Agriturismo

Relazione per il corso di

Basi di Dati

Maisam Noumi
Alessandro Rebosio
Filippo Ricciotti

6 settembre 2025

Anno Accademico 2024-2025

Indice

Capitolo 1

Analisi dei requisiti

Si ha come obiettivo la realizzazione di un database per la gestione di un agriturismo che offre un'ampia gamma di servizi: ristorazione, ospitalità, piscina, attività con animali, sport e molto altro.

1.1 Intervista

L'agriturismo "Campo Verde" offre una vasta gamma di servizi, tra cui ospitalità, ristorazione, piscina, attività con animali e sport, e necessita di un sistema integrato per la gestione delle prenotazioni, dei clienti e delle attività.

Il sistema dovrà gestire diverse tipologie di servizi, tra cui camere, tavoli al ristorante, lettini in piscina, attività con animali e campo da calcio. Ogni servizio avrà regole specifiche: ad esempio, i lettini in piscina sono prenotabili per fasce orarie di 2 ore, mentre il campo da calcio può essere prenotato per un'ora o più.

Per ogni cliente verranno memorizzati nome, cognome, codice fiscale, indirizzo email e numero di telefono. Al momento della registrazione, verrà creato un account personale con credenziali di accesso alla piattaforma. Ogni cliente potrà effettuare prenotazioni solo se autenticato, e ogni prenotazione verrà registrata con i relativi dettagli (servizio richiesto, data, orario e numero di partecipanti).

I clienti potranno acquistare sia servizi singoli che pacchetti combinati, creati dallo staff o personalizzati in base alle loro esigenze. Ogni pacchetto potrà includere più servizi (es. pernottamento + cena + attività con animali) e avrà un prezzo specifico, con eventuali sconti applicabili in base alla stagione o a promozioni speciali.

I clienti potranno lasciare recensioni per ogni servizio utilizzato, valutandolo con un voto da 1 a 5 stelle e aggiungendo un commento. Le recensioni verranno moderate dallo staff e potranno essere utilizzate per migliorare i servizi offerti.

Il sistema permetterà anche la gestione degli eventi organizzati dall'agriturismo, come tornei di calcio o laboratori con animali. Per ogni evento sarà possibile definire il numero massimo di partecipanti, il prezzo e le date disponibili. I clienti potranno iscriversi agli eventi direttamente dalla piattaforma, ricevendo una conferma e i dettagli organizzativi.

Lo staff dell'agriturismo avrà accesso a una dashboard che mostrerà in tempo reale lo stato delle prenotazioni, l'occupazione delle camere e dei servizi, e le recensioni ricevute. Potranno generare report periodici per analizzare l'andamento delle prenotazioni, le preferenze dei clienti e l'efficacia delle promozioni. Inoltre, il sistema supporterà la gestione di account multipli per lo staff, con diversi livelli di accesso in base al ruolo (es. receptionist, responsabile del ristorante, gestore della piscina).

1.2 Estrazione dei concetti principali

Dall'analisi svolta emergono le principali entità che il sistema dovrà gestire: **Cliente**, **Servizio**, **Prenotazione**, **Pacchetto**, **Evento**, **Recensione** e **Staff**.

Ogni **Cliente** potrà registrarsi fornendo i seguenti attributi:
nome, cognome, codice fiscale, email, città e numero di telefono. All'atto della registrazione, verrà creato un account contenente username e password. Solo i clienti autenticati potranno effettuare prenotazioni. Ogni cliente avrà accesso al proprio storico prenotazioni e potrà lasciare recensioni.

I **Servizi** gestiti comprendono: camere, tavoli ristorante, lettini piscina,
attività con animali e campo da calcio. Ogni servizio presenta regole specifiche: ad esempio, i lettini in piscina sono prenotabili a fasce orarie di due ore, mentre il campo da calcio è prenotabile per una o più ore. Per ciascun servizio devono essere definiti orari, capacità massima, durata prenotabile e penali in caso di cancellazione.

La **Prenotazione** sarà associata a un cliente e a uno o più servizi, e includerà data, orario e stato.

Il sistema gestirà anche **Pacchetti**, ovvero combinazioni di più servizi, con un
prezzo complessivo, eventuali sconti promozionali e possibilità di personalizzazione. I pacchetti potranno essere creati dallo staff o richiesti dai clienti.

Per quanto riguarda gli **Eventi**, come tornei o laboratori, saranno definiti con nome, descrizione, data, prezzo, numero massimo di partecipanti e, se necessario, acconto. I clienti potranno iscriversi online e riceveranno conferma via email.

Le **Recensioni** potranno essere lasciate dai clienti solo dopo aver usufruito di un servizio. Ogni recensione includerà una valutazione da 1 a 5 stelle e un commento testuale. Le recensioni verranno moderate dallo staff prima della pubblicazione.

Lo **Staff** accederà al sistema tramite un account personale con ruolo. Tramite una dashboard, lo staff potrà monitorare in tempo reale le prenotazioni, lo
stato di occupazione dei servizi e le recensioni.

Il sistema permetterà inoltre la generazione di report statistici sull'attività dell'agriturismo.

Capitolo 2

Preogettazione concettuale

In questo capitolo presenteremo lo schema ER, partendo da una versione iniziale e migliorandola passo dopo passo ad arrivare a quella definitiva, attraverso dei raffinamenti.

2.1 Schema scheletro

Dopo aver eseguito l'analisi del dominio iniziale, abbiamo creato uno schema di base con le entità e le relazioni principali, che sarà poi perfezionato nei passaggi successivi.

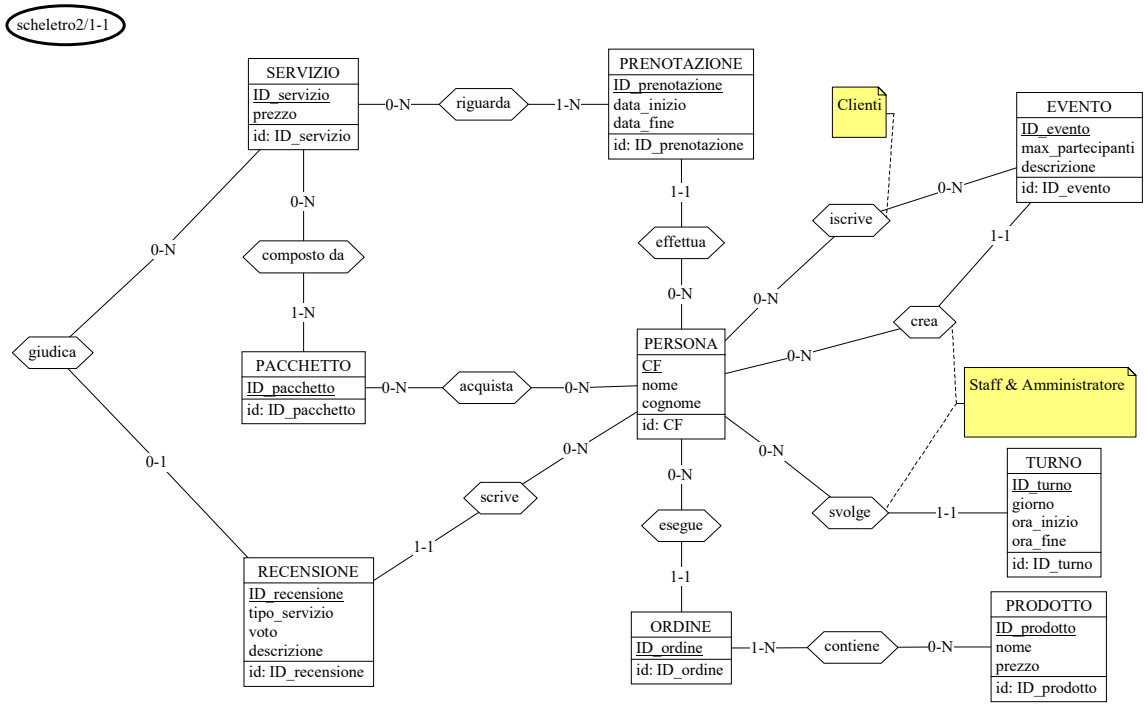

Figura 2.1: Schema ER scheletro

2.2 Raffinamenti proposti

2.2.1 Utente e Dipendente

Nel modello concettuale iniziale la **Persona** raggruppava tutte le possibili interazioni con il sistema: iscrizione, creazione di eventi, acquisto di pacchetti, prenotazioni, ordini e recensioni. Questo approccio, sebbene corretto dal punto di vista logico, risultava poco chiaro perché attribuiva a un'unica entità responsabilità molto eterogenee.

Per migliorare la rappresentazione è stato introdotto un raffinamento mediante generalizzazione/specializzazione: la superclasse **Persona** è stata mantenuta per raccogliere gli attributi comuni (CF, nome, cognome), mentre le funzionalità specifiche sono state assegnate ai sottotipi **Cliente** e **Dipendente**.

La relazione **ospita** tra **Utente** e **Persona** ci permette di associare ad un utente più ospiti, evitando di dover creare un account utente per ogni cliente, un utile comodità nei casi dei gruppi famigliari.

In questo modo i clienti gestiscono attività come acquisti, recensioni, ordini e iscrizioni agli eventi, mentre i dipendenti si occupano della creazione degli eventi e della gestione dei servizi. Tale raffinamento migliora la chiarezza semantica del modello, riduce le ambiguità e riflette meglio la separazione dei ruoli reali all'interno del dominio applicativo.

