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	

Anno accademico 2023-2024

Sommario

1. Descrizione della realtà di interesse

Titolo: **Fish2Home**

Fish2Home è un servizio di consegna del pescato del giorno che si occupa di assumere e gestire rider che prelevano il pescato (bundle di pesce) dal pescatore scelto dal cliente, e lo consegnano direttamente a casa. Fish2Home ha commissionato la progettazione e la realizzazione di un database a supporto di tale servizio.

Per poter ordinare il pesce, il cliente deve iscriversi e completare il proprio profilo utente con i dati anagrafici, i recapiti e il domicilio abituale, oltre allo username e la password per accedere al servizio. È richiesta anche la memorizzazione di almeno una carta di debito/credito. In generale il cliente potrebbe avere più di una carta di pagamento associata al suo profilo.

Esistono varie tipologie di pescatori che possono vendere pesce utilizzando questo servizio. Ovviamente sono di interesse le principali informazioni che caratterizzano il pescatore (la partita iva, il codice fiscale, il nome, una descrizione, la tipologia, l'indirizzo, ecc.). Il singolo pescatore ovviamente può possedere o meno un'unica barca (il cui tipo è da considerarsi un attributo) a seconda del tipo di pesca che esso effettua: Pesca a strascico, Pesca con palangari, Pesca con reti a strascico, Pesca con la canna, etc. Inoltre rispetto al pescato del giorno ogni pescatore offre un assortimento (Bundle) di pesci che dipende anche dalla tipologia di pesca che effettua e che lo caratterizza.

Nel caso trattato il cliente deve acquistare un bundle di pesci indicandone il peso di interesse, ma in ogni caso il servizio non offre la possibilità di fornire al cliente informazioni dettagliate sulla composizione del bundle di interesse a livello di ogni singolo pesce.

Gli ordini consistono essenzialmente in Bundle assegnati sulla base delle richieste, ma il sistema deve consentire di tracciare gli ordini effettuati dagli utenti e i bundle assegnati riportando per ogni bundle consegnato al cliente, le informazioni di pagamento, le quantità, il costo totale, e l'indirizzo di destinazione che può essere diverso dal domicilio dell'utente.

Ogni ordine deve essere recapitato a destinazione da un rider. Per organizzare le consegne, il sistema suddivide i rider in zone, assegnando ogni zona a uno o più rider. La grandezza della zona dipende dal numero di abitanti presenti. L'ordine da consegnare sarà assegnato a un rider appartenente alla zona in cui effettuare la consegna. Si vogliono tracciare i ritiri del pesce da parte del rider presso il pescatore e le consegne presso le abitazioni, riportando in particolare gli orari di ritiro e consegna. Il servizio interessa un'area geografica circoscritta in un'unica provincia.

1.1. Analisi della realtà di interesse

L'obiettivo del progetto è di realizzare un database per creare un servizio di vendita smart di bundle di pesce direttamente dai pescherecci al cliente tramite un rider.

Si tiene in considerazione che il servizio in questione è attivo nella zona della Costiera Amalfitana.

Si considera la presenza di varie tipologie di pescatori che possono vendere diverse specie di pesce organizzate in bundle catalogati a peso. Ogni pescatore può possedere al massimo una barca e può essere specializzato in più tipi di pesca (Pesca a strascico, Pesca con palangari, Pesca con reti a strascico, Pesca con la canna, ecc…). I clienti che sono iscritti al servizio possono comprare i bundle accedendo ad una descrizione e al peso. Le consegne vengono effettuate da rider.

Si è deciso di caratterizzare il progetto orientandosi in maniera più accurata alla modellazione del servizio di delivery, saranno modellati con accuratezza gli ordini, i clienti, i rider e i contenuti degli ordini, vale a dire i bundle di pesce proposti dai pescatori. Ogni pescatore può offrire dei bundle del proprio pescato, caratterizzando le sue proposte nel modo che ritiene possa meglio favorire le sue vendite. I clienti non potranno creare bundle personalizzati ma potranno solo scegliere tra quelli proposti per salvaguardare gli interessi dei pescatori.

Obiettivo del progetto è la corretta modellazione di tutto l'aspetto relativo alla vendita e all'acquisto, descrivendo in maniera meno accurata i dettagli relativi alla produzione della materia prima. Tecniche di pesca e imbarcazioni saranno comunque modellati ma solo per poter fornire più informazioni all'utente finale, non per garantire aderenza a determinate regole aziendali. Il pescatore dovrà garantire per sé che le informazioni che fornisce riguardo le proprie tecniche di pesca siano coerenti ai prodotti che offre.

2. Analisi delle specifiche

Workpackage	***Task***	***Responsabile***
WP0	Analisi delle specifiche	Intero Gruppo

2.1. Glossario dei termini

	Termine	**Descrizione**	**Sinonimi**	**Collegamenti**
1	Bundle	Pacchetto di pesci organizzato dal pescatore. Viene acquistato come parte di un'ordinazione.	pescato, assortimento	pescatore, ordine, barca
2	Pescatore	Colui che usa la piattaforma per vendere il suo pescato organizzato in bundle. Può effettuare diversi tipi di pesca con l'ausilio di una sola tipologia di imbarcazione.	-	bundle
3	Cliente	Colui che effettua ordini al pescatore per comprare bundle. Vive in una zona della sua Città di residenza.	utente	ordine, zona
4	Barca	Imbarcazione di proprietà del pescatore utilizzata per praticare uno o più metodi di pesca.	-	pescatore
5	Ordine	Acquisto di un cliente per uno o più bundle. Viene consegnato da un rider.	-	cliente, bundle, rider
6	Rider	Addetto alla consegna delle ordinazioni di pesce ai clienti residenti in una zona.	-	zona, ordine
7	Zona	Regione geografica alla quale sono assegnati uno o più rider sulla base della densità abitativa.	-	rider, ordine, cliente

Tabella 1. Glossario dei Termini

2.2. Strutturazione dei requisiti in frasi

2.2.1. Frasi di carattere generale

Fish2Home è un servizio di consegna del pescato del giorno che si occupa di assumere e gestire rider che prelevano il pescato dal pescatore scelto dal cliente, e lo consegnano direttamente a casa.

2.2.2. Frasi relative al cliente

Il cliente per accedere al servizio deve iscriversi e creare un profilo.

Il cliente per accedere al profilo deve indicare username e password.

Il profilo del cliente deve contenere username, password, dati anagrafici, i recapiti telefonici, il domicilio abituale e una o più carte di credito/debito.

Il cliente acquista un bundle di pesci indicandone il peso di interesse, potendo accedere alle informazioni relative.

2.2.3. Frasi relative al pescatore

Del pescatore si conoscono la partita iva, il codice fiscale, il nome, una descrizione, il tipo, l'indirizzo.

Il singolo pescatore può possedere o meno un'unica barca a seconda del metodo di pesca che esso effettua: Pesca a strascico, Pesca con palangari, Pesca con reti a strascico, Pesca con la canna, ecc .

Il singolo pescatore può effettuare uno o più tipi di pesca.

2.2.4. Frasi relative all' ordine

Gli ordini consistono in Bundle assegnati sulla base delle richieste dei clienti.

Il sistema deve consentire di tracciare gli ordini effettuati dagli utenti e i bundle assegnati.

Ogni ordine contiene informazioni per ogni bundle consegnato al cliente: le informazioni di pagamento, le quantità, il costo totale, e l'indirizzo di destinazione.

L'indirizzo di destinazione può essere diverso dal domicilio dell'utente.

2.2.5. Frasi relative al rider

Ogni rider è assegnato a una zona, a ogni zona possono essere assegnati più rider.

La grandezza della zona dipende dal numero di abitanti presenti.

2.2.6. Frasi relative alla consegna di ordini

Ogni ordine deve essere recapitato a destinazione da un rider.

L'ordine da consegnare sarà assegnato a un rider appartenente alla zona in cui effettuare la consegna.

Si vogliono tracciare i ritiri del pesce da parte del rider presso il pescatore e le consegne presso le abitazioni, riportando in particolare gli orari di ritiro e consegna.

2.2.7. Frasi relative al bundle

Il bundle è inserito dal pescatore.

Il bundle deve contenere informazioni quali descrizione e il peso.

2.3. Identificazione delle operazioni principali

Operazione 1: Ricerca pescatori che vendono pesce a una temperatura al momento del ritiro in media maggiore di 5°C, quindi la cui qualità è compromessa (1/mese)
Operazione 2: Selezione di tutti i clienti che vivono a "Praiano1" ma ordinano per una zona differente (1/anno)
Operazione 3: Calcolo spesa mensile di un cliente (900/mese)
Operazione 4: Creazione di un'ordinazione prenotata (200/giorno)
Operazione 5: Calcolo quantità di bundle consegnati da un rider (40/mese)
Operazione 6: Ricerca dei rider, e relative zone di lavoro, che sono gli unici rider affidati alla zona e consegnano un numero di ordini superiore alla media (1/semestre)

3. Progettazione Concettuale

Workpackage	***Task***	***Responsabile***
WP1	Progettazione Concettuale	Cirillo Francesco Pio

