Agriturismo

Relazione per il corso di

Basi di Dati

Maisam Noumi
Alessandro Rebosio
Filippo Ricciotti

7 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.

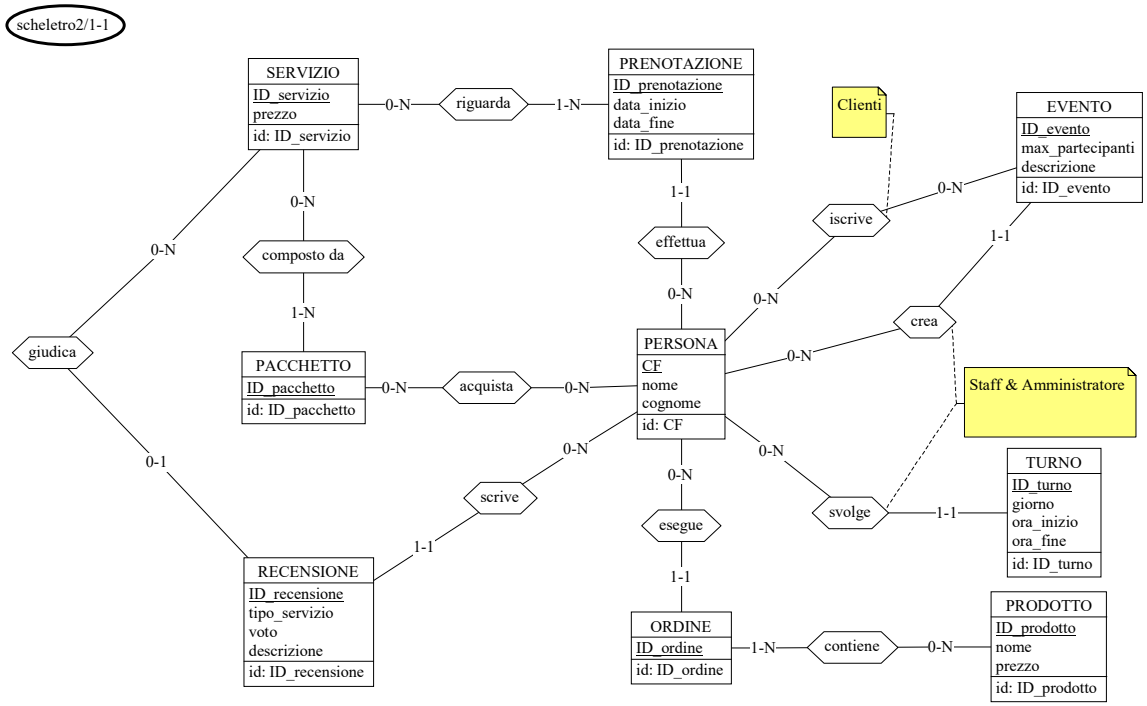

Figura 2.1: Schema ER scheletro

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.

