE venduto AND
            Bundle.ordinazione = codiceOrdine
        GROUP BY ordinazione, Ordinazione.codiceOrdine
        ORDER BY sum(prezzo) desc;
```

8.1.1. Query with View: Customer Ranking

This query displays all customers and the relative total cost of all their orders as well as the total weight of fish purchased. The tuples are sorted in descending order of expenditure in this query originated with the idea of identifying the "best customer."

```
SELECT cliente, sum(peso) AS peso_acquistato, sum(costo_ordine) AS costo_acquisti
    FROM CostoOrdinazioni JOIN Ordinazione ON
        (costoOrdinazioni.ordinazione = codiceordine)
    GROUP BY cliente
    ORDER BY costo_acquisti desc;
```

8.1.2. Query with View: Delivery Weight by City

The query allows to select very large cities (size>1.5Kmq) and the relative average weight of orders delivered in that city.

```
SELECT citta.nome, avg(peso) AS peso_acquistato
    FROM CostoOrdinazioni, Ordinazione, Citta
    WHERE costoOrdinazioni.ordinazione = codiceordine AND
        citta.nome = Ordinazione.citta
    GROUP BY citta.nome
    HAVING citta.estensione > 1.5;
```

9. Trigger

9.1. Trigger initialization: *Association (0,N) to (1,N)*

Workpackage	***Task***	***Responsible***
WP1	Trigger initialization/population database	Cirillo Francesco Pio

This trigger verifies at each insertion in the Fisherman relation and at each deletion from the Practice table that it is always verified that a Fisherman practices at least one type of fishing.

```
CREATE OR REPLACE FUNCTION almeno_una_pratica() RETURNS TRIGGER AS $$
BEGIN
    IF (EXISTS (SELECT PartitaIVA
                    FROM Pescatore
                    WHERE PartitaIVA NOT IN (SELECT Pescatore FROM Pratica))) THEN
        RAISE EXCEPTION 'ERRORE: ogni pescatore deve praticare almeno un tipo di pesca';
    END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_almeno_una_pratica
AFTER INSERT ON Pescatore
FOR EACH ROW EXECUTE PROCEDURE almeno_una_pratica();
CREATE TRIGGER trigger_almeno_una_pratica
AFTER DELETE ON Pratica
FOR EACH ROW EXECUTE PROCEDURE almeno_una_pratica();
```

It is emphasized that in order to correctly represent the minimum cardinality of associations such as: Work (on the Zone side), Part of-Ordering (on the Ordering side) and Possession (on the Customer side) would have required additional triggers to populate.

Given its significance it is decided to attach trigger_at least_one_practice.

9.2. Triggers for business constraints

Workpackage	*Task*	*Responsible*
WP4	Triggers for business constraints	Fasolino Alessandra

9.2.1. Trigger1: Calculate bundle quantity in an order.

This trigger automates the process of updating the QuantityBundle attribute of an Order by ensuring that it always matches the number of Bundles actually part of the Order.

```
CREATE OR REPLACE FUNCTION calcola_quantita_bundle() RETURNS TRIGGER AS $$
DECLARE
    new_quantita_bundle INT;
    old_quantita_bundle INT;
BEGIN
    SELECT COUNT(*) INTO new_quantita_bundle
        FROM Bundle
        WHERE Ordinazione = NEW.Ordinazione;

    UPDATE Ordinazione
        SET QuantitaBundle = new_quantita_bundle
        WHERE CodiceOrdine = NEW.Ordinazione;

    SELECT COUNT(*) INTO old_quantita_bundle
        FROM Bundle
        WHERE Ordinazione = OLD.Ordinazione;

    UPDATE Ordinazione
        SET QuantitaBundle = old_quantita_bundle
        WHERE CodiceOrdine = OLD.Ordinazione;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_calcola_quantita_bundle
AFTER INSERT OR DELETE OR UPDATE OF Ordinazione ON Bundle
FOR EACH ROW
EXECUTE PROCEDURE calcola_quantita_bundle();
```

9.2.2. Trigger2: Rider Coverage Check

This trigger ensures that a Rider can only be assigned Orders that are to be delivered to his or her Zone.

```
CREATE OR REPLACE FUNCTION controllo_copertura_rider() RETURNS TRIGGER AS $$
BEGIN
    IF (NEW.Rider not in (SELECT CodiceRider
                          FROM Rider
                          WHERE Rider.Zona = NEW.Zona AND
                              Rider.Citta = NEW.Citta)) THEN
        RAISE EXCEPTION 'ERRORE: i Rider possono solo accettare ordini della propria zona';
    END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_controllo_copertura_rider
AFTER INSERT OR UPDATE OF Rider ON OrdinazIone
FOR EACH ROW
EXECUTE PROCEDURE controllo_copertura_rider();
```

9.2.3. Trigger3: Payment Card Possession Check

This trigger verifies that all Orders are paid for using a payment card that actually corresponds to the Customer who placed the Order.

```
CREATE OR REPLACE FUNCTION controllo_possesso_carta_di_pagamento() RETURNS TRIGGER AS $$
BEGIN
    IF (NOT EXISTS (SELECT *
                    FROM CartaPagamento
                    WHERE CartaPagamento.NumeroCarta = NEW.CartaPagamento
                        AND CartaPagamento.Cliente = NEW.Cliente)) THEN
        RAISE EXCEPTION 'ERRORE: la carta di pagamento non è registrata per il cliente che ha effettuato questo ordine';
    END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_controllo_possesso_carta_di_pagamento
AFTER INSERT ON Ordinazione
FOR EACH ROW
EXECUTE PROCEDURE controllo_possesso_carta_di_pagamento();
```