Elaborato per il corso di Basi di dati

Davide Carità - 0000873616
davide.carita@studio.unibo.it

Pietro Olivi - 0001020332
pietro.olivi2@studio.unibo.it

Lorenzo Dalmonte - 0001021552
lorenzo.dalmonte4@studio.unibo.it

Contents

Chapter 1

Analisi dei requisiti

1.1 Requisiti in linguaggio naturale

Si vuole realizzare una base di dati che supporti le funzionalità di un' applicazione di compravendita di immobili nonché il monitoraggio del welfare delle principali città europee. All'interno dell'applicazione, si potrà creare un account che permetta agli utenti di promuovere e vendere le loro proprietà attraverso la piattaforma. Allo stesso tempo, gli utenti potranno utilizzare l'applicazione per cercare e visualizzare annunci immobiliari disponibili.

Uno dei servizi sarà quello di individuare la città europea che meglio si adatta alle proprie esigenze: si potrà vedere ad esempio quale città ha ottenuto i migliori punteggi a livello europeo in tema di qualità dell'aria, costo della vita o efficienza sanitaria. Oltre a ciò, saranno disponibili anche i dati relativi agli anni precedenti, in modo da avere uno storico che permetta di valutare lo sviluppo del welfare nella città selezionata. Una volta che gli utenti hanno individuato la propria città ideale, il servizio offrirà la possibilità di visualizzare annunci immobiliari specifici suddivisi per diverse zone all'interno della città. Questa suddivisione permette agli utenti di esplorare le opzioni abitative in aree specifiche che potrebbero essere più in linea con le loro preferenze e esigenze.

Gli annunci immobiliari saranno suddivisi in tre tipi: vendita, affitto e asta, e forniranno dettagli completi sugli immobili, compresi la metratura, il numero di stanze, il prezzo e altre informazioni pertinenti. Oltre a ciò, il servizio fornirà la funzionalità di avviare una conversazione tra acquirente e venditore, offrendo loro la possibilità di scambiarsi informazioni in modo rapido. Sarà data la possibilità ai giudici d'esecuzione di creare un account privile-

giato che consentirà loro di gestire delle aste immobiliari. Questo account fornirà ai giudici strumenti e funzionalità speciali per gestire l'intero processo di vendita all'asta, inclusa la creazione e la pubblicazione degli annunci, la gestione delle offerte e altre attività correlate.

1.2 Estrazione dei concetti principali

Città → Le città sono suddivise in diverse zone e vengono valutate in base a parametri di welfare. Gli utenti utilizzano l'applicazione per individuare la città che meglio si adatta alle loro esigenze e preferenze abitative.

Categoria → Le categorie di welfare sono criteri utilizzati per valutare il benessere nelle città europee. Essi includono ambiente, trasporto, economia, sanità ed istruzione. Questi parametri forniscono un quadro del livello di comfort e salute delle città. Consentono agli utenti di confrontare le città in base a questi indicatori, facilitando la scelta di una città che meglio si adatta alle proprie esigenze.

- **Ambiente** Valuta la qualità ambientale delle città, con indicatori come il pm 2.5 (particolato fine) e la percentuale di spazio verde urbano.
- **Trasporto** Valuta l'efficienza dei sistemi di trasporto urbano, considerando il tempo medio di pendolarismo e il numero di auto per persona.
- **Economia** Valuta la vitalità economica di una città attraverso indicatori come il PIL pro capite, lo stipendio medio e il tasso di disoccupazione. Consente inoltre di avere un'indicazione sul livello di prosperità e opportunità lavorative presenti nell'area urbana considerata.
- **Sanità** Valuta la qualità del sistema sanitario di una città, considerando l'aspettativa di vita e il numero di posti letto, all'interno degli ospedali, per abitante. Fornisce un'indicazione sulla salute generale e sull'accessibilità alle cure mediche
- **Istruzione** Valuta la qualità dell'istruzione in una città, considerando la percentuale di laureati, la percentuale di diplomati e il numero di università presenti oltre che dare un'indicazione sulla disponibilità e l'accessibilità all'istruzione superiore e alla formazione professionale nell'area urbana considerata.

Immobile → Un immobile si riferisce a una proprietà che può essere venduta, affittata o messa all'asta dagli utenti attraverso la piattaforma. Gli annunci immobiliari includono informazioni come la metratura, il numero di stanze, il prezzo e altre caratteristiche rilevanti dell'immobile.

Annuncio → Una presentazione di un immobile disponibile per la vendita, l'affitto o l'asta. Gli annunci forniscono dettagli completi sull'immobile, come la sua ubicazione, le specifiche, i prezzi e altre informazioni pertinenti. Gli utenti utilizzano l'applicazione per cercare, visualizzare e interagire con gli annunci immobiliari.

Account → Gli utilizzatori possono creare un account utente per promuovere e vendere le loro proprietà, cercare annunci immobiliari disponibili, avviare conversazioni con acquirenti o venditori e gestire altre attività correlate. Ai giudici d'esecuzione viene fornita la possibilità di creare un account privilegiato con strumenti e funzionalità speciali per gestire le aste immobiliari e le attività associate al processo di vendita all'asta.