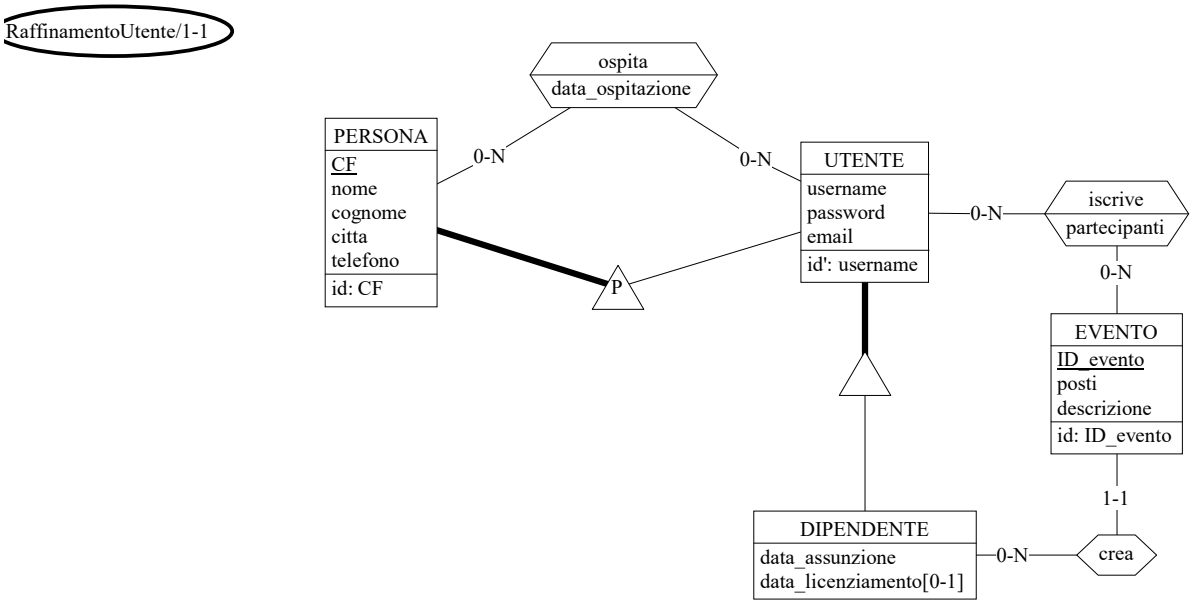

Figura 2.2: Schema ER, raffinamento utente e dipendente

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.

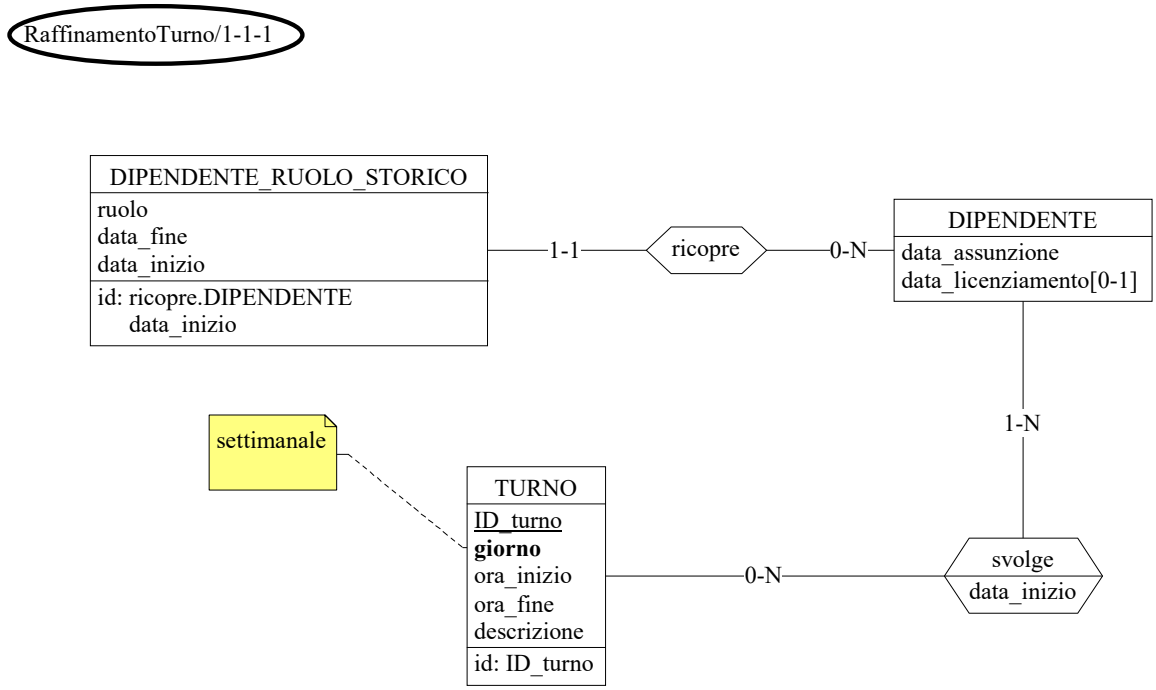

Figura 2.3: Schema ER, raffinamento turno

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.