3.1. Schema Concettuale

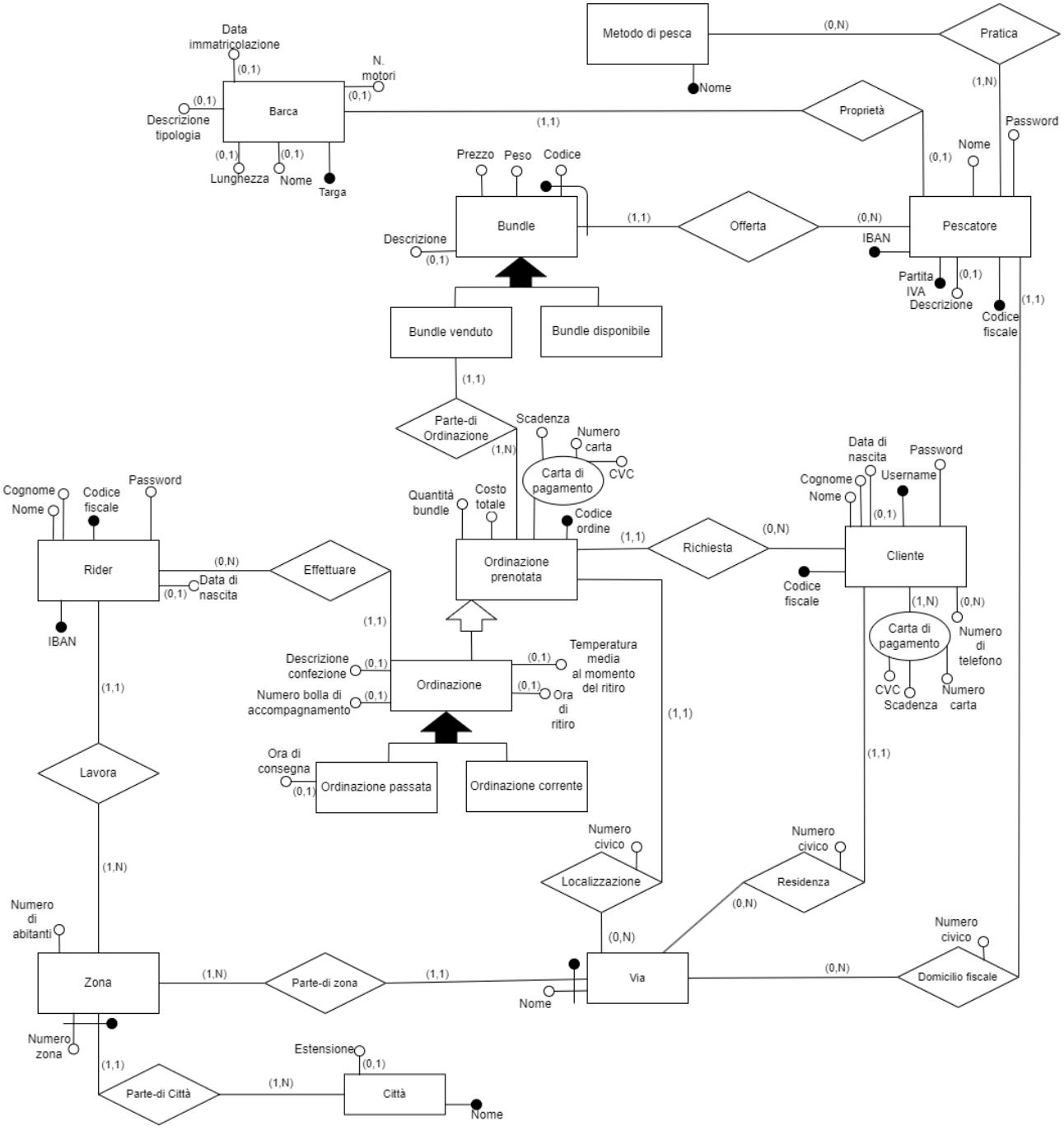

Figura 1. Schema E-R

3.1.1. Note sullo schema E-R

Per creare lo schema ER si è scelto di seguire una strategia mista in quanto pur decidendo di suddividere i requisiti in componenti separate non si è rinunciato alla definizione di uno schema scheletro al fine di mantenere una visione d'insieme.

Per semplicità si modellano tutte le diverse tipologie di strade con l'entità "Via" ignorando le differenze tra piazze, strade, vie, traverse ecc.

Si evidenzia che la scelta di identificare l'entità Città per mezzo del suo nome è giustificata e non rappresenta una criticità in quanto il servizio che si sta modellando copre un'area geografica contenuta all'interno di una provincia e, almeno in Italia attualmente, non esistono comuni omonimi nella stessa provincia né tantomeno nella stessa regione.

Si nota la scelta di non considerare il Numero di Bolla di Accompagnamento come identificatore dell'Ordinazione, questo in quanto è teoricamente possibile che più Ordinazioni facciano parte della stessa spedizione.

Per non appesantire visivamente il contenuto illustrativo della documentazione, nelle figure che mostrano frammenti del diagramma E-R (da *figura2* a *figura19* e *figura22*) non sono riportate le molteplicità degli attributi opzionali.

Si nota l'uso di due diversi tipi di pattern concettuali "Parte-di". Tutti i casi di utilizzo di questo pattern rientrano nella casistica nella quale l'entità componente (es: "Via" tra "Via" e "Zona") non ha esistenza autonoma rispetto all'entità composta. Fa eccezione l'associazione tra le entità Bundle venduto e Ordinazione Prenotata, infatti l'esistenza di un Bundle non è subordinata alla sua appartenenza ad una Ordinazione. Il Bundle è perfettamente identificato dal Codice è dalla associazione che lo lega all'entità Pescatore.

Si valuta infine lo schema concettuale ottenuto sulla base dei 4 parametri di qualità:

- Correttezza, si è prestato attenzione all'uso non improprio dei costrutti del modello ER;
- Completezza, tutti i dati di interesse sono modellati nello schema presentato e tutte le operazioni di interesse sono effettuabili utilizzando concetti descritti nello schema;
- Leggibilità, lo schema risulta comprensibile, si è prestato attenzione nella scelta dei nomi affinché risultassero significativi e sono stati analizzati tutti i concetti in modo da chiarificarli nel dettaglio;
- Minimalità, in merito alla minimalità si riconosce la presenza di ridondanze che quindi rendono lo schema non minimale, tuttavia non tutte le ridondanze sono necessariamente un male, nello specifico le ridondanze presenti non sono frutto di un errore ma bensì di accorte scelte progettuali; nella fase apposita si provvederà a decidere se mantenere o eliminare le ridondanze in questione.

3.2. Design Pattern

3.2.1. Pattern Evoluzione di un concetto

L'Ordinazione Prenotata quando viene ritirata dal rider subisce una evoluzione nel tempo in Ordinazione a causa dell'aggiunta di nuove informazioni quali ad esempio l'attributo Ora di ritiro e l'attributo Temperatura al momento del ritiro. Si modella in questo modo in maniera più accurata il momento in cui l'associazione Effettuare entra in essere, vale a dire dopo il ritiro da parte del rider; per questo l'associazione passa dall'essere opzionale (cardinalità minimo 0) a obbligatoria.

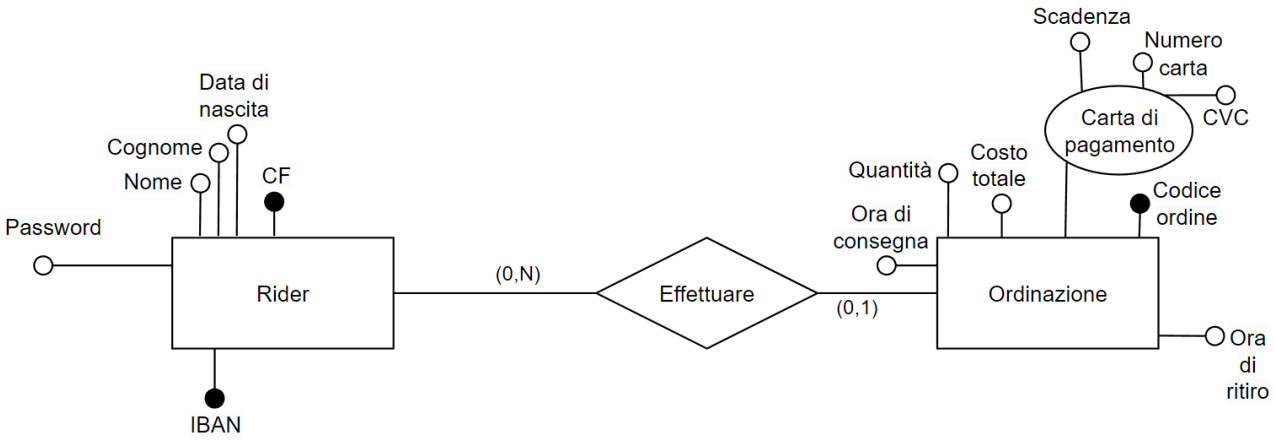

Figura 2. Schema precedente all'applicazione del Pattern EVOLUZIONE DI UN CONCETTO.

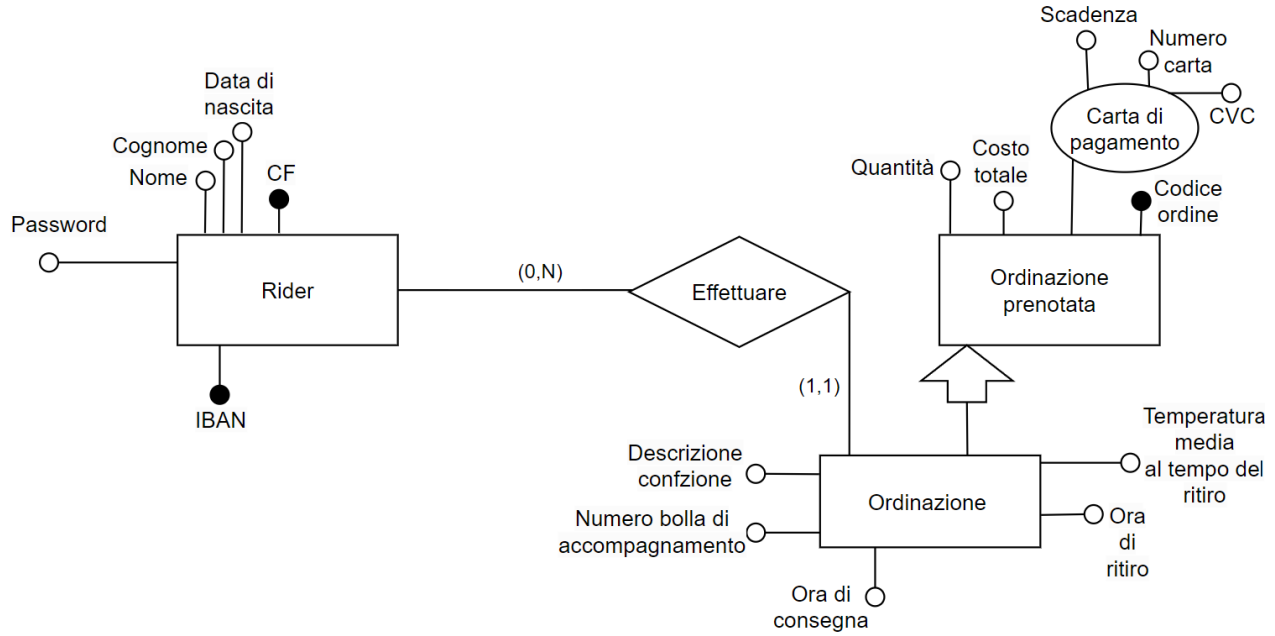

Figura 3 Schema successivo all'applicazione del Pattern EVOLUZIONE DI UN CONCETTO.