Di seguito le principali azioni richieste:

1. Creare un account utente.
2. Pubblicare un annuncio immobiliare di vendita/affitto/asta.
3. Visualizzare annunci immobiliari disponibili.
4. Mostrare dettagli completi di un immobile.
5. Avviare una conversazione con un venditore.
6. Visualizzare punteggi di welfare di una città.
7. Filtrare annunci immobiliari per criteri (es. prezzo, dimensioni).

Chapter 2

Progettazione concettuale

La progettazione concettuale, derivata dall'analisi dei concetti principali del dominio, è stata incrementale in termini di complessità. Di seguito passeremo quindi in rassegna gli stadi dello schema cronologicamente ordinati, illustrando le motivazioni che hanno portato ad effettuare i vari raffinamenti.

2.1 Schema scheletro

Il primo schema è il risultato della trasposizione su E/R delle astrazioni formulate nel capitolo precedente. In primo piano figura l'entita *città*, caratterizzata da al più N *valutazioni* generiche, ciascuna delle quali espressa con un punteggio intero nell'intervallo [0,10].

La città è identificata da un codice univoco, in modo da evitare di incappare in uno dei ricorrenti scenari di città omonime nella medesima nazione (dove lo stato ed il nome della città non sarebbero sufficienti a distinguerle). Sebbene in molti casi ciò possa essere risolto specificando la regione di appartenenza delle città "doppione", rimarrebbero comunque insolute casistiche tipiche del territorio britannico, dove città omonime potrebbero situarsi in contee diverse ma all'interno della stessa "Government Office *Region*". A titolo di esempio, basti pensare che nell'intero UK compaiono 14 città chiamate Newport! [1]

Un annuncio riguarda un solo immobile, mentre lo stesso immobile può comparire in più annunci diversi. Quest'ultima cardinalità, meno ovvia, deriva dall'esigenza di conservare lo storico di tutte le vendite/affitti di ciascuna

[1] https://www.southwalesargus.co.uk/news/19433701.newports-across-world/

abitazione, in modo da formulare in seguito un trend del prezzo in funzione del tempo (da presentare agli interessati).

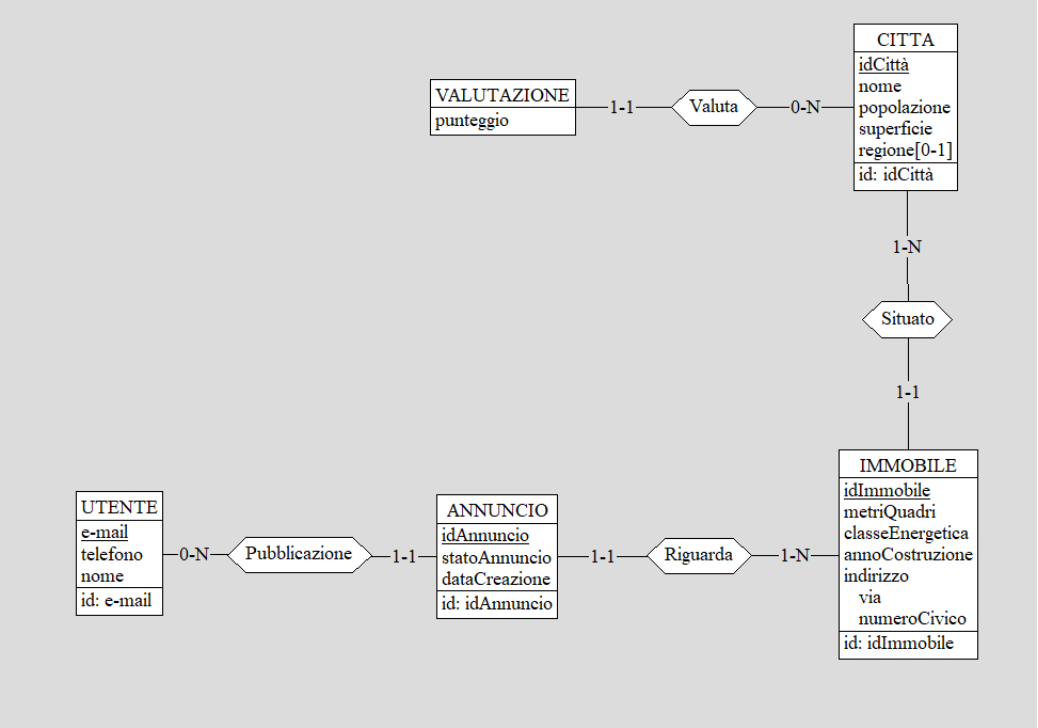

Figure 2.1: La prima versione dello schema concettuale

2.2 Divisione in zone e di immobili

Introducendo una gerarchia per differenziare le varie tipologie di *immobili* inserite negli annunci, ogni immobile dovrà essere necessariamente contraddistinto da un ID proprio. Perchè? Se tutte le dimore fossero case indipendenti, allora potrebbero essere identificate da attributi che ne specifichino la posizione. Aggiungendo gli appartamenti poi, basterebbe includere nella chiave un campo che indichi il numero dell'interno. Con l'avvento delle stanze singole tuttavia, la soluzione precedentemente adottata risulterà inadatta, poichè non risponderebbe alla necessità di un utente di affittare più stanze della stessa abitazione (ognuna con un annuncio dedicato).