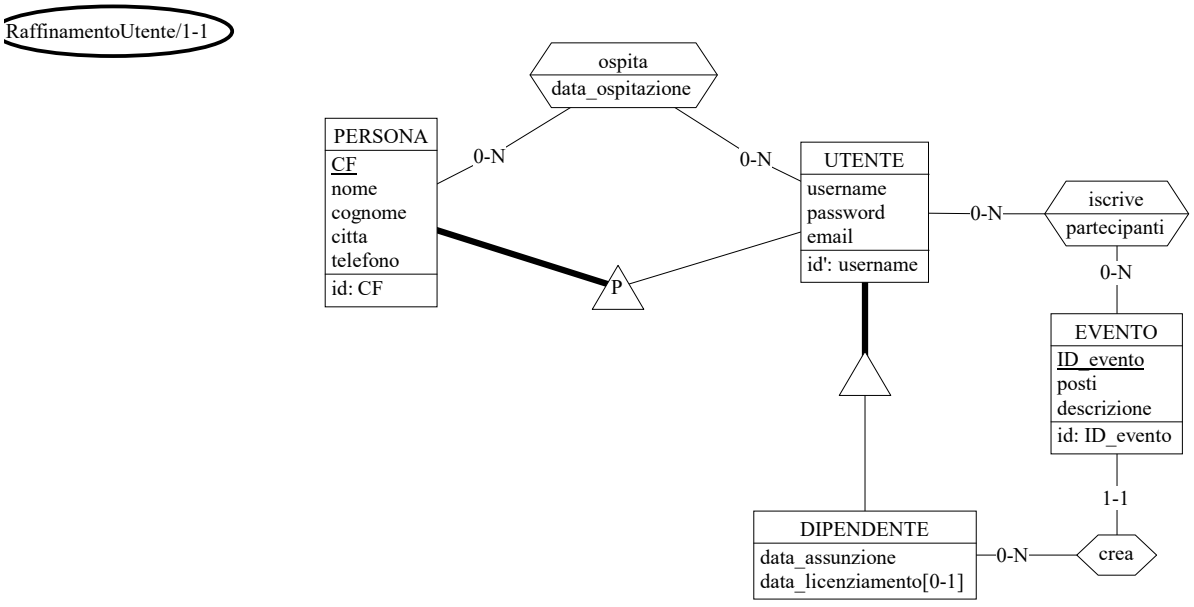

Figura 2.2: Schema ER, raffinamento utente e dipendente

2.2.2 Ruolo e Turno

Nel modello concettuale iniziale i ruoli dei dipendenti erano descritti in maniera statica: un **Dipendente** poteva ricoprire uno o più **Ruoli**, ma non era possibile rappresentare in modo chiaro la distribuzione temporale delle attività lavorative.

Per migliorare la rappresentazione è stato introdotto un raffinamento attraverso l'aggiunta dell'entità **Turno**, caratterizzata da attributi come giorno della settimana, orario di inizio e fine, ed una descrizione testuale. In questo modo diventa possibile modellare i turni di lavoro ricorrenti e collegarli direttamente ai ruoli che li richiedono.

Questa scelta arricchisce la chiarezza semantica del modello, riflette la gestione organizzativa reale e fornisce maggiore flessibilità nella pianificazione delle attività.

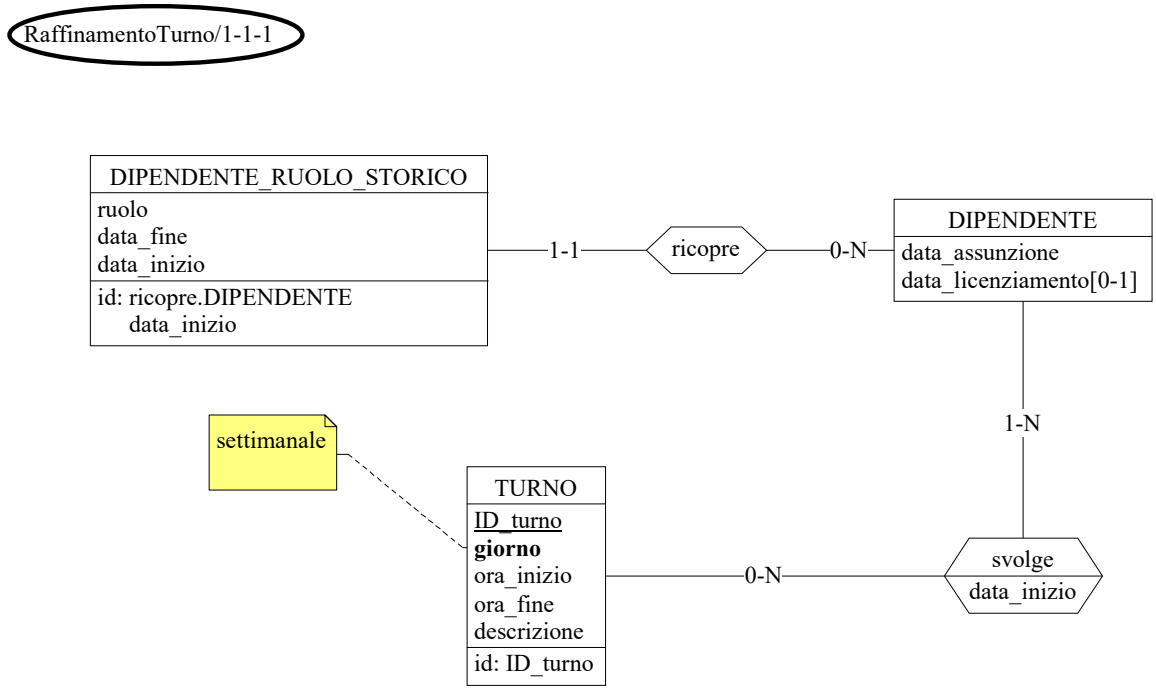

Figura 2.3: Schema ER, raffinamento turno

2.2.3 Servizi/Prenotazioni e Pacchetti

Nel modello iniziale i diversi tipi di servizi (camere, tavoli, lettini, campi da gioco, attività con animali) potevano essere rappresentati come entità distinte, con il rischio però di ridondanza e frammentazione dei dati.

Con il raffinamento si è introdotta una **generalizzazione**: è stata creata la superclasse **Servizio**, che raccoglie gli attributi comuni (ID_servizio, prezzo, tipo_servizio, status), mentre ciascuna tipologia specifica di servizio (Camera, Tavolo, Lettino, Campo da gioco, Attività con animali) è modellata come sottoclasse.

Inoltre, è stato introdotto il legame con l'entità **Prenotazione**, che consente di registrare le informazioni su data di inizio e fine e di associare ogni prenotazione a uno o più servizi specifici tramite la relazione con **Dettagli Prenotazione**. Questo raffinamento permette di gestire correttamente scenari in cui un utente prenota più servizi differenti nello stesso arco temporale.

Infine, i **Pacchetti** offrono un ulteriore livello di astrazione: ciascun pacchetto è composto da uno o più servizi e può essere acquistato dall'utente come un'unica offerta integrata. L'introduzione dei pacchetti, combinata con le prenotazioni, permette di modellare sia l'acquisto di singoli servizi che la sottoscrizione di offerte composte, rendendo il sistema più flessibile e vicino a un contesto reale.

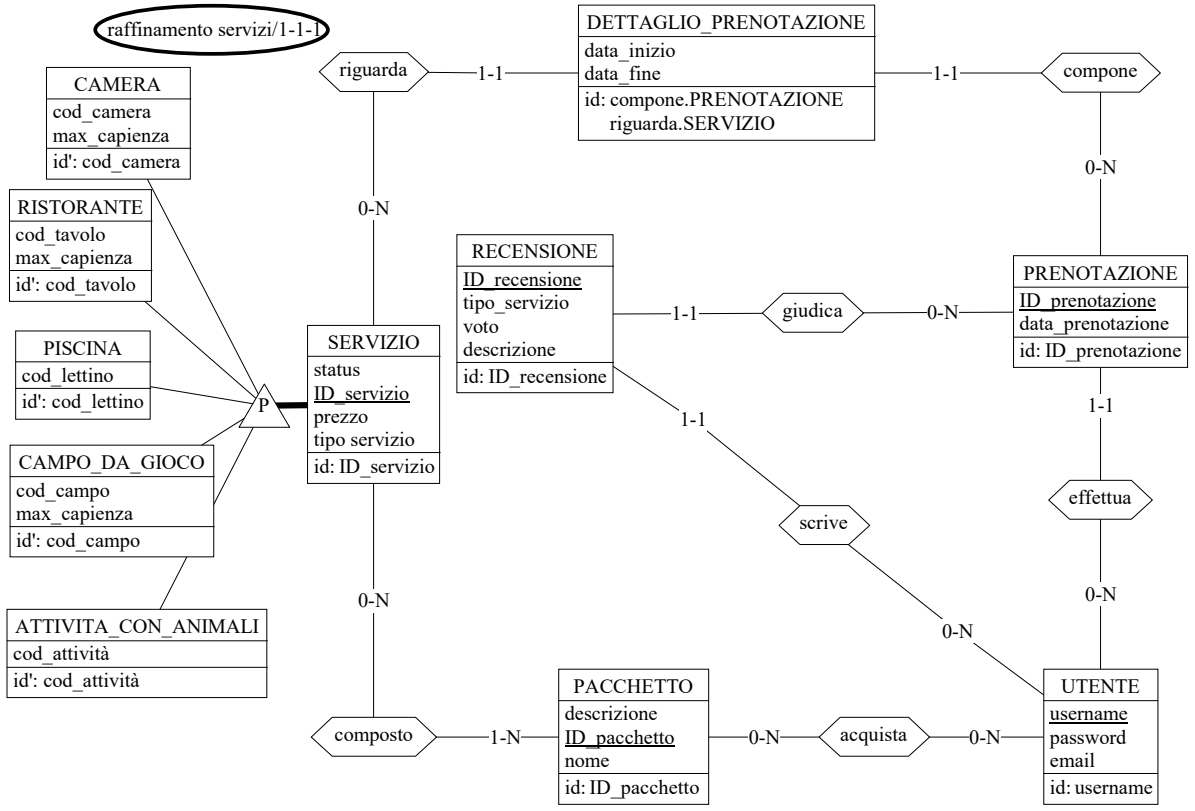

Figura 2.4: Schema ER, raffinamento servizi

2.2.4 Prodotti e ordini

Nel modello concettuale iniziale, la gestione degli ordini e dei prodotti risultava poco dettagliata: un ordine era semplicemente collegato a uno o più prodotti, senza possibilità di specificare informazioni aggiuntive come quantità o prezzo unitario.

Con il raffinamento, è stata introdotta l'entità **Dettaglio Ordine**, che funge da associazione tra **Ordine** e **Prodotto**. Ogni dettaglio ordine consente di memorizzare, per ciascun prodotto incluso in un ordine, la quantità acquistata e il prezzo applicato. Questo permette di rappresentare in modo accurato scenari reali come ordini multiprodotto, applicazione di sconti o variazioni di prezzo nel tempo.

Inoltre, viene mantenuta la generalizzazione tra **Persona** e **Utente**, già introdotta nei raffinamenti precedenti, per distinguere i dati anagrafici comuni da quelli specifici per l'accesso al sistema e la gestione degli ordini. Questo approccio migliora la flessibilità e la chiarezza del modello, consentendo una gestione più efficace delle informazioni relative agli acquisti.