3.2.2. Pattern Storicizzazione di entità

Al fine di tracciare gli ordini passati è stato applicato il pattern di storicizzazione per distinguere tra Ordinazione Corrente e Ordinazione Passata, quest'ultima entità è caratterizzata dall'attributo Ora di consegna, un'informazione di interesse nella realtà modellata.

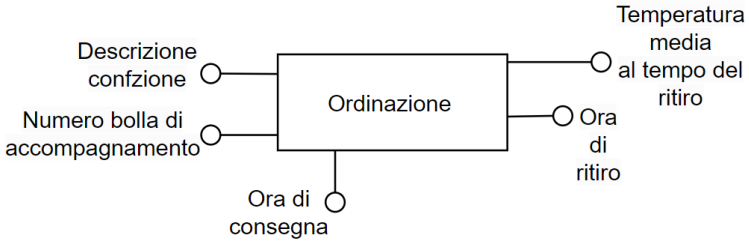

Figura 4. Schema precedente all'applicazione del Pattern STORICIZZAZIONE DI ENTITA'.

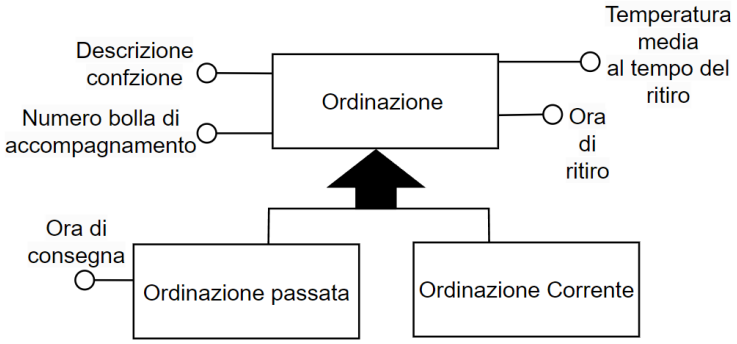

Figura 5. Schema successivo all'applicazione del Pattern STORICIZZAZIONE DI ENTITA'.

Si considera di storicizzare le ordinazioni relative all'ultimo anno.

3.3. Dizionario dei Dati

Entità	Descrizione	Attributi	Identificatore
Barca	Imbarcazione con cui il pescatore può svolgere uno o più metodi di pesca.	Data immatricolazione, N. motori, Descrizione tipologia, Lunghezza, Nome, Targa	Targa
Bundle	Pacchetto di pesce composto dal pescatore.	Peso, Prezzo, Codice, Descrizione	Codice & Pescatore
Pescatore	Individuo che pratica determinate tecniche di pesca. Offre ai clienti il suo pescato in bundle.	Partita IVA, Codice Fiscale, Nome, Descrizione, Password, IBAN	Partita IVA, Codice Fiscale, IBAN
Ordinazione	Acquisto di determinati bundle da parte di un cliente, deve essere consegnato al cliente da un rider.	Quantità bundle, Costo totale, Carta di pagamento (Numero carta, Scadenza, CVC), Codice ordine, Ora di ritiro, Ora di consegna, Temperatura media al momento del ritiro, Descrizione Confezione, Numero bolla di accompagnamento	Codice ordine
Rider	Individuo che preleva i bundle dai pescatori per consegnarli ai clienti.	Nome, Cognome, Data di nascita, Codice Fiscale, Password, IBAN	Codice Fiscale, IBAN
Cliente	Individuo che effettua delle ordinazioni.	Nome, Cognome, Data di nascita, Codice Fiscale, Username, Password, Carta di pagamento (Numero carta, Scadenza, CVC), Numero di telefono	Codice Fiscale, Username
Zona	Porzione di città che comprende un certo numero di vie.	Numero di abitanti, Numero Zona	Numero Zona & Città
Via	Via in una città che fa parte di una zona.	Nome	Nome & Zona

Tabella 2. Dizionario dei dati – Entità

Relazioni	Descrizione	Entità Coinvolte	Attributi
Offerta	Associa un Pescatore ai Bundle che mette in vendita.	Pescatore (0,N), Bundle (1,1)	
Parte-di Ordinazione	Associa un Bundle venduto all'Ordinazione di cui fa parte.	Bundle venduto (1,1), Ordinazione Prenotata (1,N)	
Effettuare	Associa il Rider alle Ordinazioni che deve consegnare.	Rider (0,N), Ordinazione (1,1)	
Richiesta	Associa un Cliente alle proprie Ordinazioni Prenotate.	Cliente (0,N), Ordinazione Prenotata (1,1)	
Lavora	Associa una Zona ai Rider che ci lavorano.	Zona (1,N), Rider (1,1)	

Parte-di zona	Associa una Zona alle Vie che la compongono.	Zona (1,N), Via (1,1)	
Localizzata	Associa una Ordinazione alla Via nella quale deve essere consegnata.	Ordinazione Prenotata (1,1), Via (0,N)	Numero civico
Proprietà	Associa una Barca al pescatore che la possiede.	Pescatore (0,1), Barca(1,1)	

Tabella 3. Dizionario dei dati - Relazioni

Si vuole specificare che per indicare due identificatori diversi per la stessa entità si sono elencati gli identificatori separandoli con la virgola; per specificare invece identificatori composti sono stati elencati i vari attributi che li compongono separati da &.

Workpackage	***Task***	***Responsabile***
WP4	Regole Aziendali	Fasolino Alessandra

3.4. Regole Aziendali

Regole di Vincolo
(RV1) Una Zona ha una dimensione massima di circa 2500 abitanti e comunque mai superiore a 3000 abitanti. **(RV2)** In un'ordinazione l'ora di consegna non può essere precedente allora di ritiro. **(RV3)** Il rider può consegnare solo ordinazioni che sono relative alla zona nella quale lavora. **(RV4)** I prezzi di Bundle e Ordinazioni devono essere tutti positivi. **(RV5)** La carta di pagamento relativa ad un ordine deve essere posseduta dal cliente che ha fatto l'ordine.

Tabella 4. Regole di vincolo

Regole di derivazione
(RD1) Il costo totale di un'Ordinazione prenotata si ottiene sommano il Prezzo dei Bundle che la compongono. **(RD2)** La quantità di bundle presenti in un'ordinazione si ottiene contando i bundle venduti che sono legati all'ordinazione in questione dall'associazione Parte-di Ordinazione

Tabella 5. Regole di derivazione

4. Progettazione Logica

Workpackage	*Task*	*Responsabile*
WP2	Progettazione Logica	Fasolino Alessandra

4.1. Ristrutturazione Schema Concettuale

4.1.1. Analisi delle Prestazioni

4.1.1.1. Tavola dei volumi

Concetto	Tipo	Volume
Pescatore	E	50
Bundle disponibile	E	150
Bundle venduto	E	90.075
Metodo di pesca	E	5
Barca	E	100
Cliente	E	900
Ordinazione prenotata	E	25
Ordinazione passata	E	60.000
Ordinazione corrente	E	25
Rider	E	40
Zona	E	20
Città	E	15
Via	E	450
Pratica	R	150
Proprietà	R	100
Offerta	R	90.225
Parte-di Ordinazione	R	90.075
Effettuare	R	60.025
Richiesta	R	60.050
Lavora	R	40
Localizzazione	R	60.050
Residenza	R	900
Domicilio fiscale	R	50
Parte-di zona	R	450
Parte-di Città	R	20

Tabella 6. Tavola dei volumi

Note sui volumi

Per stimare accuratamente il quantitativo di **clienti** in media registrati al servizio si è seguito il seguente ragionamento:

- Abitanti totali della zona coperta (Costiera Amalfitana): circa 37.500
- Numero nuclei familiari stimato: circa 9374

Considerando un unico iscritto per nucleo familiare e considerando un numero di iscritti al servizio pari al 10 per cento dei potenziali acquirenti si ottiene la stima effettuata.

4.1.1.2. Tavola delle operazioni

Operazione	Tipo	Frequenza
Operazione 1: Ricerca pescatori che vendono pesce mal conservato	B	1/mese
Operazione 2: Ricerca clienti che ordinano per una zona diversa di quella di residenza	B	1/anno
Operazione 3: Calcolo spesa mensile di un cliente	B	900/mese
Operazione 4: Creazione di un'ordinazione prenotata	I	200/giorno
Operazione 5: Calcolo quantità di Bundle consegnati da un Rider	B	40/mese
Operazione 6: Rider che lavorano da soli nella zona e consegnano più ordini della media	B	1/semestre

Tabella 7. Tavola delle operazioni

4.2. Analisi delle ridondanze

- Ridondanza 1: Costo totale (ORDINAZIONE PRENOTATA). Il costo totale dell'Ordinazione si ottiene sommando il valore dell'attributo Prezzo delle entità Bundle venduto legate all'Ordinazione in questione per mezzo dell'associazione Parte-di ordinazione.
  TIPO: Attributo derivabile da altre entità.
- Ridondanza 2: Quantità bundle (ORDINAZIONE PRENOTATA). La quantità di bundle dell'Ordinazione si ottiene contando il numero di occorrenze dell'associazione Parte-di bundle.
  TIPO: Attributo Derivabile da conteggio di occorrenze.