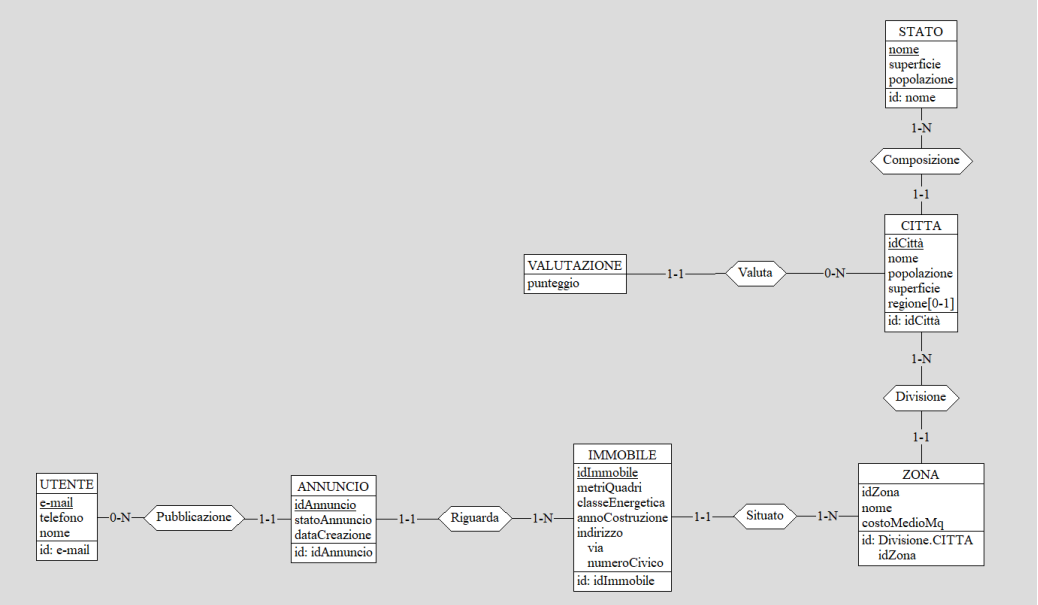

Figure 2.2: La seconda versione dello schema concettuale

2.3 Espansione delle valutazioni

Nell'amplificazione della porzione di schema relativa all'analisi delle città abbiamo scisso la precedente entità "segnaposto" *valutazione* in una serie di categorie, come illustrato nei requisiti. Ciascuna di queste è caratterizzata da parametri dal dominio numerico (e.g. *ambiente* da *PM2.5media* e *percentualeSpazioVerdeUrbano*), in base ai cui valori verrà calcolato in automatico un punteggio onnicomprensivo per la categoria, poi conservato nel campo corrispondente di *città_anno* (*punteggioAmbiente* nell'esempio citato).

Il ruolo dell'entità *città_anno* è quello di consentire la storicizzazione dei punteggi ottenuti da ciascuna città negli anni passati, in modo da poter computare lo sviluppo, o l'involuzione, a cui il luogo ha assistito. Ciascuna istanza di categoria può far riferimento a più città ed in più anni diversi: Parigi e Londra potrebbero aver registrato stessi *PILProCapite, stipendioMedio e tassoDisoccupazione* nel 2019, così come Heidelberg potrebbe aver riconfermato gli stessi valori relativi alla *sanità* dell'anno precedente. Ad ogni *città_anno*, invece, è collegata una ed una sola istanza di tutte le 5 categorie.