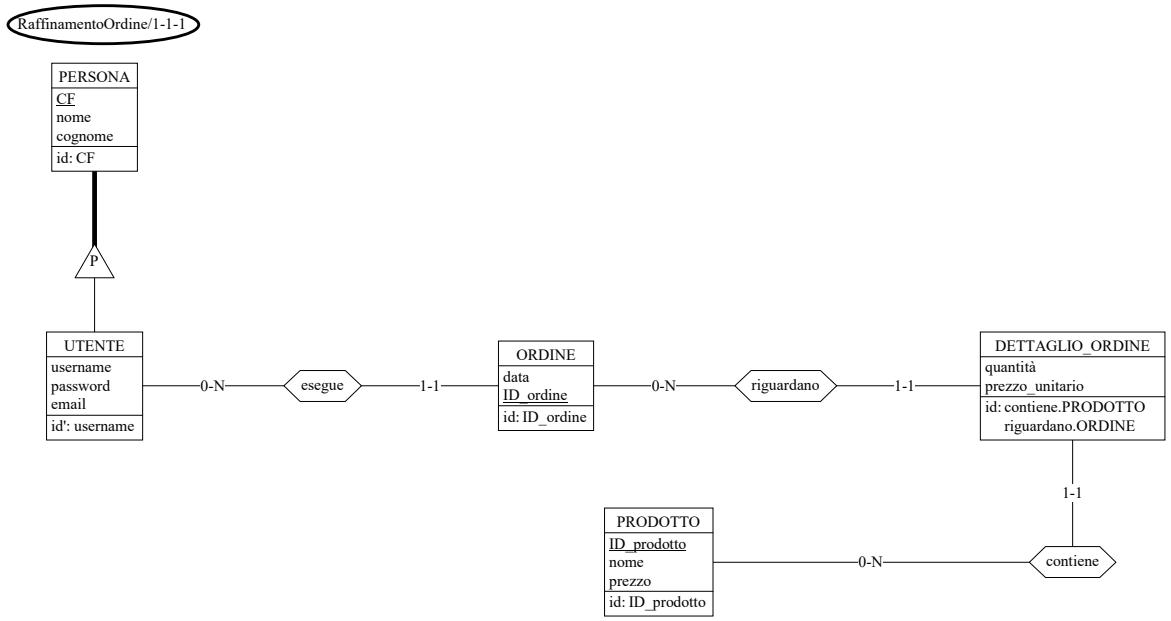

Figura 2.5: Schema ER, raffinamento prodotto

2.3 Schema concettuale finale

Qui di seguito, è presente lo schema concettuale finale con tutti i raffinamenti.

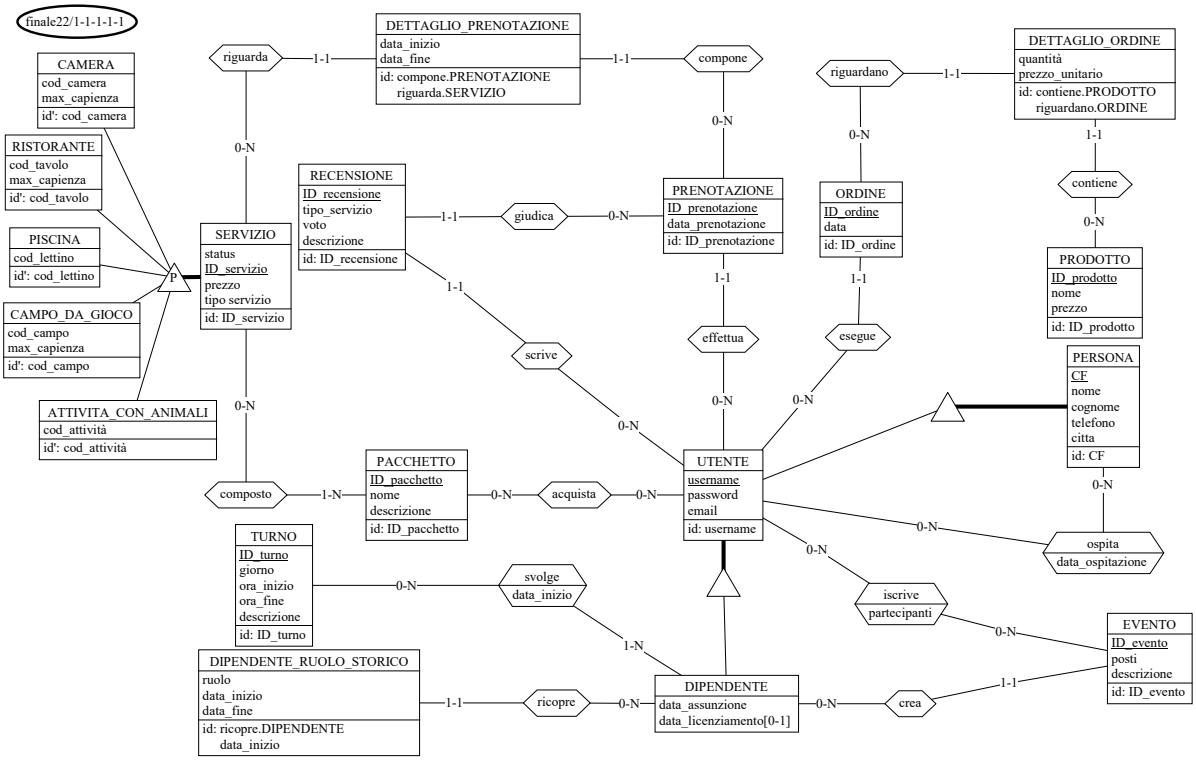

Figura 2.6: Schema ER, schema concettuale finale

Capitolo 3

Progettazione logica

3.1 Stima del volume dei dati

Per progettare il database sono stati stimati i volumi di dati gestiti in un anno da un agriturismo medio (20 camere, ristorante, attività varie). La tabella seguente riporta i valori stimati per entità e associazioni.

Tabella	**VOLUME STIMATO**	**E/A**
PERSONA	4500	E
UTENTE	2500	E
DIPENDENTE	25	E
RUOLO	5	E
PACCHETTO	30	E
SERVIZIO	200	E
CAMERA	20	E
TAVOLO	100	E
CAMPO DA GIOCO	2	E
LETTINO	45	E
ATTIVITÀ CON ANIMALI	3	E
PRENOTAZIONE	6500	E
RECENSIONE	570	E
ORDINE	5000	E
PRODOTTO	150	E
EVENTO	25	E
ASSEGNA RUOLO	30	A
COMPOSTO DA	90	A
ACQUISTA	160	A
DETTAGLIO PRENOTAZIONE	4000	A
DETTAGLIO ORDINE	15000	A
ISCRIVE	500	A
OSPITA	100	A

Tabella 3.1: Stima del volume dei dati

3.2 Descrizione operazioni

In questa sezione vengono riportate le principali operazioni che saranno svolte sulla base di dati. Si è stimata la frequenza con cui ogni operazione viene eseguita in media, nell'arco di una settimana, specificando anche il tipo di utente che la effettua e il tipo di accesso al DB (letture/scritture).

#	**Operazione**	**Op / 7gg**	**Tipo Utente**
1	Registrazione nuovo utente	10	Cliente
2	Autenticazione / login	350	Tutti (Clienti, Staff)
3	Prenotazione servizio	125	Cliente / Reception
4	Modifica / cancellazione prenotazione	30	Cliente / Reception
5	Creazione / modifica pacchetto	3	Staff / Admin
6	Acquisto pacchetto	15	Cliente
7	Inserimento recensione	40	Cliente
8	Gestione inventario / prodotti	10	Staff
9	Creazione evento	1	Staff / Admin
10	Iscrizione evento	15	Cliente
11	Creazione ordine	25	Cliente / Staff
12	Assegnazione / modifica ruolo dipendenti	1	Admin
13	Check-in / Check-out (arrivi/partenze)	50	Reception
14	Aggiornamento stato servizio (pulizia, manut.)	30	Staff operativo
15	Controllo disponibilità servizi (ricerca/filtri)	550	Tutti
16	Generazione report settimanali (occupazione, ricavi)	1	Admin
17	Applicazione sconto/promozione su prenotazione/ordine	5	Staff / Admin
18	Pianificazione turno esistente per dipendente	3	Admin
19	Calcolo tasso di occupazione camere mensile	1	Admin
20	Analisi fasce orarie più richieste per i lettini	1	Admin / Staff
21	Individuazione pacchetto più richiesto	1	Admin
22	Calcolo fatturato mensile per servizio	1	Admin
23	Analisi servizi più inclusi nei pacchetti	1	Admin

Tabella 3.2: Numero stimato di operazioni per settimana, con tipo di utente che le effettua

3.3 Analisi delle operazioni

Di seguito viene riportata un'analisi per alcune delle operazioni principali sul database dell'agriturismo.
Per stimare il carico delle operazioni sul database si è deciso di introdurre i seguenti parametri:

- A_{lett} = numero di accessi in lettura effettuati durante l'operazione. (*read*)
- A_{scr} = numero di accessi in scrittura effettuati durante l'operazione. (*write*)
- Op_{set} = numero medio di volte in cui l'operazione viene eseguita in una settimana (dalla Tabella 3.2).

Nel calcolo degli accessi si stima come doppio il peso degli accessi in scrittura, rispetto a quelli in lettura

1. **Registrazione nuovo utente**

   In questa operazione viene gestita la creazione di un nuovo utente del sistema.

Nome	**Tipo**	**Numero accessi**	**S/L**
UTENTE	E	1	L
PERSONA	E	1	S
UTENTE	E	1	S

   Il flusso dunque si articola in tre parti principali:

   (a) Verificare che l'`username` ed l'`email` non siano già presenti in `UTENTE`, al fine di evitare quindi duplicati.

   (b) Inserimento dei dati anagrafici della nuova persona in `PERSONA`.

   (c) Creazione dell'utente vero e proprio in `UTENTE`, collegato alla relativa persona.

   Sono dunque presenti $A_{lettura} = 1$ e $A_{scrittura} = 2$.

   Pertanto il **costo settimanale** è dato da:

$$\begin{aligned} C_{tot} &= O_{settimana} \cdot (A_{lett} + 2 \cdot A_{scr}) \\ &= 10 \cdot (1 + 2 \cdot 2) \\ &= 10 \cdot 5 = \mathbf{50} \end{aligned}$$

2. **Autenticazione / login**

$$Op_{sett} = 350$$

Nome	**Tipo**	**Numero accessi**	**S/L**
UTENTE	E	1	L

   Quindi si ha una sola operazione di lettura. Pertanto $C_{tot} = 350 \cdot 1 = \mathbf{350}$

3. ## Prenotazione servizio

$$Op_{sett} = 125$$

In media ogni prenotazione è associata a 2 servizi, pertanto per ogni operazione di prenotazione si accede a 2 record in `SERVIZIO` e si scrivono 2 record in `DETTAGLI_PRENOTAZIONE`.