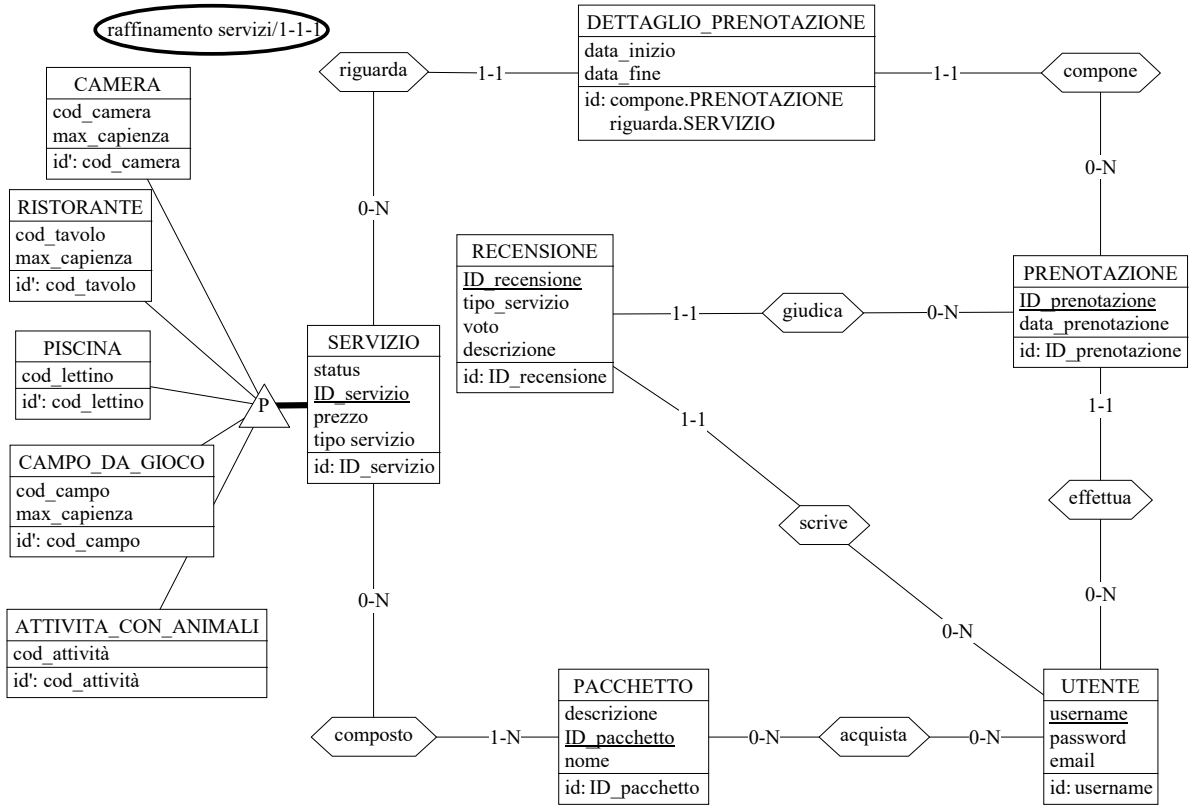

Figura 2.4: Schema ER, raffinamento servizi

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.