4.2.1. Analisi della ridondanza 1: Costo di un'ordinazione

- **Operazione 3: calcolo spesa mensile di un cliente**

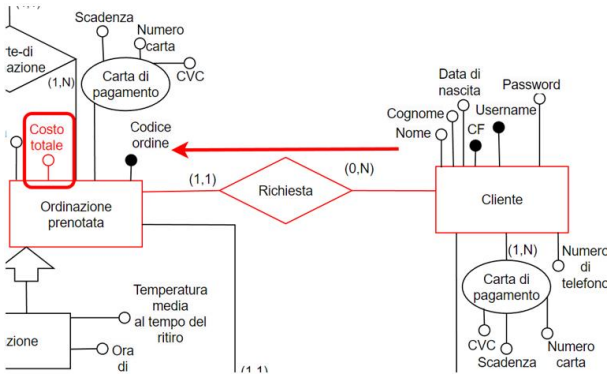

Figura 6. Percorso operazione 3 con ridondanza 1

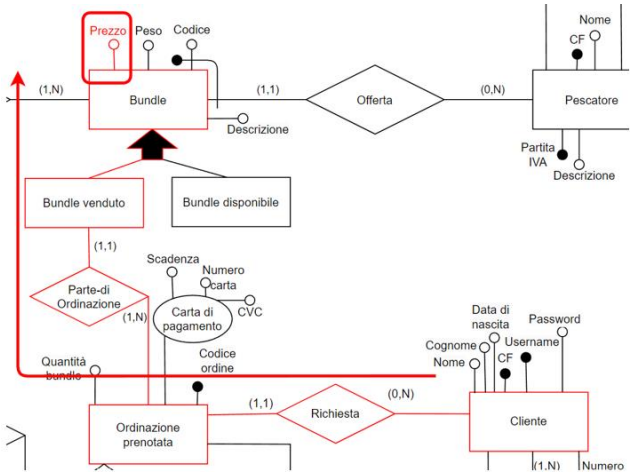

Figura 7. percorso operazione 3 senza ridondanza 1

Con Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
CLIENTE	E	1	L
RICHIESTA	R	7	L

ORDINAZIONE PASSATA	E	7	L

Tabella 8. Tavola degli accessi per l'operazione 3 in presenza di ridondanza 1

Senza Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
CLIENTE	E	1	L
RICHIESTA	R	7	L
ORDINAZIONE PASSATA	E	7	L
PARTE-DI ORDINAZIONE	R	10	L
BUNDLE VENDUTO	E	10	L

Tabella 9. Tavola degli accessi per l'operazione 3 in assenza di ridondanza 1

- **Operazione 4: creazione di un'ordinazione prenotata**

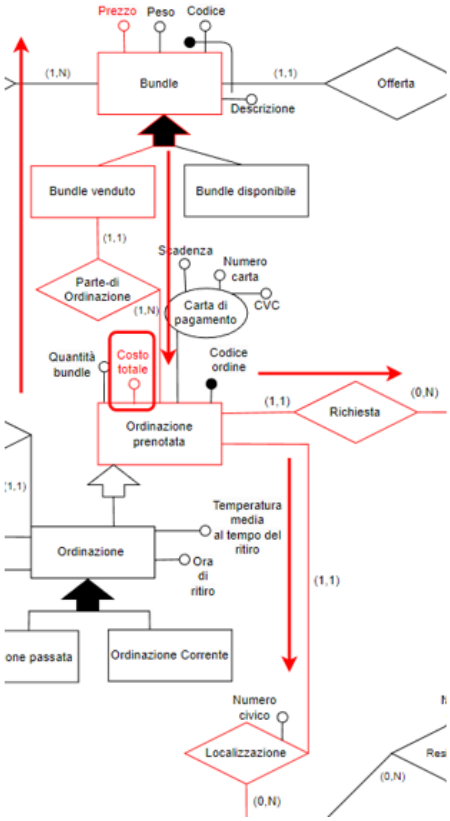

Figura 8. Percorso operazione 4 con ridondanza 1

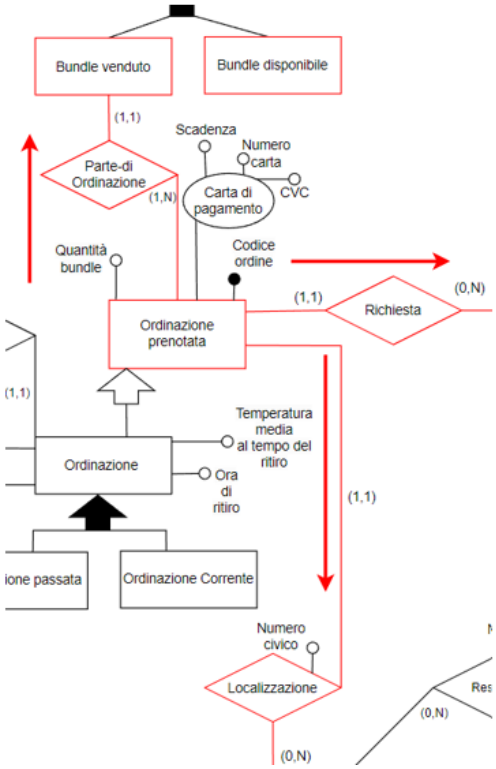

Figura 9. percorso operazione 4 senza ridondanza 1

Con Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
ORDINAZIONE PRENOTATA	E	1	S
LOCALIZZAZIONE	R	1	S
RICHIESTA	R	1	S
PARTE-DI-ORDINAZIONE	R	2	S
BUNDLE DISPONIBILE	E	2	L
BUNDLE DISPONIBILE	E	2	S
BUNDLE VENDUTO	E	2	L
ORDINAZIONE PRENOTATA	E	1	L
ORDINAZIONE PRENOTATA	E	1	S

Tabella 10. Tavola degli accessi per l'operazione 4 in presenza di ridondanza 1

Senza Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
ORDINAZIONE PRENOTATA	E	1	S
LOCALIZZAZIONE	R	1	S
RICHIESTA	R	1	S
PARTE-DI-ORDINAZIONE	R	2	S
BUNDLE DISPONIBILE	E	2	L
BUNDLE DISPONIBILE	E	2	S

Tabella 11. Tavola degli accessi per l'operazione 4 in assenza di ridondanza 1

Nota: Per entrambi i casi (operazione 4 con e senza ridondanza) la Lettura e Scrittura di Bundle disponibile viene considerata necessaria per la conversione a Bundle venduto

4.2.1.1. Valutazione della ridondanza 1

4.2.1.1.1. Calcolo occupazione memoria

Assumendo che il costo totale dell'ordinazione prenotata richieda 5 byte, sufficiente per memorizzare un tipo numeric(5,2), risulta che il dato ridondante comporta

$$5\ byte \times 60050\ ordinazioni = 300250\ byte - circa\ 293{,}21\ kilobyte$$

4.2.1.1.2. Costo della singola operazione

Con Ridondanza

OPERAZIONE	L	S
Operazione 3	15	0
Operazione 4	5	8

Tabella 12. Tavola resoconto degli accessi in presenza di ridondanza 1

Senza Ridondanza

OPERAZIONE	L	S
Operazione 3	35	0
Operazione 4	2	7

Tabella 13. Tavola resoconto degli accessi in assenza di ridondanza 1

4.2.1.1.3. Costo delle operazioni considerate le loro frequenze

Con Ridondanza

OPERAZIONE	L	S
Operazione 3	13500	0
Operazione 4	25000	40000
TOTALE	**38500**	**40000**

Tabella 14. Tavola resoconto degli accessi in funzione delle frequenze in presenza di ridondanza 1

Senza Ridondanza

OPERAZIONE	L	S
Operazione 3	31500	0
Operazione 4	10000	35000
TOTALE	**41500**	**35000**

Tabella 15. Tavola resoconto degli accessi in funzione delle frequenze in assenza di ridondanza 1

Viene considerato il costo di un accesso in scrittura doppio rispetto ad un accesso in lettura. Le quantità di letture e scritture riportate nella tabella sono relative alle frequenze mensili delle operazioni.

4.2.1.1.4. Conclusioni valutazione

Dopo aver analizzato le operazioni che coinvolgono la ridondanza si osserva che, con il carico considerato:

- In presenza di ridondanza il costo delle operazioni è di circa 118500 accessi mensili
- L'occupazione di memoria è di circa 293,21 kilobyte (dimensione poco rilevante per i moderni calcolatori)
- In assenza di ridondanza il costo delle operazioni è di 111500 accessi giornalieri

Pertanto, si decide di non mantenere la ridondanza in quanto aumenta il numero di accessi .

4.2.2. Analisi della ridondanza 2: Quantità di bundle in un'ordinazione

- **Operazione 4: creazione di un'ordinazione prenotata**

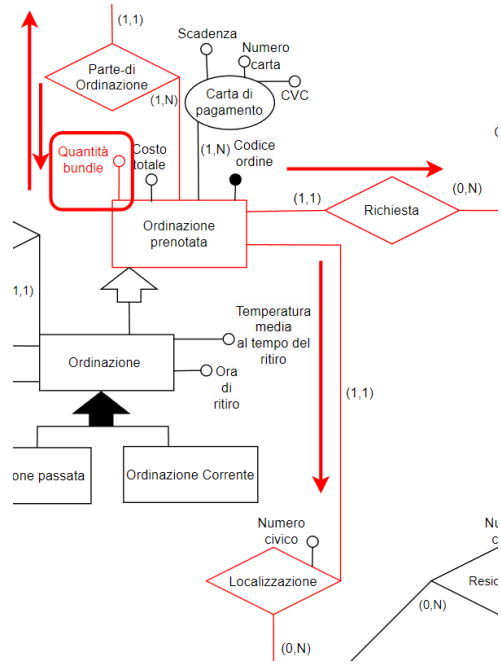

Figura 10. Percorso operazione 4 con ridondanza 2

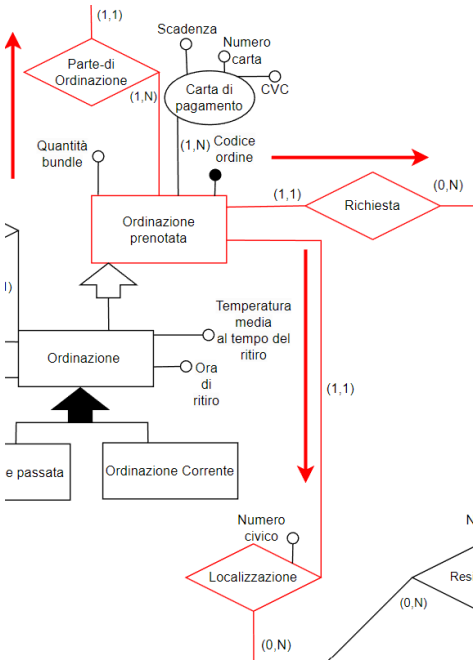

Figura 11. percorso operazione 4 senza ridondanza 2

Con Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
ORDINAZIONE PRENOTATA	E	1	S
RICHIESTA	R	1	S
LOCALIZZAZIONE	R	1	S
PARTE-DI ORDINAZIONE	R	2	S
BUNDLE DISPONIBILE	E	2	L
BUNDLE DISPONIBILE	E	2	S
PARTE-DI ORDINAZIONE	R	2	L
ORDINAZIONE PRENOTATA	E	1	L