Nome	Tipo	Numero accessi	S/L
UTENTE	E	1	L
PRENOTAZIONE	E	1	S
SERVIZIO	E	2	L
DETTAGLI_PRENOTAZIONE	E	2	S

Quindi in totale si hanno $A_{lett} = 3$ e $A_{scr} = 3$.
Pertanto il costo settimanale è:

$$\mathbf{C_{tot}} = 125 \cdot (3 + 6) = \mathbf{1125}$$

4. ## Modifica / cancellazione prenotazione

$$Op_{sett} = 30$$

In media si hanno $\frac{4000}{2000} = 2$ **servizi per prenotazione**.

Nome	Tipo	Numero accessi	S/L
PRENOTAZIONE	E	1	L
DETTAGLI_PRENOTAZIONE	A	2	L
PRENOTAZIONE	E	1	S
DETTAGLI_PRENOTAZIONE	A	2	S

Quindi in totale si hanno $A_{lett} = 3$ e $A_{scr} = 3$, questo perché:

- Si accede alla tabella `PRENOTAZIONE` per trovare la prenotazione da modificare.
- Si leggono i `DETTAGLI_PRENOTAZIONE` collegati (che sono in media 2 per prenotazione).
- Infine si aggiorna o si elimina la prenotazione (se viene cancellata, si devono rimuovere anche i relativi dettagli).

Pertanto, il costo settimanale è:

$$\mathbf{C_{tot}} = 30 \cdot (3 + 2 \cdot 3) = \mathbf{270}$$

5. ## Creazione / modifica pacchetto

In questa operazione, un membro dello staff oppure un amministratore crea/modifica un pacchetto, ovvero un insieme di servizi offerti ad un prezzo promozionale.

$$Op_{set} = 3$$

In media ogni pacchetto contiene $\frac{90}{30} = 3$ **servizi**

Nome	Tipo	Numero accessi	S/L
PACCHETTO	E	1	S
SERVIZIO	E	3	L
COMPOSTO_DA	A	3	L

Si hanno dunque $A_{lett} = 3$ e $A_{scr} = 4$.
Questo perché:

- si inserisce o aggiorna il record del `PACCHETTO`;
- si leggono in media 3 record di `SERVIZIO` per verificare la loro esistenza;
- si scrivono in media 3 record nella relazione `COMPOSTO_DA`, che collega i servizi al pacchetto.

Pertanto il costo settimanale è:

$$C_{tot} = 3 \cdot (3 + 2 \cdot 4) = \mathbf{33}$$

6. Acquisto pacchetto

$$Op_{sett} = 15$$

In media ogni pacchetto contiene $\frac{90}{30} = 3$ **servizi**.

Nome	Tipo	Numero accessi	S/L
UTENTE	E	1	L
PACCHETTO	E	1	L
ACQUISTA	A	1	S
SERVIZIO	E	3	L
DETTAGLI_PRENOTAZIONE	A	3	S

Quindi in totale si hanno $A_{lett} = 5$ e $A_{scr} = 4$.

Questo perché:

- si effettua in `UTENTE` una lettura per verificare chi effettua l'acquisto;
- si legge il `PACCHETTO` scelto;
- si scrive la relazione `ACQUISTA` per collegare utente e pacchetto;
- si leggono i `SERVIZI` contenuti nel pacchetto;
- si inseriscono i corrispondenti record in `DETTAGLI_PRENOTAZIONE`.

Pertanto il costo settimanale è:

$$\mathbf{C_{tot}} = 15 \cdot (5 + 2 \cdot 4) = \mathbf{195}$$

7. Inserimento recensione

$$Op_{sett} = 40$$

In media ogni pacchetto contiene $\frac{90}{30} = 3$ **servizi**.

Nome	Tipo	Numero accessi	S/L
UTENTE	E	1	L
SERVIZIO	E	1	L
RECENSIONE	E	1	S

Quindi in totale si hanno $A_{lett} = 2$ e $A_{scr} = 1$.

Questo perché:

- si legge l'`UTENTE` che lascia la recensione;
- si verifica il `SERVIZIO` a cui si riferisce;
- si inserisce un record in `RECENSIONE`.

Pertanto il costo settimanale è:

$$\mathbf{C_{tot}} = 40 \cdot (2 + 2 \cdot 1) = \mathbf{160}$$

8. Gestione inventario / prodotti

Per gestione inventario si intende l'aggiornamento dei prodotti del ristorante/bar da parte dello staff.

$$Op_{sett} = 10$$

Nome	Tipo	Numero accessi	S/L
PRODOTTO	E	1	L
PRODOTTO	E	1	S

Quindi in totale si hanno $A_{lett} = 1$ e $A_{scr} = 1$.

Questo perché:

- prima leggiamo il record del `PRODOTTO` per verificare l'esistenza di dati attuali
- per poi aggiornarlo (se esiste) o inserirlo

Pertanto il costo settimanale è:

$$\mathbf{C_{tot}} = 10 \cdot (1 + 2 \cdot 1) = \mathbf{30}$$

9. Creazione evento

$$Op_{sett} = 1$$

Nome	Tipo	Numero accessi	S/L
EVENTO	E	1	S

Questo perchè l'operazione comporta semplicemente l'inserimento di un nuovo `EVENTO`.

Quindi si ha solamente un'operazione di scrittura:

$$\mathbf{C_{tot}} = 1 \cdot (0 + 2 \cdot 1) = \mathbf{2}$$

10. Iscrizione evento

$$Op_{sett} = 15$$

In media ad un evento si iscrivono $\frac{500}{25} = 20$ **iscritti**

Nome	Tipo	Numero accessi	S/L
UTENTE	E	1	L
EVENTO	E	1	L
ISCRIVE	A	1	S

Questo perché:

- si legge l'`UTENTE` che si iscrive;
- si legge l'`EVENTO` scelto;
- si inserisce un nuovo record nella `ISCRIVE`.

Quindi: $A_{lett} = 2$, $A_{scr} = 1$. Pertanto il costo settimanale è:

$$\mathbf{C_{tot}} = 15 \cdot (2 + 2 \cdot 1) = \mathbf{60}$$

11. Creazione Ordine

$$Op_{sett} = 25$$

In media ogni ordine contiene $\frac{15000}{5000} = 3$ **prodotti**.

Nome	Tipo	Numero accessi	S/L
UTENTE	E	1	L
ORDINE	E	1	S
PRODOTTO	E	3	L
DETTAGLI_ORDINE	A	3	S

Questo perché:

- si legge l'`UTENTE` che effettua l'ordine;

- si inserisce un nuovo record in `ORDINE`;
- si leggono in media 3 `PRODOTTI`;
- e si scrivono 3 record in `DETTAGLI_ORDINE`.

Quindi: $A_{lett} = 4$, $A_{scr} = 4$. Pertanto il costo settimanale è:

$$\mathbf{C_{tot}} = 25 \cdot (4 + 2 \cdot 4) = \mathbf{300}$$

12. Assegnazione / modifica ruolo dipendenti

$$Op_{sett} = 1$$

Nome	Tipo	Numero accessi	S/L
DIPENDENTE	E	1	L
RUOLO	E	1	L
ASSEGNA_RUOLO	A	1	S

In questa operazione si assegna o modifica il ruolo di un dipendente:

- Si legge il `DIPENDENTE`.
- Si legge il `RUOLO`.
- Si inserisce/aggiorna l'associazione in `ASSEGNA_RUOLO`.

Totale: $A_{lett} = 2$, $A_{scr} = 1$.

$$C_{tot} = 1 \cdot (2 + 2 \cdot 1) = \mathbf{4}$$

13. Check-in / Check-out

Questa operazione gestisce l'arrivo o la partenza di un cliente.

$$Op_{sett} = 50$$

Nome	Tipo	Numero accessi	S/L
PRENOTAZIONE	E	1	L
DETTAGLI_PRENOTAZIONE	A	2	L
DETTAGLI_PRENOTAZIONE	A	2	S

- Si legge la `PRENOTAZIONE`.
- Si leggono i `DETTAGLI_PRENOTAZIONE` collegati (in media ci sono 2).
- Si aggiornano gli stessi dettagli con lo stato di check-in/check-out.

Quindi ci sono $A_{lett} = 3$ e $A_{scr} = 2$.
Pertanto il costo settimanale è dato da:

$$\mathbf{C_{tot}} = 50 \cdot (3 + 2 \cdot 2) = \mathbf{350}$$

14. Aggiornamento stato servizio

Questa operazione spiega quante volte lo staff aggiorna lo stato di un servizio.

$$Op_{sett} = 30$$

Nome	Tipo	Numero accessi	S/L
SERVIZIO	E	1	L
SERVIZIO	E	1	S

- Si legge il `SERVIZIO`.
- Si aggiorna lo stato (es. disponibile, in manutenzione, occupato).

Quindi ci sono $A_{lett} = 1$ e $A_{scr} = 1$.

$$\mathbf{C_{tot}} = 30 \cdot (1 + 2 \cdot 1) = \mathbf{90}$$

15. Controllo disponibilità servizi

Un utente o lo staff verificano la disponibilità di un servizio in un dato periodo. Si assume che gli utenti guardino in media 3 servizi e che, quindi, non consultino solamente uno.

$$Op_{sett} = 550$$

Nome	Tipo	Numero accessi	S/L
SERVIZIO	E	3	L

- Gli utenti ricercano tra più `SERVIZI` (camere, tavoli, attività, ecc.).
- Operazione di sola lettura.

Quindi solamente $A_{lett} = 3$.

$$\mathbf{C_{tot}} = 550 \cdot 3 = \mathbf{1650}$$

16. Generazione report settimanali

Lo staff genera report riassuntivi sulle prenotazioni e sugli ordini. In media un report include 20 prenotazioni e 10 ordini.

$$Op_{sett} = 1$$

Nome	Tipo	Numero accessi	S/L
PRENOTAZIONE	E	20	L
ORDINE	E	10	L

- L'admin richiede statistiche (occupazione, ricavi).

- Molte letture, nessuna scrittura.