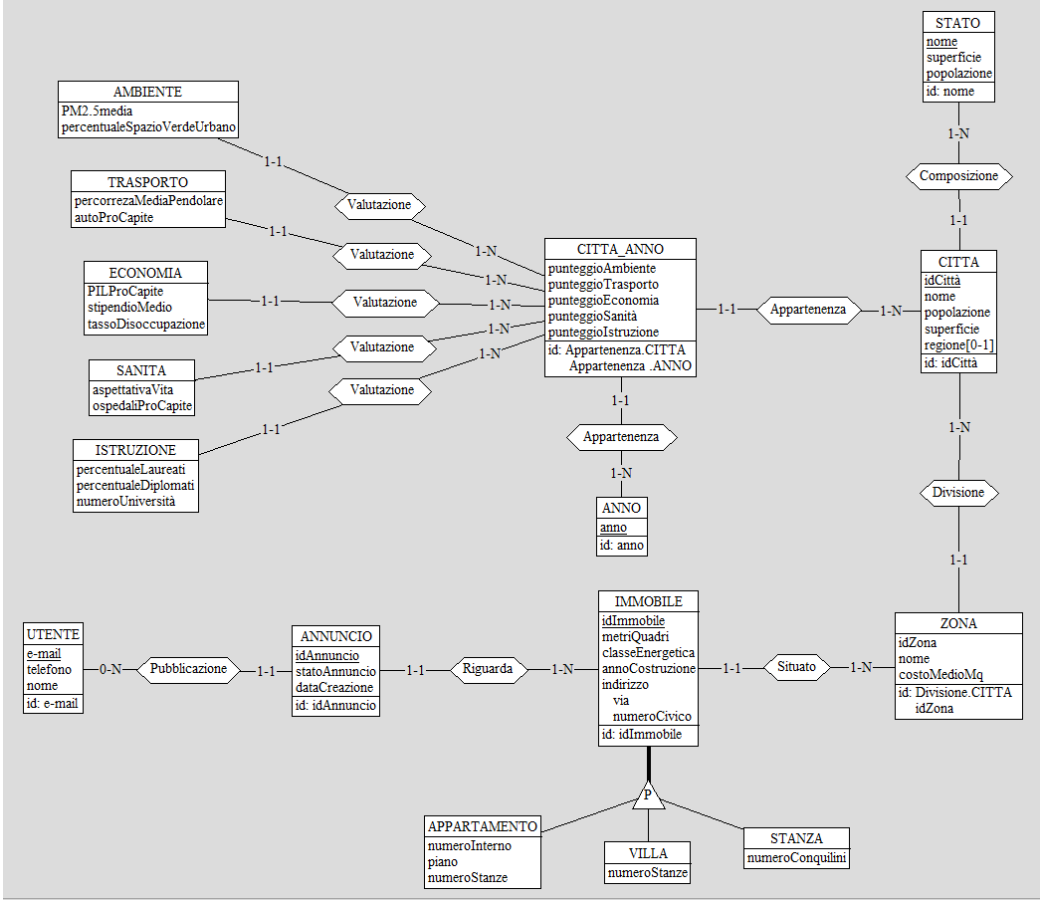

Figure 2.3: La terza versione dello schema concettuale

2.4 Aste e giudici

Per enfatizzare la differente natura degli annunci pubblicabili dagli utenti ordinari e le aste riservate ad i giudici, si è deciso di scindere l'entità *annuncio* rispettivamente in *annuncio_utente* ed *asta*.

Le aste sono tipologie di annunci la cui gestione (dunque pubblicazione) è riservata ad i giudici di esecuzione. In particolare, ciascuna asta verrà amministrata da uno ed un solo giudice, il quale però potrà dirigerne fino ad N. Per riprodurre le dinamiche di ascesa del prezzo ed infine di vendita, viene istituita l'entità rialzo.

Durante il periodo di attività di un'asta, scandito da una data di inizio ed una di fine, gli utenti hanno la facoltà di avanzare più rialzi (Il collegare

direttamente utente ad asta avrebbe ristretto il numero massimo di rialzi effettuabili da un utente, nella stessa asta, ad 1). I rialzi ricevuti in una determinata asta sono identificati univocamente dall' utente che li ha effettuati e l'istante temporale in cui sono stati offerti. Al termine di un asta, si può risalire al prezzo di vendita, ove avvenuta, verificando il rialzo più recente ad essa collegato.

2.5 Introduzione della messaggistica

La messaggistica viene modellata attraverso le associazioni tra le entità *utente* $\Leftrightarrow$ *messaggio* e *messaggio* $\Leftrightarrow$ *annuncio_utente*. In prima battuta ci si potrebbe erroneamente domandare il motivo del non aver optato per una relazione ad anello tra *utenti*, con il testo del messaggio racchiuso in un campo dell' associazione, tuttavia tale configurazione consentirebbe di memorizzare nella base di dati al più un *messaggio* per ogni coppia di utenti!

Sarebbe inoltre fuorviante collegare ogni messaggio direttamente a mittente e destinatario poiché non si terrebbe conto del caso in cui le stesse due persone dovessero contattarsi in merito ad annunci diversi, anche contemporaneamente, e non si riuscirebbe dunque a dedurre il contesto dei messaggi. Per risolvere quest'ultima problematica si è deciso di interporre l'enitità *messaggio* tra *utente*, nonchè il potenziale acquirente, ed *annuncio_utente*, ossia il gestore dell'inserzione nelle vesti del suo annuncio. I messaggi all'interno di ogni chat vengono identificati dall'istante temporale in cui sono inviati (timestamp).