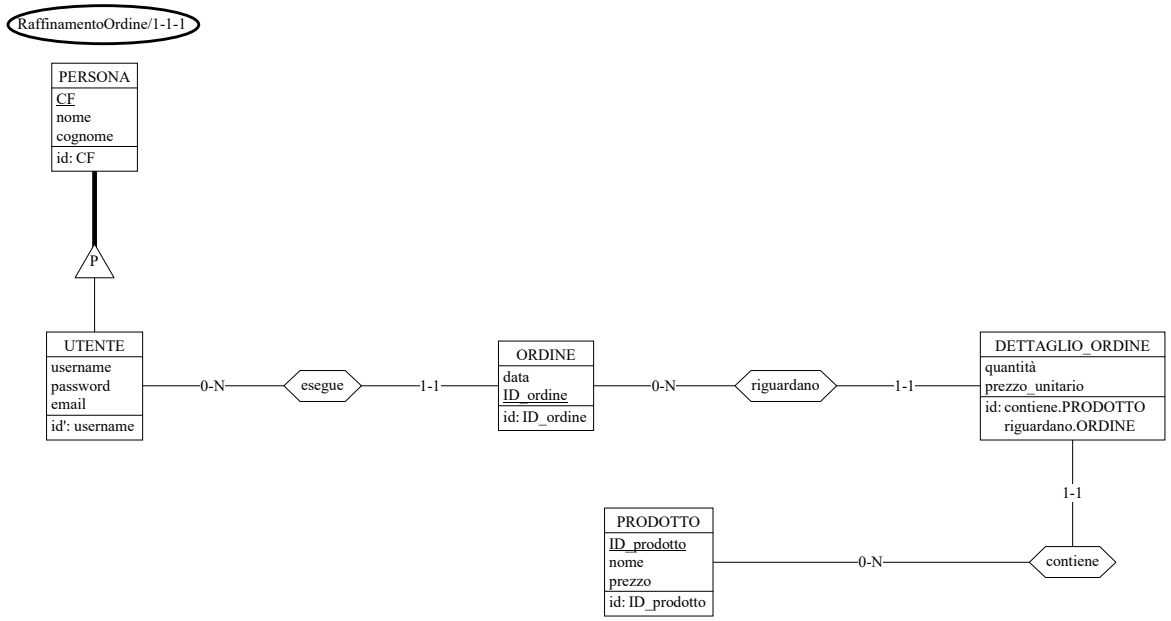

Figura 2.5: Schema ER, raffinamento prodotto

2.3 Schema concettuale finale

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

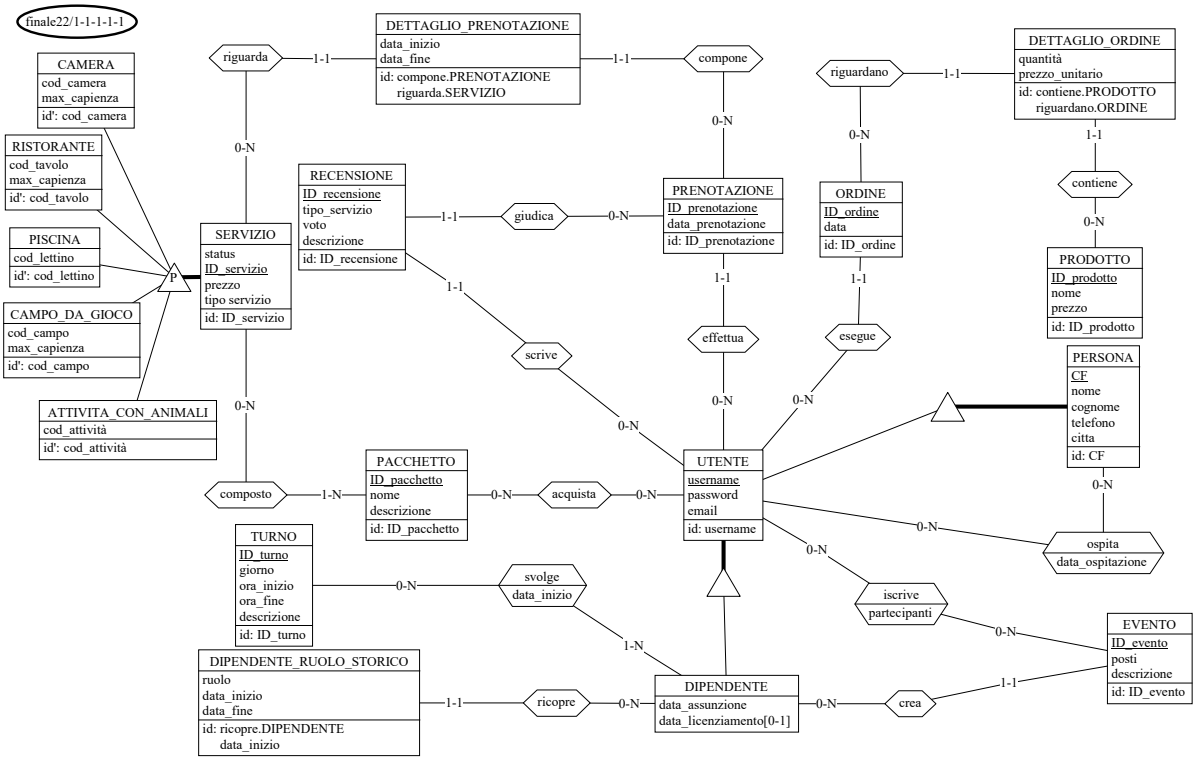

Figura 2.6: Schema ER, schema concettuale finale

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	Prenotazione servizio	125	Cliente / Reception
2	Acquisto pacchetto	15	Cliente
3	Inserimento recensione	40	Cliente
4	Creazione ordine	25	Cliente / Staff
5	Check-in / Check-out (arrivi/partenze)	50	Reception
6	Controllo disponibilità servizi (ricerca/filtri)	550	Tutti
7	Generazione report settimanali (occupazione, ricavi)	1	Admin
8	Applicazione sconto/promozione su prenotazione/ordine	5	Staff / Admin
9	Pianificazione turno esistente per dipendente	3	Admin
10	Calcolo tasso di occupazione camere mensile	1	Admin
11	Analisi fasce orarie più richieste per i lettini	1	Admin / Staff
12	Individuazione pacchetto più richiesto	1	Admin
13	Calcolo fatturato mensile per servizio	1	Admin
14	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. ## 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:

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

2. ## 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:

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

3. 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:

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

4. 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:

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

5. 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:

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

6. 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 $A_{lett} = 3$.

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

7. 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.

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

8. **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 $A_{scr} = 1$.

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

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

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