L'operazione è di sola lettura perché consiste nell'estrarre dati da più entità per creare il report.

$$\mathbf{C_{tot}} = 1 \cdot (30) = \mathbf{30}$$

17. **Applicazione sconto / promozione** Un amministratore o un membro dello staff inserisce o modifica una promozione su pacchetti o servizi.

$$Op_{sett} = 5$$

Nome	Tipo	Numero accessi	S/L
PRENOTAZIONE / ORDINE	E	1	L
PRENOTAZIONE / ORDINE	E	1	S

- Si legge la prenotazione o l'ordine.
- Si aggiorna applicando sconto o promozione.

Quindi si ha $A_{lett} = 1$ e $A_{scr} = 1$.

$$\mathbf{C_{tot}} = 5 \cdot (1 + 2 \cdot 1) = \mathbf{15}$$

18. **Pianificazione turno esistente per dipendente**

$$Op_{sett} = 3$$

Nome	Tipo	Numero accessi	S/L
DIPENDENTE	E	1	L
SVOLGE	A	1	S
TURNO	E	1	L

Questo perché:

- `SVOLGE` è l'associazione che collega il dipendente al turno.
- `TURNO` è l'entità interna che rappresenta l'unità di pianificazione.

Quindi si hanno: $A_{lett} = 2$, $A_{scr} = 1$.

$$C_{tot} = 3 \cdot (2 + 2 \cdot 1) = \mathbf{12}$$

19. **Calcolo tasso di occupazione camere**

$$Op_{sett} = 1$$

È un'operazione gestionale volta ad individuare i mesi/scenari a bassa occupazione e pianificare promozioni mirate.

Nome	Tipo	Numero accessi	S/L
PRENOTAZIONE	E	2000	L
DETTAGLI_PRENOTAZIONE	A	4000	L
CAMERA	E	20	L
SERVIZIO	E	200	L
UTENTE	E	2000	L

Questo perché:

- si leggono tutte le `PRENOTAZIONI` effettuate,
- si consultano i `DETTAGLI_PRENOTAZIONE` per sapere quali camere sono state usate,
- si leggono le entità `CAMERA` e `SERVIZIO` per filtrare i soli servizi di tipo "camera",
- si accede agli `UTENTI` collegati per fini di reportistica (segmentazione per tipologia di cliente).

Si hanno quindi: $A_{\text{lett}} = 2000 + 4000 + 20 + 200 + 2000 = 8220$ Pertanto il costo settimanale è dato da:

$$\mathbf{C_{tot}} = 1 \cdot (8220 + 0) = \mathbf{8220}$$

20. Analisi fasce orarie più richieste per i lettini

È un'operazione volta a dimensionare i turni del personale sui picchi di domanda, al fine di migliorare il servizio e gestire al meglio le risorse.

Nome	Tipo	Numero accessi	S/L
DETTAGLI_PRENOTAZIONE	A	4000	L
LETTINO	E	45	L
PRENOTAZIONE	E	2000	L
UTENTE	E	2000	L

Questo perché:

- i `DETTAGLI_PRENOTAZIONE` permettono di risalire a quali lettini sono stati richiesti e quando,
- la tabella `LETTINO` serve a distinguere i diversi lettini,
- da `PRENOTAZIONE` si ricavano le date/ore di utilizzo,
- si leggono gli `UTENTI` per distinguere tipologie di clientela (es. ospiti giornalieri vs soggiornanti).

Si ha quindi $A_{\text{lett}} = 4000 + 45 + 2000 + 2000 = 8045$

$$\mathbf{C_{tot}} = 1 \cdot (8045 + 0) = \mathbf{8045}$$

21. Individuazione pacchetto più richiesto

$$Op_{sett} = 1$$

Per promuovere l'esperienza più popolare e migliorare i servizi collegati.

Nome	Tipo	Numero accessi	S/L
ACQUISTA	A	160	L
PACCHETTO	E	30	L
UTENTE	E	160	L
COMPOSTO_DA	A	90	L
SERVIZIO	E	200	L

Questo perché:

- si leggono le relazioni `ACQUISTA` per sapere quanti pacchetti sono stati acquistati,
- da `PACCHETTO` si individuano i pacchetti specifici,
- gli `UTENTI` permettono di segmentare la clientela,
- da `COMPOSTO_DA` e `SERVIZIO` si vedono i contenuti del pacchetto più richiesto.

Si ha $A_{\text{lett}} = 160 + 30 + 160 + 90 + 200 = 640$

$$\mathbf{C_{tot}} = 1 \cdot (640 + 0) = \mathbf{640}$$

22. Calcolo fatturato per servizio

$$Op_{sett} = 0.25$$

Per valutare la redditività dei singoli servizi e riallocare il budget.

Nome	Tipo	Numero accessi	S/L
ORDINE	E	5000	L
DETTAGLI_ORDINE	A	15000	L
SERVIZIO	E	200	L
PRODOTTO	E	500	L
UTENTE	E	5000	L

Questo perché:

- si leggono gli `ORDINI` registrati,
- da `DETTAGLI_ORDINE` si ottiene la quantità/prezzo dei prodotti acquistati,
- da `SERVIZIO` si mappa l'ordine a uno specifico servizio (es. ristorante, spa),
- i `PRODOTTI` collegati aiutano a distinguere la parte di ricavi da beni materiali,
- gli `UTENTI` permettono di segmentare la clientela in termini di spesa.

Si hanno quindi $A_{\text{lett}} = 5000 + 15000 + 200 + 500 + 5000 = 25700$

$$\mathbf{C_{tot}} = 1 \cdot (25700 + 0) = \mathbf{25700}$$

23. **Analisi servizi più inclusi nei pacchetti**

$$Op_{sett} = 1$$

Questa operazione è necessaria a capire le combinazioni preferite e progettare nuovi pacchetti garantendo i servizi più richiesti.

Nome	Tipo	Numero accessi	S/L
PACCHETTO	E	30	L
SERVIZIO	E	200	L
COMPOSTO_DA	A	90	L
ACQUISTA	A	160	L
UTENTE	E	160	L

Questo perché:

- si leggono i `PACCHETTI` definiti,
- si consultano i `SERVIZI` inclusi tramite la relazione `COMPOSTO_DA`,
- la relazione `ACQUISTA` mostra quali pacchetti sono effettivamente scelti,
- si leggono gli `UTENTI` per analizzare preferenze della clientela.

Si ha $A_{\text{lett}} = 30 + 200 + 90 + 160 + 160 = 640$

$$\mathbf{C_{tot}} = 1 \cdot (640 + 0) = \mathbf{640}$$

3.4 Analisi delle ridondanze

Nel modello ER definitivo sono state individuate quattro ridondanze:

- `prezzo_unitario` in `DETTAGLIO_ORDINE`
- `quantita` in `DETTAGLIO_ORDINE`
- `tipo_servizio` in `RECENSIONE`
- `status` in `SERVIZIO`

3.4.1 Analisi attributo `prezzo_unitario` in `DETTAGLIO_ORDINE`

L'attributo `prezzo_unitario` è ridondante perché il prezzo è già presente in `PRODOTTO`.

Mantendo la **ridondanza** il prezzo verrebbe letto direttamente dalla riga di `DETTAGLIO_ORDINE`. **Senza ridondanza** ad ogni dettaglio d'ordine occorrerebbe un join con `PRODOTTO`, raddoppiando i costi di accesso.

Op.	Descrizione	Con rid.	Senza rid.
11	Creazione ordine (sett.)	300	600
16	Report settimanali (sett.)	30	60
22	Fatturato per servizio ($Op_{set} = 0.25$)	6 425	12 850
Totale		**6 755**	**13 510**

Tabella 3.3: Analisi ridondanza `prezzo_unitario`

A fronte di questa analisi, decidiamo di mantenere l'attributo con la ridondanza perché dimezza i costi e preserva i prezzi storici degli ordini.

3.4.2 Analisi attributo `quantita` in `DETTAGLIO_ORDINE`

La quantità è ridondante perché si potrebbero inserire q tuple identiche. Mantenendo **la ridondanza**, si salverebbe un solo record con il campo `quantita`, **Senza ridondanza** occorrerebbe invece replicare ogni voce d'ordine q volte, aumentando linearmente i costi con $\bar{q}$.

Op.	Descrizione	Con rid.	Senza rid.
11	Creazione ordine (sett.)	300	$300 \times \bar{q}$
16	Report settimanali (sett.)	30	$30 \times \bar{q}$
22	Fatturato per servizio ($Op_{set} = 0.25$)	6 425	$6\,425 \times \bar{q}$
Totale		**6 755**	$\gg 6\,755$

Tabella 3.4: Analisi ridondanza `quantita`

A fronte di questa analisi, conviene mantenere la ridondanza perché riduce drasticamente il volume dei dati.

3.4.3 Analisi attributo `tipo_servizio` in `RECENSIONE`

Il tipo del servizio è già recuperabile da `SERVIZIO`. Mediante la **ridondanza** la query sulle recensioni accede direttamente all'attributo. **Senza ridondanza** servirebbe un join tra `RECENSIONE` e `SERVIZIO`, raddoppiando i costi.

Op.	Descrizione	Con rid.	Senza rid.
7	Inserimento recensione (sett.)	160	320
20	Analisi fasce orarie (sett.)	8 045	16 090
23	Analisi pacchetti (sett.)	640	1 280
Totale		**8 845**	**17 690**

Tabella 3.5: Analisi ridondanza `tipo_servizio`

A fronte di questa analisi, decidiamo di mantenere la ridondanza perché evita join costosi nelle query statistiche.

3.4.4 Analisi attributo `status` in `SERVIZIO`

La disponibilità di un servizio può essere calcolata dalle prenotazioni. Con la **ridondanza** si legge direttamente l'attributo `status` in `SERVIZIO`. Invece, **senza ridondanza** occorre cercare nelle prenotazioni attive di `DETTAGLIO_PRENOTAZIONE`, con più join e accessi.

Op.	Descrizione	Con rid.	Senza rid.
3	Prenotazione servizio (sett.)	1 125	2 250
13	Check-in / Check-out (sett.)	350	700
14	Aggiornamento stato (sett.)	90	180
15	Controllo disponibilità (sett.)	1 650	3 300
Totale		**3 215**	**6 430**

Tabella 3.6: Analisi ridondanza `status`

A fronte di questa analisi, ci conviene mantenere la ridondanza perché dimezza i costi delle operazioni più frequenti del sistema.