ORDINAZIONE PRENOTATA	E	1	S

Tabella 16. Tavola degli accessi per l'operazione 4 in presenza di ridondanza 2

Senza Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
ORDINAZIONE PRENOTATA	E	1	S
RICHIESTA	R	1	S
LOCALIZZAZIONE	R	1	S
PARTE-DI ORDINAZIONE	R	2	S
BUNDLE DISPONIBILE	E	2	L
BUNDLE DISPONIBILE	E	2	S

Tabella 17. Tavola degli accessi per l'operazione 4 in assenza di ridondanza 2

- **Operazione 5: calcolo quantità di bundle consegnati da un rider**

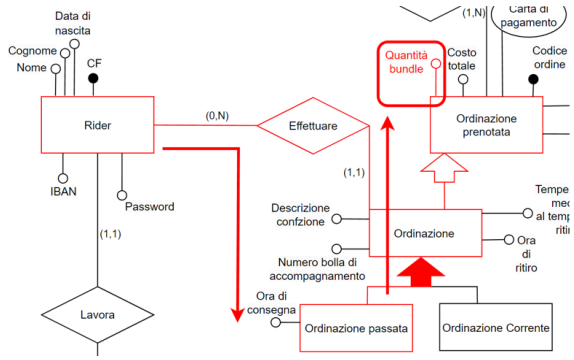

Figura 12. Percorso operazione 5 con ridondanza 2

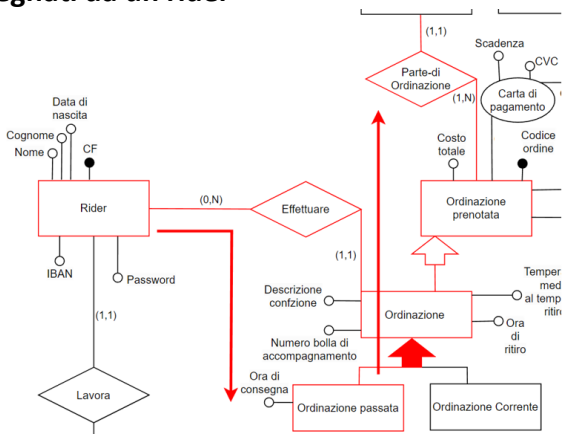

Figura 13. percorso operazione 5 senza ridondanza 2

Con Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
RIDER	E	1	L
EFFETTUARE	R	1500	L
ORDINAZIONE PASSATA	E	1500	L

Tabella 18. Tavola degli accessi per l'operazione 5 in presenza di ridondanza 2

Senza Ridondanza

CONCETTO	COSTRUTTO	ACCESSI	TIPO
RIDER	E	1	L
EFFETTUARE	R	1500	L
ORDINAZIONE PASSATA	E	1500	L
PARTE-DI ORDINAZIONE	R	2250	L

Tabella 19. Tavola degli accessi per l'operazione 5 in assenza di ridondanza 2

4.2.2.1. Valutazione della ridondanza 2

4.2.2.1.1. Calcolo occupazione memoria

Assumendo che la quantità di bundle in ordinazione richieda 4 byte, sufficiente per memorizzare numeri interi, risulta che il dato ridondante comporta

$$4\ byte \times 90225\ bundle = 360900\ byte - circa\ 352{,}44\ kilobyte$$

4.2.2.1.2. Costo della singola operazione

Con Ridondanza

OPERAZIONE	L	S
Operazione 4	5	8
Operazione 5	3001	0

Tabella 20. Tavola resoconto degli accessi in presenza di ridondanza 2

Senza Ridondanza

OPERAZIONE	L	S
Operazione 4	2	7
Operazione 5	5251	0

Tabella 21. Tavola resoconto degli accessi in assenza di ridondanza 2

4.2.2.1.3. Costo delle operazioni considerate le loro frequenze

Con Ridondanza

OPERAZIONE	L	S
Operazione 4	25000	40000
Operazione 5	120040	0
TOTALE	**145040**	**40000**

Tabella 22. Tavola resoconto degli accessi in funzione delle frequenze in presenza di ridondanza 2

Senza Ridondanza

OPERAZIONE	L	S
Operazione 4	10000	35000
Operazione 5	210040	0
TOTALE	**220040**	**35000**

Tabella 23. Tavola resoconto degli accessi in funzione delle frequenze in assenza di ridondanza 2

Viene considerato il costo di un accesso in scrittura doppio rispetto ad un accesso in lettura.

4.2.2.1.4. Conclusioni valutazione

Dopo aver analizzato le operazioni che coinvolgono la ridondanza si osserva che, con il carico considerato:

- In presenza di ridondanza il costo delle operazioni è di circa 225040 accessi mensili
- L'occupazione di memoria è di circa 352,44 kilobyte (dimensione poco rilevante per i moderni calcolatori)
- In assenza di ridondanza il costo delle operazioni è di 290040 accessi mensili

Pertanto, si decide di mantenere la ridondanza in quanto diminuisce il numero di accessi.

4.3. Eliminazione delle generalizzazioni

4.3.1. Generalizzazione Bundle

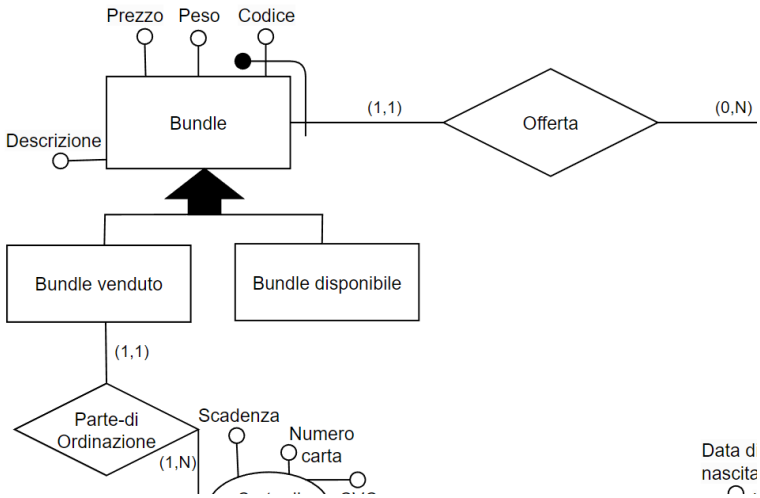

Figura 14. Generalizzazione Bundle prima dell'eliminazione

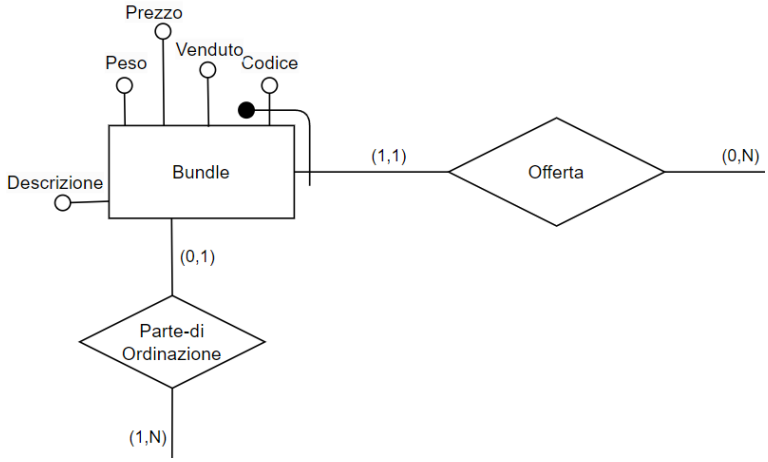

Figura 15. Entità Bundle dopo l'eliminazione

Si ritiene opportuno applicare la strategia di accorpamento delle figlie nel genitore (strategia 1) in quanto anche se alcune operazioni differenziano tra Bundle venduto e Bundle disponibile le due entità sono caratterizzate dagli stessi attributi quindi non ci sarà spreco di memoria, accorpandole, dovuto ad attributi nulli.

4.3.2. Generalizzazione Ordinazione

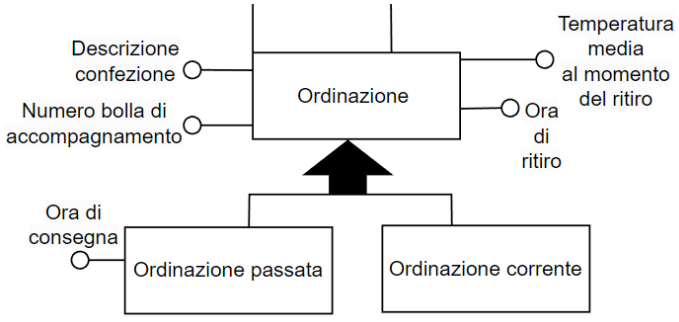

Figura 16. Generalizzazione Ordinazione prima dell'eliminazione

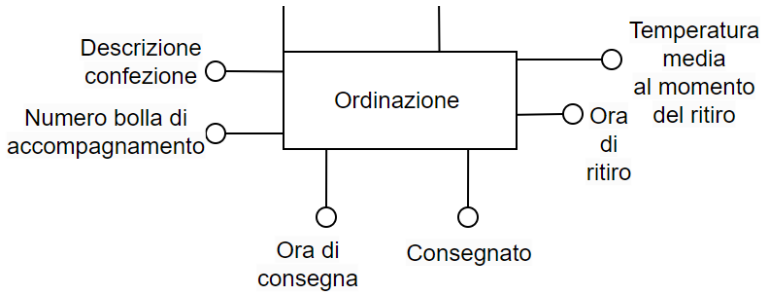

Figura 17. Entità Ordinazione dopo l'eliminazione

Tenuto in considerazione che la maggior parte delle operazioni relative alle entità in questione non differenziano fra le occorrenze delle due figlie della generalizzazione si ritiene opportuno adottare la strategia di accorpamento delle figlie nel genitore (strategia 1).