10. **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:

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

11. 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 $A_{\text{lett}} = 4000 + 45 + 2000 + 2000 = 8045$

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

12. 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 $A_{\text{lett}} = 160 + 30 + 160 + 90 + 200 = 640$

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

13. 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 $A_{\text{lett}} = 5000 + 15000 + 200 + 500 + 5000 = 25700$

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

14. **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 `prezzo_unitario` in `DETTAGLIO_ORDINE`

Operazione	Con ridondanza	Senza ridondanza
Op. 4	300	∼400
Op. 13	25.700	∼30.000
Totale	**26.000**	**∼30.400**

Decisione: Conviene mantenere la ridondanza: il costo di scrittura è minimo e si evitano join pesanti nelle operazioni di reporting.

3.4.2 `quantita` in `DETTAGLIO_ORDINE`

Operazione	Con ridondanza	Senza ridondanza
Op. 4	300	∼500
Op. 13	25.700	∼30.000
Totale	**26.000**	**∼30.500**

Decisione: Mantenere la ridondanza: la quantità è fondamentale per la business logic e la sua presenza semplifica e velocizza tutte le operazioni di lettura e reporting.

3.4.3 `tipo_servizio` in `RECENSIONE`

Operazione	Con ridondanza	Senza ridondanza
Op. 3	160	∼200
Op. 7	30	∼50
Totale	**190**	**∼250**

Decisione: Mantenere la ridondanza per semplificare e velocizzare le query di lettura.

3.4.4 Ridondanza: `status` in `SERVIZIO`

Operazione	Con ridondanza	Senza ridondanza
Op. 6	1.650	∼10.000
Op. 10	8.220	∼15.000
Totale	**9.870**	**∼25.000**

Decisione: Mantenere la ridondanza per garantire performance ottimali nelle ricerche in tempo reale e ridurre la complessità delle query di lettura.

3.4.5 Riepilogo decisioni

In conclusione, tutte le ridondanze individuate (`prezzo_unitario`, `quantita`, `tipo_servizio`, `status`) sono state mantenute perché rendono più semplici e veloci le operazioni di lettura e reporting, migliorando le prestazioni del sistema.