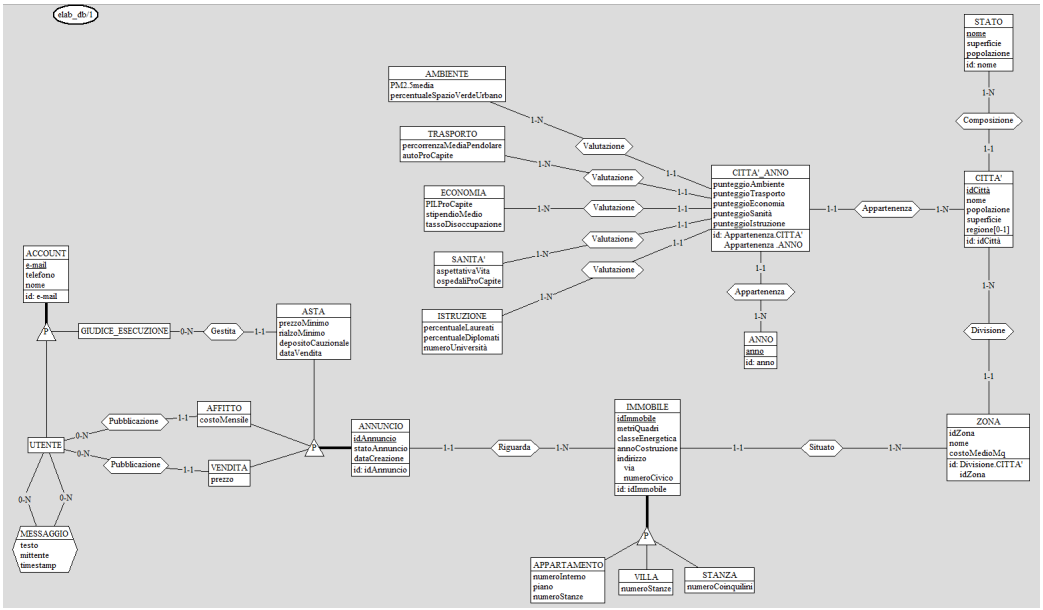

Figure 2.4: L'ultima versione dello schema concettuale

Chapter 3

Progettazione logica

3.1 Stima del volume dei dati

Concetto	Costrutto	Volume
Stato	E	50
Composizione	R	700
Città	E	700
Divisione	R	7.000
Zona	E	7.000
Città_Anno	E	14.000
Appartenenza	R	14.000
Riferimento	R	14.000
Anno	E	20
Ambiente	E	12.000
Valutazione_A	R	12.000
Trasporto	E	12.000
Valutazione_T	R	12.000
Economia	E	12.000
Valutazione_E	R	12.000
Sanità	E	12.000
Valutazione_S	R	12.000
Istruzione	E	12.000
Valutazione_I	R	12.000

Concetto	Costrutto	Volume
Situato	R	350.000
Immobile	E	350.000
Appartamento	E	116.700
Villa	E	116.700
Stanza	E	116.700
Riguarda	R	450.000
Annuncio	E	450.000
Asta	E	50.000
Annuncio_Utente	E	400.000
Affitto	E	200.000
Vendita	E	200.000
Invio/Ricezione	R	15.000.000
Ricezione/Invio	R	15.000.000
Messaggio	E	15.000.000
Pubblicazione	R	400.000
Account	E	1.505.000
Utente	E	1.500.000
Offerta	R	750.000
Rialzo	E	750.000
Accoglimento	R	750.000
Gestione	R	50.000
Giudice_Esecuzione	E	5.000

3.2 Descrizione delle operazioni principali e stima della loro frequenza

Codice	Operazione	Frequenza
1	Registrare un nuovo account	1.500 al giorno
2	Pubblicare un annuncio immobiliare (vendita, affitto o asta)	500 al giorno
3	Visualizzare annunci immobiliari attivi di una città in ordine cronologico	50.000 al giorno
4	Suddividere gli annunci in vendita, affitto o asta all'interno di una città	35.000 al giorno
5	Ordinare gli immobili in base a particolari filtri (e.g. metratura, classe energetica)	5.000 al giorno
6	Filtrare gli annunci per zona	40.000 al giorno
7	Ordinare gli annunci di affitto per costo mensile	12.000 al giorno
8	Mostrare aste attive in una città	7.500 al giorno
9	Ordinare le aste per prezzo attuale crescente	4.000 al giorno
10	Effettuare un rialzo all'interno di un'asta (controllando la sua validità)	500 al giorno
11	Contattare un utente in merito ad un annuncio creato	15.000 al giorno
12	Ricostruire una conversazione tra due utenti	30.000 al giorno
13	Andamento del prezzo di un immobile in funzione del tempo	5.000 al giorno
14	Comparare il prezzo di un immobile al mq con quello degli immobili nella stessa zona	10.000 al giorno
15	Comparare il prezzo di un immobile al mq con quello degli immobili nella stessa città	7.500 al giorno
16	Comparare il prezzo medio al mq di una zona con quello della città	3.500 al giorno
17	Ordinare le zone di una città per costo medio al mq	1.500 al giorno
18	Stilare una top 5 città per una o più categorie	3.000 al giorno
19	Classificare città in base all'evoluzione in una categoria rispetto all'anno precedente	500 al giorno
20	Ordinare le città in base a valori specifici di una categoria	1.000 al giorno
21	Calcolare performance di uno stato in una categoria (da ogni città)	2.000 al giorno
22	Aggiornamento annuale dei punteggi di una città (dati i campi delle categorie)	1 all'anno

3.3 Schemi di navigazione e tabelle degli accessi

OP 1: Registrare un nuovo account

Concetto	Costrutto	Accessi	Tipo
Account	E	1	Scrittura
		Totale: 1S $\rightarrow$ 3.000 al giorno	

OP 2: Pubblicare un annuncio immobiliare (vendita, affitto o asta) Viene di seguito presentato il caso in cui si voglia aggiungere un annuncio relativo ad un immobile non ancora presente nella base di dati: sarà

necessario aggiornare, dunque riscrivere, i campi *numeroImmobili* e *costoMedioMq*.