Si ritiene quindi necessario aggiungere un attributo (Consegnato) che permette di distinguere tra i due tipi di occorrenze. L'eventuale valore NULL dell'attributo Ora di consegna viene quindi contestualizzato dall'attributo Consegnato; quest'ultimo permette di sapere se la consegna non è stata ancora effettuata (e quindi non se ne conosce l'orario) o se la consegna è giunta al termine ma l'informazione Ora di consegna è assente.

4.3.3. Generalizzazione Ordinazione prenotata

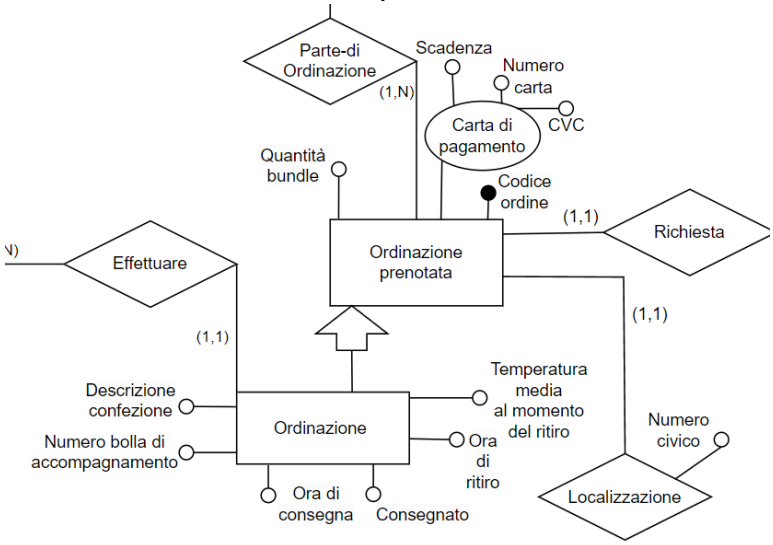

Figura 18. Generalizzazione Ordinazione prenotata prima dell'eliminazione

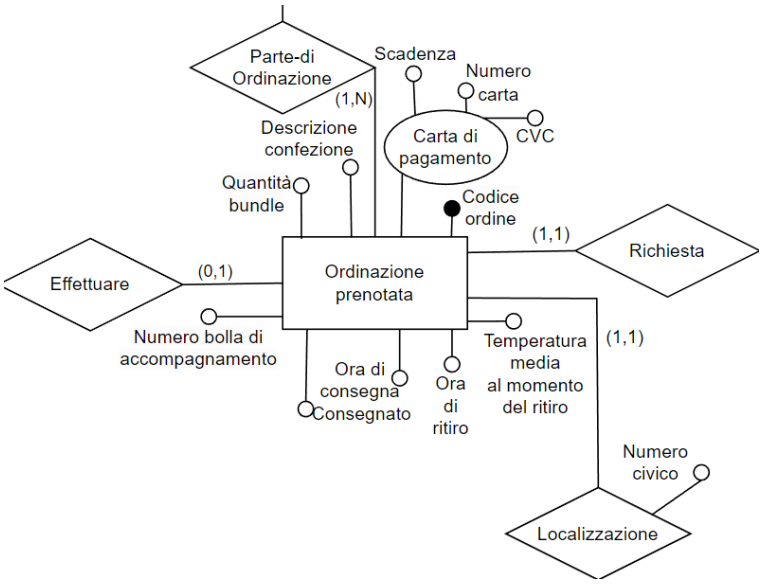

Figura 19. Entità Ordinazione dopo l'eliminazione

Si considera opportuno applicare la strategia di accorpamento delle figlie nel genitore della generalizzazione (strategia 1). Si nota infatti l'assenza di operazioni che differenziano occorrenze tra l'entità genitore e l'entità figlia. Anche l'operazione di conteggio delle ordinazioni effettuate da un rider effettivamente non necessita di accedere all'entità ordinazione in quanto è sufficiente contare le occorrenze dell'associazione effettuare legate al rider di interesse.

La strategia scelta è anche in linea con il concetto alla base di questa generalizzazione cioè l'evoluzione del concetto di ordinazione che quando viene prelevata dal rider acquisisce nuovi attributi e associazioni.

4.4. Partizionamento/Accorpamento Entità e Associazioni

4.4.1. Eliminazione attributo multivalore Numero di telefono in Cliente

La rimozione dell'attributo multivalore Numero di telefono dell'entità Cliente è stata eseguita tramite la reificazione dell'attributo Numero di telefono in entità. L'entità creata è stata legata a Cliente per mezzo di un'associazione le cui cardinalità rispecchiano la situazione iniziale.

4.4.2. Eliminazione attributo composto e multivalore Carta di credito di Cliente e di Ordinazione

La rimozione di un attributo multivalore richiede la reificazione dell'attributo ad entità, di conseguenza, essendo Carta di pagamento dell'entità Cliente un attributo multivalore composto, non si è potuto semplicemente aggiungere i suoi attributi componenti all'entità Cliente, come per un attributo composto tradizionale, ma lo si è invece dovuto reificare.

Considerata la presenza dell'Entità Carta di Pagamento invece di scegliere la via tradizionale

Anche per l'eliminazione dell'attributo composto Carta di Pagamento dell'entità Ordinazione si è deciso di non usare il metodo tradizionale di accorpamento degli attributi componenti nell'entità madre in quanto, vista la presenza dell'Entità Carta di Pagamento reificata dall'attributo omonimo

di Cliente, si è ritenuto più adeguato legare l'Entità Carta di Pagamento non solo a Cliente ma anche ad Ordinazione. Le due diverse associazioni presentano cardinalità differenti per rispecchiare le diverse situazioni iniziali.

4.5. Scelta degli identificatori principali

4.5.1. Introduzione nuovo identificatore in Cliente

L'entità Cliente presenta due diversi identificatori candidati: Codice Fiscale e Username.

La scelta per l'identificatore principale ricade su Username poiché i requisiti richiedono che il Cliente possa accedere alla piattaforma tramite username e password. La presenza di un'operazione che distingue le occorrenze di cliente sulla base dell'attributo Username lo rende più che idoneo a ricoprire il ruolo di identificatore principale.

4.5.2. Scelta identificatore principale in Rider

L'entità Rider presenta due diversi identificatori candidati: Codice Fiscale, IBAN.

Nonostante la presenza degli identificatori candidati Codice Fiscale e IBAN, si preferisce introdurre un altro attributo, Codice Rider, in quanto lo si ritiene più calzante all'identificazione di un utente di tipo Rider per un servizio come quello modellato.

4.5.3. Scelta identificatore principale in Pescatore

L'entità Pescatore presenta tre diversi identificatori candidati: Codice Fiscale e Partita IVA, IBAN.

I tre sono egualmente preferibili dal punto di vista dei requisiti per l'identificatore principale (essere non nulli, essere costituiti da un singolo attributo, essere identificatori interni). Si preferisce l'attributo Partita IVA in quanto lo si ritiene più calzante all'identificazione di un'attività commerciale.