Riepilogo analisi delle ridondanze

Tutte le ridondanze vengono mantenute poiché:

- `prezzo_unitario` e `quantita` assicurano efficienza e storicità degli ordini.
- `tipo_servizio` evita join costosi sulle recensioni.
- `status` riduce drasticamente i costi delle prenotazioni e dei controlli disponibilità.

3.5 Riepilogo operazioni

Nella tabella qui sotto, riprendiamo l'analisi delle operazioni, facendone un riepilogo, aggiungendo il costo totale per ogni operazione.

#	**Operazione**	**Costo tot. / 7gg**	**Tipo Utente**
1	Registrazione nuovo utente	50	Cliente
2	Autenticazione / login	350	Tutti (Clienti, Staff)
3	Prenotazione servizio	1125	Cliente / Reception
4	Modifica / cancellazione prenotazione	270	Cliente / Reception
5	Creazione / modifica pacchetto	33	Staff / Admin
6	Acquisto pacchetto	195	Cliente
7	Inserimento recensione	160	Cliente
8	Gestione inventario / prodotti	30	Staff
9	Creazione evento	2	Staff / Admin
10	Iscrizione evento	60	Cliente
11	Creazione ordine	300	Cliente / Staff
12	Assegnazione / modifica ruolo dipendenti	4	Admin
13	Check-in / Check-out (arrivi/partenze)	350	Reception
14	Aggiornamento stato servizio (pulizia, manut.)	90	Staff operativo
15	Controllo disponibilità servizi (ricerca/filtri)	1650	Tutti
16	Generazione report settimanali (occupazione, ricavi)	30	Admin
17	Applicazione sconto/promozione su prenotazione/ordine	15	Staff / Admin
18	Pianificazione turno esistente per dipendente	12	Admin
19	Calcolo tasso di occupazione camere mensile	8220	Admin
20	Analisi fasce orarie più richieste per i lettini	8045	Admin / Staff
21	Individuazione pacchetto più richiesto	640	Admin
22	Calcolo fatturato mensile per servizio	6425	Admin
23	Analisi servizi più inclusi nei pacchetti	640	Admin

Tabella 3.7: Coste delle operazioni per settimana, con tipo di utente che le effettua

3.6 Raffinamento dello schema

Definiti gli attributi da mantenere nella base dati, si passerà a perfezionare lo schema ER, ridefinendo gli elementi non traducibili immediatamente nel modello relazionale. Lavoreremo per tappe fino a raggiungere lo schema logico corrispondente.

3.6.1 Rimozione gerarchie

Gerarchia di Servizio

Lo schema presenta una gerarchia **totale ed esclusiva** dell'entità `SERVIZIO` nelle sottoclassi `CAMERA`, `TAVOLO`, `LETTINO`, `CAMPO_DA_GIOCO`, `ATTIVITA_CON_ANIMALI`.

Optiamo per un **collasso verso il basso** mantenendo l'entità padre `SERVIZIO` come tabella contenente gli attributi comuni, mentre le sottoclassi diventano tabelle separate collegate tramite chiave esterna. Questa scelta preserva l'integrità referenziale e permette di mantenere le specificità di ogni tipo di servizio.

La struttura risultante sarà:

- `SERVIZIO(ID_servizio, prezzo, tipo_servizio, status)`
- `CAMERA(ID_servizio, cod_camera, max_capienza)`
- `TAVOLO(ID_servizio, cod_tavolo, max_capienza)`
- `LETTINO(ID_servizio, cod_lettino)`
- `CAMPO_DA_GIOCO(ID_servizio, cod_campo, max_capienza)`
- `ATTIVITA_CON_ANIMALI(ID_servizio, cod_attività)`

Dove `ID_servizio` nella tabella `SERVIZIO` funge da chiave primaria e come chiave esterna in tutte le tabelle figlie.

Gerarchia di Persona

L'entità `PERSONA` è generalizzata in `UTENTE` e `DIPENDENTE`. Scegliamo di mantenere le tre tabelle collegate tramite chiavi esterne, utilizzando `CF` come chiave primaria e rendendo `username` univoco in `UTENTE` e `DIPENDENTE`:

- `PERSONA(CF, nome, cognome)`
- `UTENTE(username : PERSONA, username UNIQUE, password, email)`
- `DIPENDENTE(CF : PERSONA, username UNIQUE, data_assunzione)`

Questa scelta permette di:

- Utilizzare `CF` come identificatore univoco per tutte le persone del sistema
- Garantire l'unicità di `username` per le operazioni di login e autenticazione

3.6.2 Reificazione associazioni molti-a-molti

Le seguenti associazioni molti-a-molti sono state reificate in tabelle, poiché le associazioni di questo tipo non possono essere rappresentate direttamente negli schemi relazionali:

- `COMPOSTO_DA(ID_pacchetto, ID_servizio)`
- `ACQUISTA(username, ID_pacchetto, data)`
- `ISCRIVE(username, ID_evento, data)`

3.6.3 Scelta degli identificatori principali

Per semplificare le query e garantire consistenza, abbiamo scelto i seguenti identificatori:

- `SERVIZIO`: `ID_servizio` (surrogato)
- `PERSONA`: `CF` (naturale)
- `UTENTE`: `username` (naturale)
- `DIPENDENTE`: `username` (naturale, da UTENTE)
- `PRENOTAZIONE`: `ID_prenotazione` (surrogato)
- `ORDINE`: `ID_ordine` (surrogato)
- `RECENSIONE`: `ID_recensione` (surrogato)
- `EVENTO`: `ID_evento` (surrogato)
- `PACCHETTO`: `ID_pacchetto` (surrogato)
- `PRODOTTO`: `ID_prodotto` (surrogato)
- `RUOLO`: `ID_ruolo` (surrogato)
- `TURNO`: `ID_turno` (surrogato)

Gli identificatori surrogati sono stati preferiti laddove non esisteva un identificatore naturale stabile. Per le entità con codici fissi (es. `CAMERA`, `TAVOLO`) è stato mantenuto l'identificatore naturale come attributo, ma la chiave primaria rimane `ID_servizio` ereditata dalla tabella padre.

3.7 Schema relazionale finale

La fase di riorganizzazione è ora conclusa e lo schema ottenuto è immediatamente traducibile in relazioni relazionali. Di seguito presentiamo lo schema logico.

3.7.1 Schema relazionale (grafico)

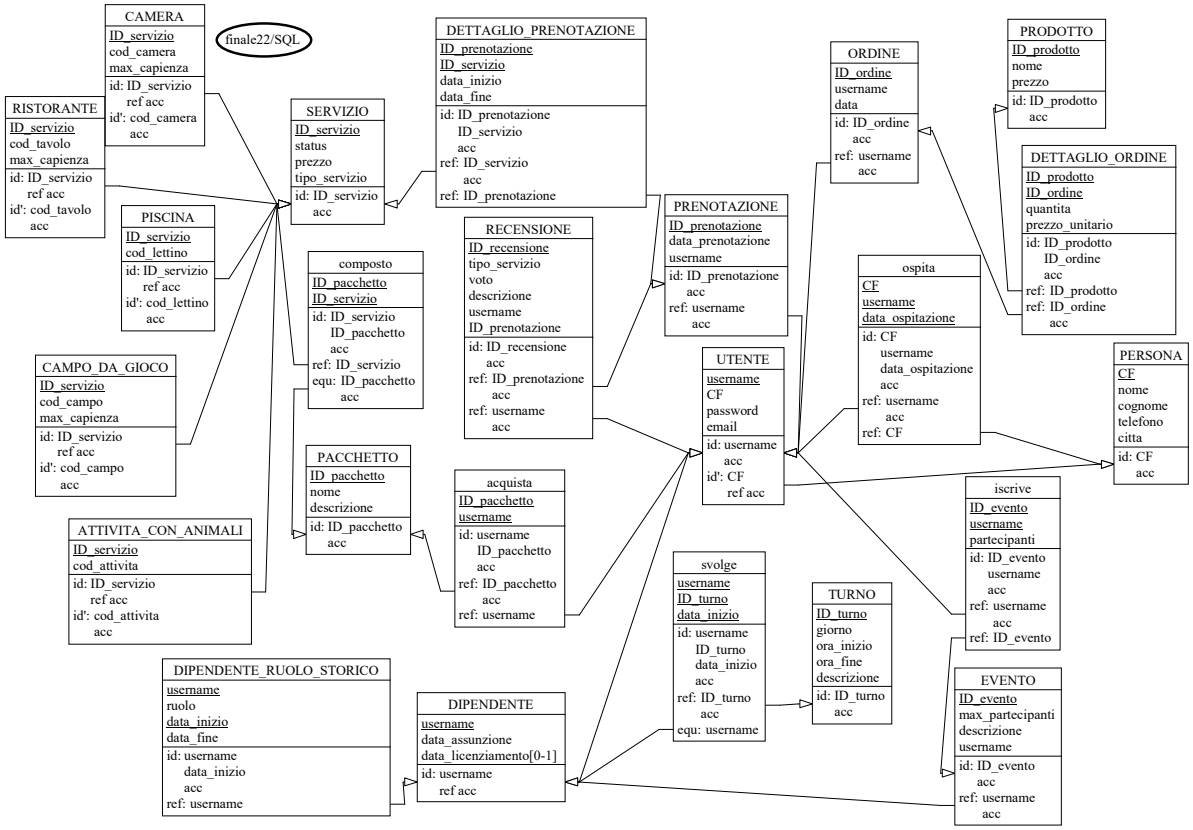

Figura 3.1: Schema logico, schema relazionale finale

3.7.2 Schema relazionale (testuale)

Di seguito viene riportata la traduzione in schema relazionale delle entità e relazioni progettate.