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	Prenotazione servizio	1125	Cliente / Reception
2	Acquisto pacchetto	195	Cliente
3	Inserimento recensione	160	Cliente
4	Creazione ordine	300	Cliente / Staff
5	Check-in / Check-out (arrivi/partenze)	350	Reception
6	Controllo disponibilità servizi (ricerca/filtri)	1650	Tutti
7	Generazione report settimanali (occupazione, ricavi)	30	Admin
8	Applicazione sconto/promozione su prenotazione/ordine	15	Staff / Admin
9	Pianificazione turno esistente per dipendente	12	Admin
10	Calcolo tasso di occupazione camere mensile	8220	Admin
11	Analisi fasce orarie più richieste per i lettini	8045	Admin / Staff
12	Individuazione pacchetto più richiesto	640	Admin
13	Calcolo fatturato mensile per servizio	25700	Admin
14	Analisi servizi più inclusi nei pacchetti	640	Admin

Tabella 3.3: Costo 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)

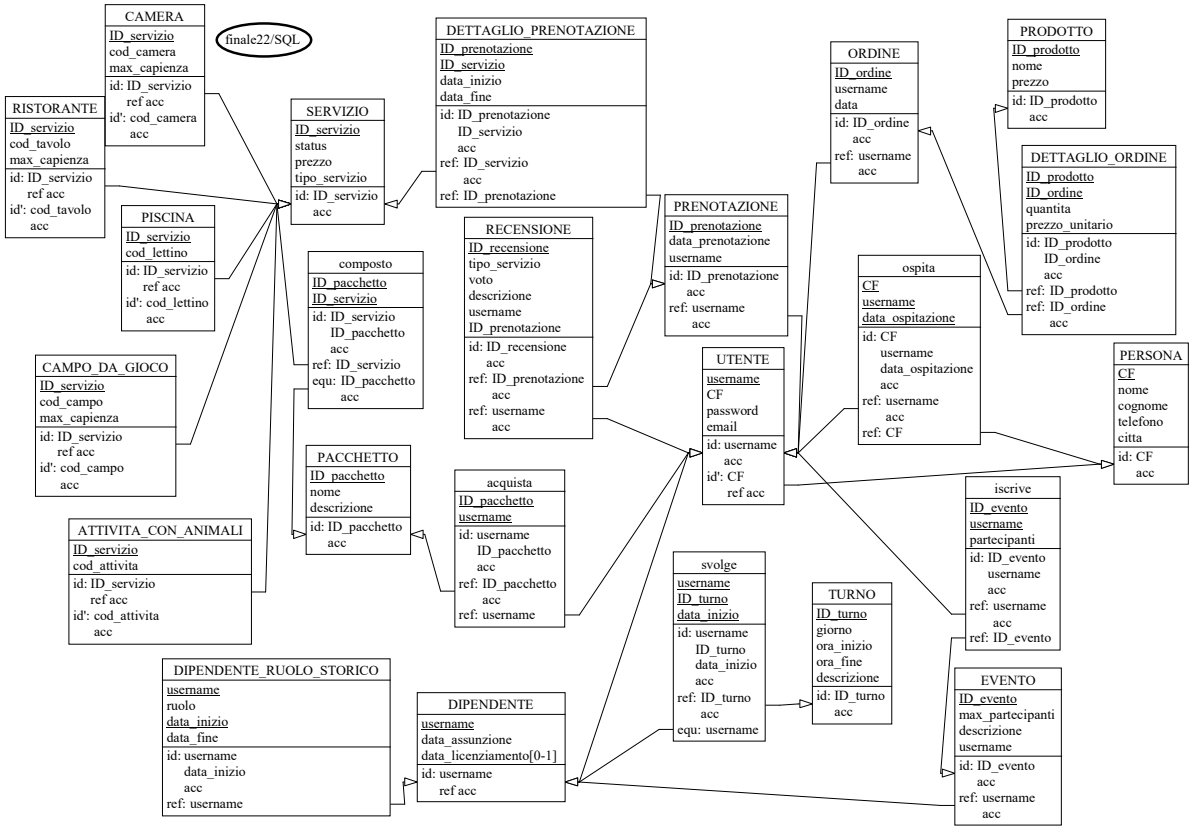

Figura 3.1: Schema logico, schema relazionale finale

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 Dati

Una volta creato il nostro database, riportiamo di seguito una parte del codice relazionale utilizzato per la sua implementazione.