4.6. Schema ristrutturato finale

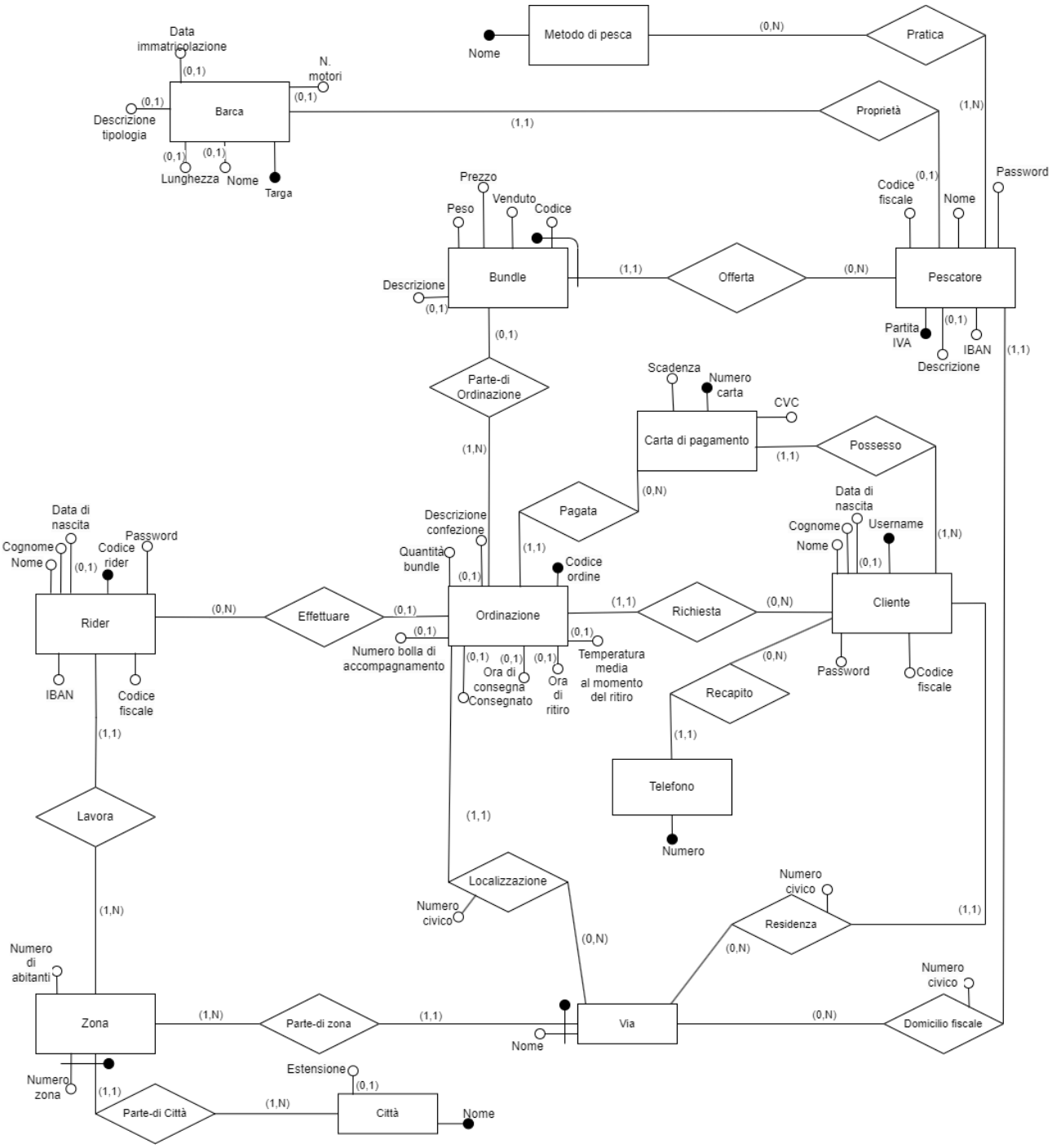

Figura 20. Schema ER Ristrutturato

4.7. Schema logico

Nello schema sono rispettate le seguenti convenzioni:

- Sottolineato a linea continua usato per attributi che fanno parte dell'identificatore primario;
- Sottolineato a linea tratteggiata usato per attributi che rappresentano vincoli d'integrità referenziale;
- Sottolineato con doppia linea continua per attributi che svolgono entrambe le funzioni di chiave primaria e vincolo di integrità referenziale
- Con il simbolo * (asterisco) usato per indicare gli attributi che possono assumere valore nullo .

Pescatore(PartitaIVA, Via, Zona, Citta, NumeroCivico, CodiceFiscale, IBAN, Nome, Password, Descrizione*),
MetodoDiPesca(Nome)
Pratica(MetodoDiPesca, Pescatore)
Barca(Targa, Proprietario, DataImmatricolazione*, NMotori*, DescrizioneTipologia*, Lunghezza*, Nome*)
Bundle(Codice, Pescatore, Ordinazione*, Peso, Prezzo, Venduto, Descrizione*)
Ordinazione(CodiceOrdine, DescrizioneConfezione*, Consegnato*, NumeroBollaDiAccompagnamento*, TemperaturaMediaAlMomentoDelRitiro*, OraRitiro*, OraConsegna*, Cliente, Via, Zona, Citta, NumeroCivico, CartaPagamento, Rider*, QuantitaBundle)
Cliente(Username, DataNascita*, Cognome, Nome, Password, CodiceFiscale, Via, Zona, Citta, NumeroCivico)
CartaPagamento(NumeroCarta, CVC, Scadenza, Cliente)
Telefono(Numero, Cliente)
Rider(CodiceRider, Zona, Citta, Password, IBAN, Cognome, Nome, DataNascita*, CodiceFiscale)
Citta(Nome, Estensione)
Zona(NumeroZona, Citta, NumeroAbitanti)
Via(Nome, Zona, Citta)

4.8. Documentazione dello schema logico

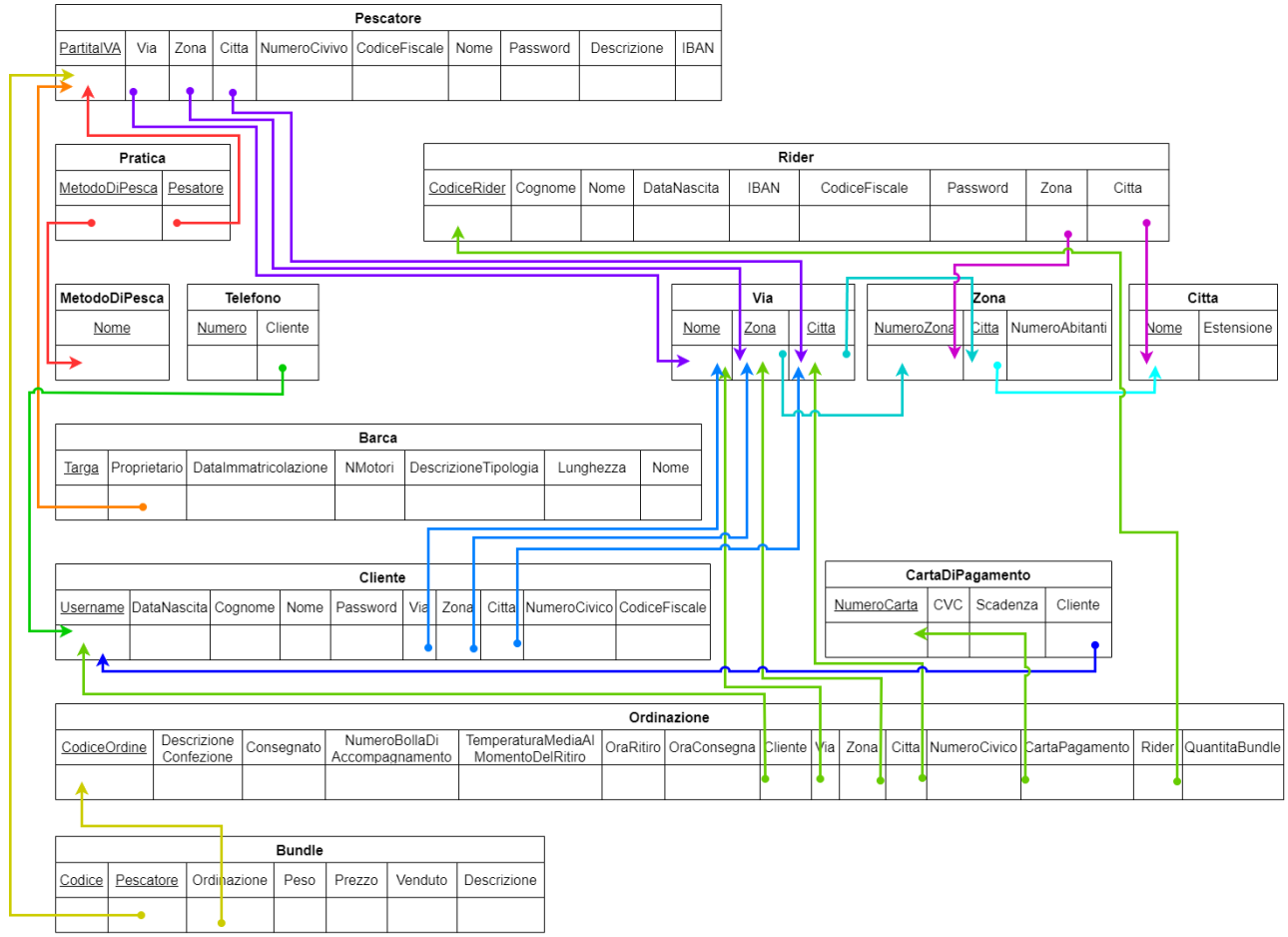

Figura 21. Schema logico

5. Normalizzazione

Workpackage	*Task*	*Responsabile*
WP3	Normalizzazione	Cirillo Francesco Pio

5.1. Considerazioni sullo schema concettuale

Prima di approcciare le verifiche delle Forme Normali basandosi sullo schema logico sono presentate alcune considerazioni sullo schema concettuale al fine di verificare la qualità dello schema. Si precisa che si parla in questo momento dello schema ER post ristrutturazione, non avrebbe infatti senso parlare di forme normali per l'ER originale che, presentando attributi composti e multivalore, sicuramente non rispetta la prima forma normale. L'ER ristrutturato è quindi sicuramente in 1NF.

Partendo dal presupposto che preliminarmente è possibile considerare ciascuna entità e ciascuna associazione come una relazione presentiamo le seguenti considerazioni.

5.1.1. Verifiche di normalizzazione su entità

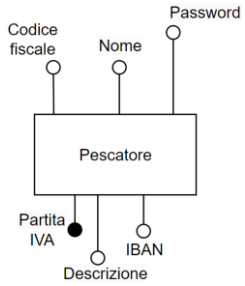

Figura 22. Entità Pescatore

Analizziamo l'entità Pescatore, consideriamo nello specifico i suoi attributi.

Risulta banale osservare che l'entità è in Seconda Forma Normale in quanto il suo identificatore è composto da un unico attributo.

Notiamo che tutti gli attributi dipendono funzionalmente dall'identificatore dell'entità: Partita IVA. Ci sono inoltre altre dipendenze funzionali: tutti gli attributi dipendono da IBAN (chiave candidata) e tutti gli attributi dipendono da Codice Fiscale (chiave candidata).

Dalle dipendenze individuate si può notare sia che *non ci sono attributi non chiave che dipendono transitivamente dalla chiave* (3NF) sia che *ogni determinante delle dipendenze funzionali presenti è una chiave candidata* (BCNF).

Per mezzo di ragionamenti similari è possibile analizzare tutte le entità dell'ER e dimostrare che l'ER nel complesso è in BCNF.

5.1.2. Verifiche di normalizzazione su associazioni

Si evidenzia l'assenza di associazioni non binarie. Ogni associazione binaria è, sicuramente, in BCNF, di conseguenza tutte le associazioni dell'ER sono in BCNF.

5.2. Prima Forma Normale

Il Database risulta rispettare i criteri della Prima Forma Normale, in quanto costituito da Relazioni che rispettano i criteri della Prima Forma Normale poiché *prive di attributi composti e attributi multivalore*.

Durante la Ristrutturazione dello Schema Concettuale sono infatti stati rimossi attributi composti e attributi multivalore il che ha portato ad uno Schema Logico privo degli stessi.