Concetto	Costrutto	Accessi	Tipo
Annuncio	E	1	Scrittura
Riguarda	R	1	Scrittura
Immobile	E	1	Scrittura
Situato	R	1	Scrittura
Zona	E	1	Scrittura
		Totale: 5S → 5.000 al giorno	

Nell'eventualità in cui venisse pubblicato un'annuncio relativo ad un immobile già comparso sulla piattaforma, e pertanto già considerato nel calcolo del *costoMedioMq* del quartiere di appartenenza, dovremmo leggere il prezzo al mq designato dal proprietario più recente rispetto all'attuale, sottrarlo a costoMedioMq*numeroImmobili, aggiungere il nuovo e ri-dividere per lo stesso numeroImmobili.

Concetto	Costrutto	Accessi	Tipo
Annuncio	E	1	Lettura
Annuncio	E	1	Scrittura
Riguarda	R	1	Scrittura
Immobile	E	1	Lettura
Situato	R	1	Lettura
Zona	E	1	Scrittura
		Totale: 3L + 3S → 1.800 al giorno	

OP 3: Visualizzare annunci immobiliari attivi di una città in ordine cronologico
Data una città, prendiamo le sue 10 zone, che hanno in media 5 immobili l'una. Visto che gli annunci totali *attivi* (worst case 450.000) sono ≥ degli immobili (350.000) (considerando che lo stesso immobile può essere pubblicato sia come affitto che vendita) uso lo stesso loro rapporto (1,3) per calcolare le letture che farò negli annunci.

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
Divisione	R	10	Lettura
Zona	E	10	Lettura
Situato	R	50	Lettura
Immobile	E	50	Lettura
Riguarda	R	65	Lettura
Annuncio	E	65	Lettura
		Totale: 251L → 12.550.000 al giorno	

OP 4: Suddividere gli annunci in vendita, affitto o asta all'interno di una città

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
Divisione	R	10	Lettura
Zona	E	10	Lettura
Situato	R	50	Lettura
Immobile	E	50	Lettura
Riguarda	R	65	Lettura
Annuncio	E	65	Lettura
		Totale: 251L → 8.785.000 al giorno	

OP 5: Ordinare gli annunci in base a particolari filtri sull'immobile (e.g. metratura, classe energetica)

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
Divisione	R	10	Lettura
Zona	E	10	Lettura
Situato	R	50	Lettura
Immobile	E	50	Lettura
Riguarda	R	65	Lettura
Annuncio	E	65	Lettura
		Totale: 251L → 1.255.000 al giorno	

OP 6: Filtrare gli annunci per zona

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
Divisione	R	10	Lettura
Zona	E	10	Lettura
Situato	R	50	Lettura
Immobile	E	50	Lettura
Riguarda	R	65	Lettura
Annuncio	E	65	Lettura
		Totale: 251L → 10.040.000 al giorno	

OP 7: Ordinare gli annunci di affitto per costo mensile

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
Divisione	R	10	Lettura
Zona	E	10	Lettura
Situato	R	50	Lettura
Immobile	E	50	Lettura
Riguarda	R	65	Lettura
Annuncio	E	65	Lettura
		Totale: 251L → 3.012.000 al giorno	

OP 8: Mostrare aste attive in una città

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
Divisione	R	10	Lettura
Zona	E	10	Lettura
Situato	R	50	Lettura
Immobile	E	50	Lettura
Riguarda	R	65	Lettura
Annuncio	E	65	Lettura
		Totale: 251L → 1.882.500 al giorno	

OP 9: Ordinare le aste per prezzo attuale crescente

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
Divisione	R	10	Lettura
Zona	E	10	Lettura
Situato	R	50	Lettura
Immobile	E	50	Lettura
Riguarda	R	65	Lettura
Annuncio	E	65	Lettura
		Totale: 251L → 1.004.000 al giorno	

OP 10: Effettuare un rialzo all'interno di un'asta (controllando la sua validità)

Concetto	Costrutto	Accessi	Tipo
Utente	E	1	Lettura
Offerta	R	1	Lettura
Rialzo	E	1	Lettura
Riceve	R	1	Lettura
Asta	E	1	Lettura
Offerta	R	1	Scrittura
Rialzo	E	1	Scrittura
Riceve	R	1	Scrittura
		Totale: 5L + 3S → 5.500 al giorno	

OP 11: Contattare un utente in merito ad un annuncio creato

Concetto	Costrutto	Accessi	Tipo
Utente	E	1	Lettura
Annuncio_Utente	E	1	Lettura
Invio/Ricezione	R	1	Scrittura
Messaggio	E	1	Scrittura
Invio/Ricezione	R	1	Scrittura
		Totale: 2L + 3S → 120.000 al giorno	

OP 12: Ricostruire una conversazione tra due utenti

Concetto	Costrutto	Accessi	Tipo
Utente	E	1	Lettura
Annuncio_Utente	E	1	Lettura
Invio/Ricezione	R	10	Lettura
Messaggio	E	10	Lettura
Invio/Ricezione	R	10	Lettura
		Totale: 32L $\rightarrow$ 960.000 al giorno	