4.1 Check

Sono stati utilizzati vincoli di tipo `CHECK` per definire alcuni domini e assicurare semplici proprietà degli attributi. Di seguito un esempio di vincolo `CHECK` utilizzato per assicurare che il prezzo di ogni prodotto sia maggiore o uguale a zero:

```
CREATE TABLE PRODOTTO (
        ID_prodotto INT AUTO_INCREMENT NOT NULL,
        nome VARCHAR(100) NOT NULL,
        prezzo DECIMAL(8,2) NOT NULL CHECK (prezzo >= 0),
        CONSTRAINT ID_PRODOTTO_ID PRIMARY KEY (ID_prodotto)
);
```

4.2 Viste

Poiché la visualizzazione dei servizi disponibili è centrale nell'applicazione, è stata creata la vista `V_SERVIZI_DISPONIBILI`, che mostra rapidamente tutti i servizi attualmente prenotabili, con le informazioni essenziali già aggregate.

```
CREATE VIEW V_SERVIZI_DISPONIBILI AS
SELECT
S.ID_servizio,
S.tipo_servizio,
S.prezzo,
S.status,
COALESCE(R.cod_tavolo, P.cod_lettino, CG.cod_campo, CA.cod_camera, A.
    cod_attivita) AS codice,
COALESCE(R.max_capienza, CG.max_capienza, CA.max_capienza) AS
    max_capienza,
A.descrizione AS dettaglio_descrizione
FROM SERVIZIO S
LEFT JOIN RISTORANTE R ON R.ID_servizio = S.ID_servizio
LEFT JOIN PISCINA P ON P.ID_servizio = S.ID_servizio
LEFT JOIN CAMPO_DA_GIOCO CG ON CG.ID_servizio = S.ID_servizio
LEFT JOIN CAMERA CA ON CA.ID_servizio = S.ID_servizio
LEFT JOIN ATTIVITA_CON_ANIMALI A ON A.ID_servizio = S.ID_servizio
```

```
WHERE S.status = 'DISPONIBILE';
```

4.3 Trigger

Nel database sono stati implementati trigger per garantire vincoli di business complessi, come il controllo dei posti disponibili negli eventi. Di seguito è riportato un esempio di trigger che impedisce l'iscrizione a un evento se non ci sono abbastanza posti e aggiorna automaticamente il numero di posti disponibili:

```
CREATE TRIGGER trg_decrementa_posti_evento
BEFORE INSERT ON iscrive
FOR EACH ROW
BEGIN
DECLARE posti_disponibili INT;

-- Recupero i posti disponibili
SELECT posti
INTO posti_disponibili
FROM EVENTO
WHERE ID_evento = NEW.ID_evento;

-- Se non ci sono abbastanza posti, blocco l'iscrizione
IF posti_disponibili < NEW.partecipanti THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Posti insufficienti per questo evento';
ELSE
-- Decremento i posti disponibili
UPDATE EVENTO
SET posti = posti_disponibili - NEW.partecipanti
WHERE ID_evento = NEW.ID_evento;
END IF;
```