PERSONA (CF, nome, cognome)

OSPITA (CF_ospite : UTENTE, CF_utente : PERSONA, data_ospitazione)

UTENTE (CF : PERSONA, username, password, email)

TURNO (ID_turno, giorno, ora_inizio, ora_fine, descrizione)

TAVOLO (ID_servizio : SERVIZIO, cod_tavolo, max_capienza)

SVOLGE (ID_ruolo : RUOLO, ID_turno : TURNO)

SERVIZIO (ID_servizio, status, prezzo, tipo_servizio)

RUOLO (ID_ruolo, tipo_ruolo)

RECENSIONE (ID_recensione : PRENOTAZIONE, tipo_servizio, voto, descrizione, CF : PERSONA)

PRODOTTO (ID_prodotto, nome, prezzo)

PRENOTAZIONE (ID_prenotazione, CF : PERSONA)

PACCHETTO (ID_pacchetto, descrizione, nome)

ORDINE (ID_ordine, data, CF : PERSONA)

ISCRIVE (ID_evento : EVENTO, CF : PERSONA)

LETTINO (cod_lettino : LETTINO, ID_servizio : SERVIZIO)

EVENTO (ID_evento, max_partecipanti, descrizione, CF : PERSONA)

DIPENDENTE (CF : PERSONA, data_assunzione)

DETTAGLI_PRENOTAZIONE (ID_prenotazione : PRENOTAZIONE, data_inizio, data_fine, ID_servizio : SERVIZIO)

DETTAGLI_ORDINE (ID_prodotto : PRODOTTO, quantita, prezzo_unitario, ID_ordine : ORDINE)

COMPOSTO_DA (ID_pacchetto : PACCHETTO, ID_servizio : SERVIZIO)

CAMPO_DA_GIOCO (ID_servizio : SERVIZIO, max_capienza, cod_campo : CAMPO_DA_GIOCO)

CAMERA (ID_servizio : SERVIZIO, max_capienza, cod_camera : CAMERA)

ATTIVITA_CON_ANIMALI (ID_servizio : SERVIZIO, cod_attivita : ATTIVITA_CON_ANIMALI)

Capitolo 4

Progettazione della Base di Dati

Ora che abbiamo creato la nostra base di dati, mappiamo le nostre relazioni in tabelle, per poi utilizzarle nel nostro

4.1 Check

Nel file SQL sono stati utilizzati vincoli di tipo `CHECK` per definire alcuni domini e assicurare semplici proprietà degli attributi. Di seguito un elenco dei controlli presenti nel file:

- `TURNO`: `CHECK (ora_inizio < ora_fine)` per assicurare orari coerenti.
- `DIPENDENTE_RUOLO_STORICO`: `CHECK (data_inizio <= data_fine)` per la validit00E0 degli intervalli.
- `SERVIZIO` e `PRODOTTO`: `CHECK (prezzo >= 0)` per il prezzo non negativo.
- Tabelle con attributi di capienza (`TAVOLO`, `CAMPO_DA_GIOCO`, `CAMERA`): `CHECK (max_capienza >= 1)`.
- `DETTAGLI_PRENOTAZIONE`: `CHECK (data_inizio <= data_fine)`.
- `RECENSIONE`: `CHECK (voto BETWEEN 1 AND 5)`.
- `DETTAGLI_ORDINE`: `CHECK (quantita > 0)` e `CHECK (prezzo_unitario >= 0)`.
- `EVENTO`: `CHECK (posti >= 0)` per il numero di posti rimasti non negativo.
- `iscrive`: `CHECK (partecipanti > 0)` per il numero di partecipanti non negativo o nullo.

Nota I vincoli `CHECK` sono stati usati per vincoli locali semplici (dominio di attributi e relazione tra date/orari). Vincoli piu complessi (ad es. check che richiedono subquery) non sono utilizzati come `CHECK` nel file finale; per logiche relazionali complesse sono stati impiegati i trigger.

4.2 Viste

TODO scegliere.

4.3 Trigger

Nel Database sono presenti i seguenti trigger (riportati qui tali e quali come definiti nel file):

```
1
2       DELIMITER \$\$
3
4       -- Trigger: impedisce di eliminare l'ultimo turno di un
    dipendente
5       CREATE TRIGGER trg_check_dipendente_turni
6       BEFORE DELETE ON svolge
7       FOR EACH ROW
8       BEGIN
9       IF NOT EXISTS (
10      SELECT 1 FROM svolge
11      WHERE username = OLD.username
12      AND ID_turno <> OLD.ID_turno
13      ) THEN
14      SIGNAL SQLSTATE '45000'
15      SET MESSAGE_TEXT = 'Un dipendente deve avere almeno un turno';
16      END IF;
17      END\$\$
18
19      -- Trigger: impedisce di eliminare l'ultimo servizio da un
    pacchetto
20      CREATE TRIGGER trg_check_pacchetto_servizi
21      BEFORE DELETE ON composto
22      FOR EACH ROW
23      BEGIN
24      IF NOT EXISTS (
25      SELECT 1 FROM composto
26      WHERE ID_pacchetto = OLD.ID_pacchetto
27      AND ID_servizio <> OLD.ID_servizio
28      ) THEN
29      SIGNAL SQLSTATE '45000'
30      SET MESSAGE\_TEXT = 'Un pacchetto deve avere almeno un
    servizio';
31      END IF;
32      END\$\$
33
34      -- Trigger: che decrementa il numero di posti rimasti dell'
    evento in base al numero di partecipanti iscritti
35      CREATE TRIGGER trg_decrementa_posti_evento
36      BEFORE INSERT ON iscrive
37      FOR EACH ROW
38      BEGIN
39      DECLARE posti_disponibili INT;
40
41      -- Recupero i posti disponibili
42      SELECT posti
43      INTO posti_disponibili
44      FROM EVENTO
```

```
45          WHERE ID_evento = NEW.ID_evento;
46
47          -- Se non ci sono abbastanza posti, blocco l'iscrizione
48          IF posti < NEW.partecipanti THEN
49          SIGNAL SQLSTATE '45000'
50          SET MESSAGE_TEXT = 'Posti insufficienti per questo evento';
51          ELSE
52          -- Decremento i posti disponibili
53          UPDATE EVENTO
54          SET posti = posti - NEW.partecipanti
55          WHERE ID_evento = NEW.ID_evento;
56          END IF;
57          END\$\$
58
59          -- Trigger: che si assicura che un utente abbia usufruito del
    servizio che vuole recensire
60          CREATE TRIGGER trg_recensione_valida
61          BEFORE INSERT ON RECENSIONE
62          FOR EACH ROW
63          BEGIN
64          DECLARE fine_servizio DATE;
65
66          -- Recupero la data_fine dal dettaglio prenotazione
    corrispondente
67          SELECT DP.data_fine
68          INTO fine_servizio
69          FROM DETTAGLI_PRENOTAZIONE DP
70          WHERE DP.ID_prenotazione = NEW.ID_prenotazione
71          AND DP.ID_servizio = (
72          SELECT S.ID_servizio
73          FROM SERVIZIO S
74          WHERE S.tipo_servizio = NEW.tipo_servizio
75          LIMIT 1
76          )
77          LIMIT 1;
78
79          -- Se non trovo alcun dettaglio collegato al servizio
80          IF fine_servizio IS NULL THEN
81          SIGNAL SQLSTATE '45000'
82          SET MESSAGE_TEXT = 'Non esiste un dettaglio prenotazione
    valido per questa recensione.';
83          END IF;
84
85          -- Se il servizio non e' ancora terminato, blocco la
    recensione
86          IF fine_servizio >= CURDATE() THEN
87          SIGNAL SQLSTATE '45000'
88          SET MESSAGE_TEXT = 'La recensione puo' essere inserita solo
    dopo la fine del servizio prenotato.';
89          END IF;
90          END\$\$
91
92          DELIMITER ;
```

Listing 4.1: TRIGGERS

Commento sui trigger I trigger presenti realizzano vincoli di business pi00F9 complessi che non possono essere espressi tramite semplici `CHECK` (ad esempio verifiche che richiedono l'esistenza di righe correlate o l'aggiornamento di attributi aggregati come i posti disponibili). I trigger sono usati per:

- impedire cancellazioni che lascerebbero entit00E0 orfane (es. ultimo turno o ultimo servizio di un pacchetto);
- mantenere coerenza di quantit00E0 (decremento automatico dei posti in `EVENTO`);
- validare l'inserimento di recensioni solo se la prenotazione/servizio risulta concluso.

4.4 Eventi

Nel file SQL è presente un evento che aggiorna periodicamente lo stato dei servizi in base alle prenotazioni. Di seguito la definizione presente nel file:

```
1
2       CREATE EVENT IF NOT EXISTS evt_aggiorna_stato_servizi
3       ON SCHEDULE EVERY 1 HOUR
4       STARTS CURRENT_TIMESTAMP
5       DO
6       BEGIN
7       -- 1. Servizi che devono essere OCCUPATI perche' la
    prenotazione  e' attiva
8       UPDATE SERVIZIO S
9       SET S.status = 'OCCUPATO'
10      WHERE EXISTS (
11      SELECT 1
12      FROM DETTAGLI_PRENOTAZIONE DP
13      WHERE DP.ID_servizio = S.ID_servizio
14      AND DP.data_inizio <= NOW()
15      AND DP.data_fine > NOW()
16      );
17
18      -- 2. Servizi che devono tornare DISPONIBILI perche' la
    prenotazione  e' finita
19      UPDATE SERVIZIO S
20      SET S.status = 'DISPONIBILE'
21      WHERE S.status = 'OCCUPATO'
22      AND NOT EXISTS (
23      SELECT 1
24      FROM DETTAGLI_PRENOTAZIONE DP
25      WHERE DP.ID_servizio = S.ID_servizio
26      AND DP.data_fine > NOW()
27      );
28      END\$\$
```

Listing 4.2: EVENTI

Comportamento L'evento viene schedulato ogni ora e si occupa di sincronizzare l'attributo `status` della tabella `SERVIZIO` con le prenotazioni attive: segna come `OCCUPATO` i servizi con prenotazioni in corso e come `DISPONIBILE` i servizi marcati come occupati ma non piu coperti da prenotazioni.

4.5 Traduzione delle operazioni

In questa sezione vengono riportate le principali operazioni del sistema e la relativa traduzione in query SQL. Per ciascuna operazione viene fornita una breve spiegazione del funzionamento seguita dalla query SQL corrispondente.