5.3. Seconda Forma Normale

Di seguito si verifica la Seconda Forma Normale del Database analizzando le singole relazioni che lo compongono.

La Seconda Forma Normale è verificata per una relazione se questa è in Prima Forma Normale e se ogni attributo non primo (attributi che non appartengono alla chiave primaria) ha una dipendenza funzionale piena dalla chiave della relazione, cioè non ci sono attributi non chiave che dipendono parzialmente dalla chiave. Ne consegue che le relazioni in Prima Forma Normale che hanno la chiave primaria composta da un solo attributo sono in Seconda Forma Normale. Questo è il caso delle seguenti relazioni:

- Pescatore;
- Rider;
- Telefono;
- Citta;
- Barca;
- Cliente;
- CartaDiPagamento;
- Ordinazione.

Dalla definizione di Seconda Forma Normale si deduce anche che le relazioni, in Prima Forma Normale, prive di attributi non primi soddisfano i criteri per la Seconda Forma Normale. Questo è il caso delle seguenti relazioni:

- Pratica;
- Via.

Resta quindi da verificare se le relazioni Zona e Bundle sono in Seconda Forma Normale.

Zona(NumeroZona, Citta, NumeroAbitanti)

NumeroZona Citta → NumeroAbitanti

L'unico attributo non primo della relazione Zona è NumeroAbitanti, questo è legato da una dipendenza funzionale piena alla chiave primaria della relazione. Questo perché rimuovendo un qualsiasi attributo della chiave primaria dalla dipendenza questa perde di validità. Quindi si deduce che Zona è una relazione in Seconda Forma Normale.

Bundle(Codice, Pescatore, Ordinazione, Peso, Prezzo, Venduto)

Codice Pescatore → Ordinazione
Codice Pescatore → Peso
Codice Pescatore → Prezzo
Codice Pescatore → Venduto

Si nota che non sono presenti altre dipendenze funzionali oltre a quelle banali infatti gli attributi non primi della relazione Bundle sono legati da dipendenze funzionali piene alla chiave primaria della relazione. Quindi si deduce che Bundle è una relazione in Seconda Forma Normale.

Tutte le relazioni sono in Seconda Forma Normale quindi il Database è in Seconda Forma Normale.

5.4. Terza Forma Normale

Di seguito si verifica la Terza Forma Normale del Database analizzando le singole relazioni che lo compongono.

La Terza Forma Normale è verificata per una relazione se questa è in Seconda Forma Normale e se per ogni dipendenza funzionale non banale X → A definita su di essa è verificata almeno una delle seguenti condizioni:

- *X contiene una chiave candidata K della relazione;*
- *A appartiene ad almeno una chiave candidata della relazione.*

Cliente(Username, DataNascita, Cognome, Nome, Password, CodiceFiscale, Via, Zona, Citta, NumeroCivico)

Si evidenzia che tutti gli attributi dipendono funzionalmente dall'attributo Username che è infatti chiave primaria della relazione. Si evidenzia anche che tutti gli attributi dipendono funzionalmente dall'attributo CodiceFiscale che è infatti chiave candidata della relazione.

Si può osservare che la definizione di Terza Forma Normale è valida per la relazione Cliente infatti tutti i determinanti delle dipendenze funzionali individuate sono chiavi della relazione. Risulta anche vero che *non sono presenti dipendenze transitive della forma K → A, dove K è la chiave ed esiste un altro insieme di attributi X, non chiave, con le dipendenze K → X e X → A.*

Con un procedimento similare è stata verificata l'aderenza ai criteri della Terza Forma Normale per tutte le altre relazioni del Database, tuttavia si decide di non allegare i procedimenti svolti al fine di non appesantire la documentazione vista la similarità a quanto già proposta con la relazione Cliente.

5.5. Forma Normale di Boyce e Code

Di seguito si verifica la Forma Normale di Boyce e Code del Database analizzando le singole relazioni che lo compongono.

La Forma Normale di Boyce e Code è verificata per una relazione se questa è in Prima Forma Normale e se per ogni dipendenza funzionale non banale X → A definita su di essa, X contiene una chiave candidata K della relazione, cioè X è superchiave della relazione.

Si può dimostrare che se una relazione ha solo una chiave allora la Terza Forma Normale e la Forma Normale di Boyce e Code coincidono. Tutte le relazioni della base dati eccetto Pescatore, Cliente e Rider hanno un'unica chiave candidata, per cui possiamo concludere che, avendo già dimostrato che sono in Terza Forma Normale, queste sono in Forma Normale di Boyce e Code.

In riferimento alla relazione Cliente si nota che, osservando le dipendenze descritte nel paragrafo precedente, tutti i determinanti sono chiavi candidate della relazione; ne consegue che Cliente è in Forma Normale di Boyce e Code.

Ragionamenti analoghi a quelli svolti per Cliente permettono di dimostrare che anche Rider e Pescatore sono in Forma Normale di Boyce e Code.

Si conclude che la base dati è in Forma Normale di Boyce e Code in quanto tutte le relazioni componenti sono in Forma Normale di Boyce e Code.

6. Script Creazione e Popolamento Database

Workpackage	***Task***	***Responsabile***
WP2	SQL: Script creazione e popolamento	Fasolino Alessandra

Nel Query Tool del database postgres:

```
1  CREATE DATABASE fish2home;
```

Tra la creazione delle relazioni e il popolamento delle stesse è presente la creazione dei trigger che è stata però qui omessa in quanto presentata nella sezione apposita.
Si pone l'accento su alcuni punti rilevanti come:

- riga16 - Creazione di un dominio per facilitare l'inserimento dell' attributo Password;
- riga 18/37 - Tre metodologie differenti per la creazione di una chiave primaria;
- riga 58/65 - Due diversi modi per stabilire un vincolo di integrità referenziale, inserendo anche una politica di reazione al secondo;
- riga 73 - Uso di valori di default;
- riga118/148 - Utilizzo di costrutti per specificare vincoli .

Nel Query Tool del database fish2home:

```
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
```

```
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');
```

```
/*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, 'CNTMTE88A12F205W',
            'Pescheria marini', 'matteoContiPescheria', 'IT12X34567890123456789024');
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', 'IT98Y34567890123456789023');
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');

```

```
/*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. Query SQL

Workpackage	*Task*	*Responsabile*
WP3	SQL: Query	Cirillo Francesco Pio

7.1. Query con operatore di aggregazione e join: qualità di vendita dei pescatori

Seleziona il nome e la Partita IVA dei pescatori che vendono Ordinazioni mediamente ad una temperatura superiore a 5 gradi.

```
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. Query nidificata complessa: Ricerca rider sotto sforzo

Selezione dei rider e zona e città associate che non condividono la zona con nessun altro rider e hanno consegnato più della media. Pensata per individuare dove inserire nuovi rider.
Si nota la presenza di 3 diverse query nidificate nella clausola WHERE della query principale.
L'interpretazione complessa è presente nella prima e nella terza in quanto per queste avviene il passaggio di Binding.

```
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. Query insiemistica: Ricerca Clienti che ordinano presso indirizzi diversi dalla residenza

Seleziona i Clienti che vivono nella zona "Praiano1" che hanno effettuato ordinazioni presso un'altra zona.

```
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. Eventuali Altre query

7.4.1. Ricerca pescatori peggiori nelle zone più popolose

Estrai le città composte da almeno una zona popolosa (NAbitanti > 1000) per cui la media del Numero di Bundle in ogni Ordinazione è maggiore di 2. Le tuple sono ordinate in ordine decrescente sulla base della temperatura media al momento del ritiro.

```
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. Viste

Workpackage	*Task*	*Responsabile*
WP4	Viste	Fasolino Alessandra

8.1. Vista *CostoOrdinazioni*

Questa vista permette di visualizzare agevolmente la composizione delle ordinazioni. Sono esposti infatti per ogni ordinazione: numero di bundle componenti, costo e peso totale.

```
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 con Vista: Graduatoria clienti

Questa query consente di visualizzare tutti i clienti e il relativo costo totale di tutte le loro ordinazioni come anche il peso totale di pesce acquistato. Le tuple sono ordinate in ordine decrescente di spesa in questa query nasce con l'idea di individuare il "cliente migliore".

```
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 con Vista: Peso consegne per Città

La query permette di selezionare le città molto estese (estensione>1,5Kmq) e il relativo peso medio di ordinazioni consegnate in quella città.

```
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 inizializzazione: *Associazione (0,N) a (1,N)*

Workpackage	***Task***	***Responsabile***
WP1	Trigger inizializzazione/popolamento database	Cirillo Francesco Pio

Il trigger in questione verifica ad ogni inserimento nella relazione Pescatore e ad ogni eliminazione dalla tabella Pratica che sia sempre verificato che un Pescatore pratichi almeno un tipo di pesca.

```
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();
```

Si sottolinea che al fine di rappresentare in maniera corretta le cardinalità minime di associazioni quali: Lavora (dal lato di Zona), Parte di-Ordinazione (dal lato di Ordinazione) e Possesso (dal lato di Cliente) sarebbero stati necessari altri trigger per il popolamento.

Vista la sua significatività si decide di allegare trigger_almeno_una_pratica.

9.2. Trigger per vincoli aziendali

Workpackage	*Task*	*Responsabile*
WP4	Trigger per vincoli aziendali	Fasolino Alessandra

9.2.1. Trigger1: Calcolo quantità bundle in un'ordinazione

Il trigger in questione automatizza il processo di aggiornamento dell'attributo QuantitaBundle di Ordinazione facendo in modo che questo sia sempre corrispondente al numero di Bundle effettivamente parte dell'Ordinazione.

```
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: Controllo copertura rider

Il trigger in questione assicura che ad un Rider possano essere assegnate solamente Ordinazioni che vanno recapitate nella sua Zona di appartenenza.

```
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: Controllo possesso carta di pagamento

Questo trigger verifica che tutte le Ordinazioni siano pagate usando una carta di pagamento effettivamente corrispondente al Cliente che ha effettuato l'Ordinazione.

```
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();
```