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:

```
        CREATE EVENT IF NOT EXISTS evt_aggiorna_stato_servizi
ON SCHEDULE EVERY 1 HOUR
STARTS CURRENT_TIMESTAMP
DO
BEGIN
UPDATE SERVIZIO S
SET S.status = 'OCCUPATO'
WHERE EXISTS (
        SELECT 1
        FROM DETTAGLIO_PRENOTAZIONE DP
        WHERE DP.ID_servizio = S.ID_servizio
        AND DP.data_inizio <= NOW()
        AND DP.data_fine > NOW()
        );

        UPDATE SERVIZIO S
        SET S.status = 'DISPONIBILE'
```

```
        WHERE S.status = 'OCCUPATO'
        AND NOT EXISTS (
        SELECT 1
        FROM DETTAGLIO_PRENOTAZIONE DP
        WHERE DP.ID_servizio = S.ID_servizio
        AND DP.data_fine > NOW()
);
```

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 più 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. **Prenotazione servizio**
   Crea una nuova prenotazione per un utente e aggiunge i dettagli dei servizi prenotati (ad esempio camere, tavoli, ecc.).

```
INSERT INTO PRENOTAZIONE (ID_prenotazione, username)
VALUES (?, ?);

INSERT INTO DETTAGLIO_PRENOTAZIONE (ID_prenotazione, ID_servizio,
    data_inizio, data_fine)
VALUES (?, ?, ?, ?);
```

2. **Acquisto pacchetto**
   Registra l'acquisto di un pacchetto da parte di un utente.

```
INSERT INTO ACQUISTA (username, ID_pacchetto)
VALUES (?, ?);
```

3. **Inserimento recensione**
   Permette a un utente di lasciare una recensione su un servizio utilizzato.

```
INSERT INTO RECENSIONE (ID_recensione, username, ID_prenotazione,
    tipo_servizio, voto, descrizione)
VALUES (?, ?, ?, ?, ?, ?);
```

4. **Creazione ordine**
   Crea un nuovo ordine e aggiunge i dettagli dei prodotti acquistati.

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

INSERT INTO DETTAGLIO_ORDINE (ID_ordine, ID_prodotto, quantita,
    prezzo_unitario)
VALUES (?, ?, ?, ?);
```

5. **Check-in / Check-out**
   Aggiorna lo stato di check-in o check-out nei dettagli della prenotazione.

```sql
UPDATE DETTAGLIO_PRENOTAZIONE
SET stato = 'CHECKIN'
WHERE ID_prenotazione = ? AND ID_servizio = ?;

UPDATE DETTAGLIO_PRENOTAZIONE
SET stato = 'CHECKOUT'
WHERE ID_prenotazione = ? AND ID_servizio = ?;
```

6. **Servizi attivi**
   Restituisce tutti i servizi attualmente attivi e disponibili per la prenotazione.

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

7. **Generazione report settimanali**
   Conta il numero di prenotazioni per ciascun servizio in un intervallo di date specificato.

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

8. **Applicazione sconto/promozione**
   Applica uno sconto a una prenotazione o a un ordine.

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

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

9. **Pianificazione turno esistente per dipendente**
   Associa un dipendente a un turno pianificato.

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

10. **Calcolo tasso di occupazione camere**
    Calcola la percentuale di camere occupate in un certo periodo.

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

11. Analisi fasce orarie più richieste per i lettini
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.ID_servizio = s.ID_servizio
WHERE s.tipo_servizio = 'LETTINO'
GROUP BY ora
ORDER BY num_prenotazioni DESC;
```

12. Individuazione pacchetto più richiesto
Individua il pacchetto acquistato più volte dagli utenti.

```
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;
```

13. Calcolo fatturato mensile per servizio
Calcola il fatturato mensile per ciascun servizio sulla base degli ordini effettuati.

```
SELECT s.ID_servizio, SUM(dp.prezzo_unitario * dp.quantita) AS
   fatturato_mensile
FROM DETTAGLIO_ORDINE dp
JOIN ORDINE o ON dp.ID_ordine = o.ID_ordine
JOIN SERVIZIO s ON dp.ID_servizio = s.ID_servizio
WHERE MONTH(o.data) = :mese AND YEAR(o.data) = :anno
GROUP BY s.ID_servizio;
```

14. Analisi servizi più inclusi nei pacchetti
Conta quante volte ciascun servizio è incluso nei pacchetti disponibili.

```
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.