1) Registrazione nuovo utente

L'operazione inserisce un nuovo utente nella piattaforma.

```
INSERT INTO UTENTE (username, password, email)
VALUES (:username, :password, :email);
```

2) Autenticazione / login

Quest'operazione verifica che le credenziali inserite corrispondano ad un utente registrato.

```
SELECT *
FROM UTENTE
WHERE username = :username
  AND password = :password;
```

3) Prenotazione servizio

L'operazione crea una prenotazione e i relativi dettagli sui servizi scelti.

```
INSERT INTO PRENOTAZIONE (ID_prenotazione, username)
VALUES (:id_prenotazione, :username);

INSERT INTO DETTAGLIO_PRENOTAZIONE (ID_prenotazione, ID_servizio,
data_inizio, data_fine)
VALUES (:id_prenotazione, :id_servizio, :data_inizio, :data_fine);
```

4) Modifica / cancellazione prenotazione

L'operazione consente di rimuovere una prenotazione esistente e i relativi dettagli.

```
DELETE FROM DETTAGLIO_PRENOTAZIONE
WHERE ID_prenotazione = :id_prenotazione;

DELETE FROM PRENOTAZIONE
WHERE ID_prenotazione = :id_prenotazione;
```

5) Creazione / modifica pacchetto

L'operazione inserisce un nuovo pacchetto e ne definisce i servizi inclusi.

```
INSERT INTO PACCHETTO (ID_pacchetto, nome, descrizione)
VALUES (:id_pacchetto, :nome, :descrizione);

INSERT INTO COMPOSTO (ID_pacchetto, ID_servizio)
VALUES (:id_pacchetto, :id_servizio);
```

6) Acquisto pacchetto

L'operazione registra l'acquisto di un pacchetto da parte di un utente.

```
INSERT INTO ACQUISTA (username, ID_pacchetto)
VALUES (:username, :id_pacchetto);
```

7) Inserimento recensione

Quest'operazione permette ad un utente di lasciare una recensione per un servizio.

```
INSERT INTO RECENSIONE (ID_recensione, username, ID_prenotazione,
tipo_servizio, voto, descrizione)
VALUES (:id_recensione, :username, :id_prenotazione, :tipo_servizio,
:voto, :descrizione);
```

8) Gestione inventario / prodotti

L'operazione aggiorna i dati di un prodotto esistente, come il prezzo o nome.

```
UPDATE PRODOTTO
SET nome = :nome, prezzo = :prezzo
WHERE ID_prodotto = :id_prodotto;
```

9) Creazione evento

L'operazione crea un nuovo evento con nome, data e descrizione.

```
INSERT INTO EVENTO (ID_evento, max_partecipanti, descrizione, username)
VALUES (:id_evento, :max_partecipanti, :descrizione, :username);
```

10) Iscrizione evento

L'operazione registra l'iscrizione di un utente ad un evento.

```
INSERT INTO ISCRIVE (ID_evento, username)
VALUES (:id_evento, :username);
```

11) Creazione ordine

Tale operazione crea un ordine e i relativi dettagli dei prodotti acquistati.

```
INSERT INTO ORDINE (ID_ordine, data, username)
VALUES (:id_ordine, :data, :username);

INSERT INTO DETTAGLIO_ORDINE (ID_ordine, ID_prodotto, quantita,
prezzo_unitario)
VALUES (:id_ordine, :id_prodotto, :quantita, :prezzo_unitario);
```

12) Assegnazione / modifica ruolo dipendenti

Quest'operazione attribuisce un ruolo ad un dipendente.

```
INSERT INTO ASSEGNA_RUOLO (username, ID_ruolo)
VALUES (:username, :id_ruolo);
```

13) Check-in / Check-out

Quest'operazione aggiorna lo stato della prenotazione al momento dell'arrivo o della partenza.

```
-- Query ancora da definire
```

14) Aggiornamento stato servizio

Quest'operazione modifica lo stato di un servizio (es. disponibile, in manutenzione).

```
UPDATE SERVIZIO
SET status = :status
WHERE ID_servizio = :id_servizio;
```

15) Controllo disponibilità servizi

Quest'operazione elenca i servizi attivi e disponibili alla prenotazione.

```
SELECT *
FROM SERVIZIO
WHERE status = 'ATTIVO';
```

16) Generazione report settimanali

Quest'operazione conta le prenotazioni per ciascun servizio in un determinato intervallo.

```
SELECT s.ID_servizio, COUNT(*) AS num_prenotazioni
FROM DETTAGLIO_PRENOTAZIONE dp
JOIN SERVIZIO s ON dp.riguarda = s.ID_servizio
WHERE dp.data_inizio BETWEEN :data_inizio AND :data_fine
GROUP BY s.ID_servizio;
```

17) Applicazione sconto/promozione

L'operazione che applica uno sconto a una prenotazione o ad un ordine.

```
UPDATE PRENOTAZIONE
SET sconto = :sconto
WHERE ID_prenotazione = :id_prenotazione;

UPDATE ORDINE
SET sconto = :sconto
WHERE ID_ordine = :id_ordine;
```

18) Pianificazione turno esistente per dipendente

L'operazione inserisce un turno pianificato per un dipendente.

```
INSERT INTO SVOLGE (username, ID_turno, data_inizio)
VALUES (:username, :id_turno, :data_inizio);
```

19) Calcolo tasso di occupazione camere

L'operazione che calcola la percentuale di camere occupate in un certo periodo.

```
SELECT COUNT(DISTINCT dp.riguarda) * 100.0 / COUNT(*)
AS tasso_occupazione
FROM DETTAGLIO_PRENOTAZIONE dp
JOIN SERVIZIO s ON dp.riguarda = s.ID_servizio
JOIN CAMERA c ON c.cod_camera = s.ID_servizio
WHERE s.tipo = 'CAMERA';
```

20) Analisi fasce orarie più richieste per i lettini

Quest'operazione restituisce le ore più richieste per le prenotazioni dei lettini.

```
SELECT EXTRACT(HOUR FROM dp.data_inizio) AS ora, COUNT(*)
AS num_prenotazioni
FROM DETTAGLIO_PRENOTAZIONE dp
JOIN SERVIZIO s ON dp.riguarda = s.ID_servizio
WHERE s.tipo = 'LETTINO'
GROUP BY ora
ORDER BY num_prenotazioni DESC;
```

21) Individuazione pacchetto più richiesto

Quest'operazione ha lo scopo di individuare il pacchetto acquistato più volte.

```
SELECT p.ID_pacchetto, COUNT(*) AS num_acquisti
FROM PACCHETTO p
JOIN ACQUISTA a ON p.ID_pacchetto = a.ID_pacchetto
GROUP BY p.ID_pacchetto
ORDER BY num_acquisti DESC
LIMIT 1;
```

22) Calcolo fatturato mensile per servizio

L'operazione somma i ricavi mensili per ciascun servizio.

```
-- Query ancora da definire
```

23) Analisi servizi più inclusi nei pacchetti

Quest'operazione conta il numero di volte che ciascun servizio compare nei pacchetti.

```
SELECT s.ID_servizio, COUNT(*) AS frequenza
FROM COMPOSTO_DA c
JOIN SERVIZIO s ON c.ID_servizio = s.ID_servizio
GROUP BY s.ID_servizio
ORDER BY frequenza DESC;
```

Capitolo 5

Progettazione dell'applicazione

L'applicazione è stata sviluppata utilizzando il framework **Django**, che permette di creare applicazioni web in modo rapido e strutturato.

Grazie a Django, è stato possibile gestire facilmente il routing delle pagine, l'interazione con il database e l'autenticazione degli utenti, garantendo al tempo stesso sicurezza e scalabilità.

5.1 Funzionalità di base

L'applicazione sviluppata offre tutte le funzionalità di base necessarie per la gestione di un agriturismo, tra cui la registrazione e autenticazione degli utenti, la prenotazione dei servizi, l'acquisto di pacchetti, la gestione degli ordini e delle recensioni, nonché la partecipazione agli eventi.

Oltre a queste funzionalità, l'applicativo consente anche la gestione completa del database direttamente dalla sua interfaccia: gli utenti con i giusti privilegi possono aggiungere, modificare o eliminare dati relativi a servizi, prodotti, eventi, pacchetti e personale, senza la necessità di accedere manualmente al database.

Questo garantisce un controllo centralizzato, pratico e sicuro su tutte le informazioni del sistema.

5.2 Gestione Utente

Gli utenti possono registrarsi autonomamente tramite l'apposita sezione del sito. Una volta creato l'account, non avranno più accesso soltanto alle funzionalità del sito vetrina (come la consultazione delle informazioni generali, dei servizi e delle offerte), ma potranno interagire pienamente con il sistema.

In particolare, gli utenti autenticati avranno la possibilità di effettuare prenotazioni per camere, tavoli, lettini, attività e pacchetti, acquistare prodotti, iscriversi agli eventi organizzati dall'agriturismo e lasciare recensioni sui servizi di cui hanno usufruito.

Accedendo alla propria area personale, ogni utente potrà visualizzarei propri dati anagrafici, consultare lo storico delle prenotazioni e degli ordini, modificare o cancellare le prenotazioni secondo i termini previsti, e monitorare lo stato delle proprie richieste.

Inoltre, la sezione profilo consente di inserire nuove recensioni, visualizzare quelle già pubblicate e ricevere eventuali comunicazioni dallo staff.

5.3 Gestione Staff

5.4 Gestione Admin

Gli utenti con privilegi di amministratore possono accedere alla dashboard di Django (admin panel) per gestire in modo completo i dati del sistema. Tramite questa interfaccia, l'admin ha la possibilità di aggiungere, modificare ed eliminare le tuple di tutte le tabelle del database, in modo semplice e sicuro, senza dover interagire direttamente con il database. Questo consente una gestione centralizzata e controllata delle informazioni, facilitando le operazioni di amministrazione e manutenzione della piattaforma.

Appendice A

Guida Utente

A.1 Clonazione del repository

Clonare il progetto da GitHub e accedere alla cartella:

```
> git clone https://github.com/alessandrorebosio/D25-farmhouse.git
> cd DB25-farmhouse
```

A.2 Installazione delle dipendenze

Si consiglia di utilizzare un ambiente virtuale Python per isolare le dipendenze del progetto.

```
> python3 -m venv venv
```

Attivazione dell'ambiente virtuale

```
# Su Linux/macOS:
> source venv/bin/activate
# Su Windows:
> venv\Scripts\activate
```

Installazione delle dipendenze dal file requirements.txt

```
> pip install -r requirements.txt
```

A.3 Creazione del database

Per creare il database MySQL a partire dagli script SQL forniti, assicurarsi di avere MySQL installato e in esecuzione.

```
> mysql -u root -p < app/sql/db.sql
> mysql -u root -p < app/sql/data.sql
```

Verrà richiesta la password dell'utente `root`. Il comando eseguirà tutte le istruzioni SQL contenute nel file `db.sql`, creando tabelle, vincoli e dati di esempio necessari per l'applicazione.

A.4 Avvio dell'applicazione

Per avviare l'applicazione Django, assicurarsi che l'ambiente virtuale sia attivo e che il database sia stato creato correttamente.

```
> python manage.py migrate
> python manage.py runserver
```

L'applicazione sarà accessibile all'indirizzo `http://localhost:8000/` tramite browser. Effettuare il login o la registrazione per iniziare a utilizzare il sistema.