OP 13: Andamento del prezzo di un immobile in funzione del tempo

Concetto	Costrutto	Accessi	Tipo
Immobile	E	1	Lettura
Riguarda	R	1.3	Lettura
Annuncio	E	1.3	Lettura
		Totale: 3.6L $\rightarrow$ 18.000 al giorno	

OP 14: Comparare il prezzo di un immobile al mq con quello degli immobili nella stessa zona

Concetto	Costrutto	Accessi	Tipo
Zona	E	1	Lettura
Situato	R	1	Lettura
Immobile	E	1	Lettura
Riguarda	R	1	Lettura
Annuncio	E	1	Lettura
Annuncio_Utente	E	1	Lettura
Vendita	E	1	Lettura
		Totale: 7L $\rightarrow$ 70.000 al giorno	

OP 15: Comparare il prezzo di un immobile al mq con quello degli immobili nella stessa città

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
Divisione	R	10	Lettura
Zona	E	10	Lettura
Situato	R	1	Lettura
Immobile	E	1	Lettura
Riguarda	R	1	Lettura
Annuncio	E	1	Lettura
Annuncio_Utente	E	1	Lettura
Vendita	E	1	Lettura
		Totale: 27L → 202.500 al giorno	

OP 16: Comparare il prezzo medio al mq di una zona con quello della città

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
Divisione	R	10	Lettura
Zona	E	10	Lettura
		Totale: 21L → 73.500 al giorno	

OP 17: Ordinare le zone di una città per costo medio al mq

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
Divisione	R	10	Lettura
Zona	E	10	Lettura
		Totale: 21L → 31.500 al giorno	

OP 18: Stilare una top 5 città per una o più categorie

Concetto	Costrutto	Accessi	Tipo
Anno	E	1	Lettura
Città_Anno	E	700	Lettura
Appartenenza	R	5	Lettura
Città	E	5	Lettura
		Totale: 711L → 2.133.000 al giorno	

OP 19: Classificare città in base all'evoluzione in una o più categorie rispetto all'anno precedente

Concetto	Costrutto	Accessi	Tipo
Anno	E	2	Lettura
Città_Anno	E	1.400	Lettura
Appartenenza	R	1.400	Lettura
Città	E	1.400	Lettura
		Totale: 4.202L → 2.101.000 al giorno	

OP 20: Ordinare le città in base a valori specifici di una o più categorie

Concetto	Costrutto	Accessi	Tipo
Anno	E	1	Lettura
Città_Anno	E	700	Lettura
Valutazione_A	R	700	Lettura
Ambiente	E	700	Lettura
Valutazione_T	R	700	Lettura
Trasporto	E	700	Lettura
Valutazione_E	R	700	Lettura
Economia	E	700	Lettura
Valutazione_S	R	700	Lettura
Sanità	E	700	Lettura
Valutazione_I	R	700	Lettura
Istruzione	E	700	Lettura
		Totale: 7.701L → 7.701.000 al giorno	

OP 21: Calcolare performance di uno stato in una o più categorie (da ogni città)

Concetto	Costrutto	Accessi	Tipo
Stato	E	1	Lettura
Composizione	R	14	Lettura
Città	E	14	Lettura
Appartenenza	R	28	Lettura
Città_Anno	E	28	Lettura
		Totale: 85L → 170.000 al giorno	

OP 22: Aggiornamento annuale dei punteggi di una città (dati i campi delle categorie)

Concetto	Costrutto	Accessi	Tipo
Città	E	1	Lettura
Appartenenza	R	1	Lettura
Città_Anno	E	1	Lettura
		Totale: 85L → 170.000 al giorno	

3.4 Raffinamento dello schema

Eliminazione delle gerarchie

La gerarchia che suddivide le tipologie di *account*, avendo copertura totale e detenendo le due specializzazioni mansioni differenti, viene raffinata attraverso un collasso verso il basso. Nonostante le ridondanze di alcuni campi, abbiamo ritenuto questo fosse l'approccio migliore.

Nel caso dell'entità *immobile* si è invece deciso di procedere accorpando le entità figlie *villa, appartamento e stanza* nella prima. Questa scelta deriva dal mancato (o meglio trascurabile) utilizzo delle specializzazioni nelle operazioni precedentemente formulate.

Nella riformulazione della doppia gerarchia *annuncio-annuncio_utente* abbiamo adottato due criteri distinti. In primis abbiamo scisso l'*annuncio* in *asta, annuncio_utente* poichè modellano due concetti semanticamente diversi. Diversamente, per l'entità *annuncio_utente* abbiamo optato per un collasso verso l'alto essendo che la quasi totalità delle operazioni interagisce esclusivamente con l'entità madre, e risulterebbe pertanto vana la differenziazione delle entità per due soli campi, che sono stati invece riportati come opzionali in *annuncio_utente*.

Eliminazione degli attributi composti

L'unico attributo composto che compare nello schema concettuale è il campo *indirizzo*, contenuto nell'entità *immobile*. Avendo ciascun immobile uno ed un solo indirizzo, si è deciso di scomporre questo nelle sue componenti, *via, numeroCivico*, anch'esse uniche, senza dover quindi prestare attenzione alla loro coerenza a livello applicativo.

Scelta delle chiavi primarie

Lo schema presenta già tutte le chiavi primarie necessarie ad individuare le singole entità. Per ciascuna delle entità "categoria" si è optato per la realizzazione di algoritmi di hashing che, presi in input i valori dei campi di una categoria, generano un intero positivo univoco che assumerà il ruolo di chiave primaria. Segue una breve trattazione sulla progettazione di tali algoritmi.

Per creare hash univoci serve una funzione invertibile, ovvero è possibile ricavare l'input partendo dall'output della funzione. Supponiamo di voler creare un hash partendo da due dati A e B tali che A $\in$ [0, 5] e B $\in$ [0,7] Si può semplicemente usare la funzione $f1 = A * 10 + B$, intuitivamente le decine del codice ricavato rappresentano A e le unità B. È altrettanto ovvio però che ci siano molti valori fra 0 e 57 (f1min e f1max) che rimangono inutilizzati, causando perciò spreco di spazio. Proponiamo quindi una soluzione ottimizzata con $f2 = A * 8 + B$. Usiamo la base 8 piuttosto che la base 10, eliminando lo spreco di spazio che avevamo precedentemente. La base non è scelta a caso, per salvare B la base minima necessaria è la base 8, poichè si usano 8 simboli diversi per rappresentare il dato. Adesso tutti i dati fra f1min e f1max sono validi.

Ma se dovessimo prendere in considerazione più di due dati? Inserisco C [0,1], un booleano per esempio. Proviamo $f3 = C * 64 + A * 8 + B$. Non è una soluzione orrenda, ma si ripresenta lo spreco di possibili valori intermedi, questa volta su A. Per rappresentare A non è necessaria la base 8, basterebbe la base 6, perciò tutti i valori di f3 in cui A $\in$ [6,7] sono sprecati... Introduciamo le basi miste! Non dobbiamo per forza usare la stessa base per ogni cifra. Riprendiamo un attimo $f3 = C*64+A*8+B$. Qui il problema è il coefficiente applicato a C, ma come si trova il coefficiente ottimale? Semplice, basta trovare il massimo valore che può assumere l'espressione $A * 8 + B$. Sostituiamo $A = 5$ e $B = 7$ per trovare $5 * 8 + 7 = 47$. Per la logica delle basi, il numero successivo azzererà i valori di A e B e incrementerà di 1 C, quindi il coefficiente ottimale di C è $47 + 1 = 48$.

Questo metodo funziona con un numero variabile di dati, basta trovare il valore massimo assunto dall'espressione che lo precede e incrementarlo di 1 per il nuovo coefficiente. L'ultima cosa da controllare è quindi che il dato sia immagazzinabile in una variabile. Basta controllare che il valore massimo del hash sia minore o uguale di 2^{BIT} - 1, dove BIT è il numero di bit a disposizione (in genere 32). Il tipo di variabile ottimale per salvare questo hash è un unsigned int.

```
hashAmbiente = (pm25media*10) * 1001
        + percentualeSpazioVerdeUrbano * 10;
hashEconomia = (tassoDisoccupazione*10) * 7239535
        + (PILProCapite/1000) * 8035
        + stipendioMedio-300;
hashIstruzione = (numeroUniversita*10) * 1002001
        + (percentualeDiplomati * 10) * 1001
        + percentualeLaureati * 10;
hashSanita = (postiLettoProCapite*10) * 501
        + (aspettativaVita * 10 - 500);
hashTrasporto = (autoProCapite*100) * 151
        + percorrenzaMediaPendolare * 100;
```

Eliminazione degli identificatori esterni

Nello schema E/R sono eliminate le seguenti relazioni:
- *Composizione*, importando il nome dello stato in *città*
- *AppartenenzaAnno*, importando anno in *città_anno*
- *AppartenenzaCittà*, importando idCittà in *città_anno*
- *Valutazione_A*, importando hashAmbiente in *città_anno*
- *Valutazione_T*, importando hashTrasporto in *città_anno*
- *Valutazione_E*, importando hashEconomia in *città_anno*
- *Valutazione_S*, importando hashSanita in *città_anno*
- *Valutazione_I*, importando hashIstruzione in *città_anno*
- *Divisione*, importando idCittà in *zona*
- *Situato*, importando idZona in *immobile*
- *RiguardaAnnuncio*, importando idImmobile in *annuncio_utente*
- *RiguardaAsta*, importando idImmobile in *asta*
- *Gestita*, importando l'email identificante il giudice in *asta*
- *Riceve*, importando idAnnuncio in *rialzo*
- *Offerta*, importando l'email identificante l'utente in *rialzo*
- *Pubblicazione*, importando l'email identificante l'utente in *annuncio_utente*
- *Invio/Ricezione*, importando l'email identificante l'utente in *messaggio*
- *Ricezione/Invio*, importando idAnnuncio in *messaggio*

3.5 Analisi delle ridondanze

Durante la fase di sviluppo dello schema concettuale, si è deciso di aggiungere il campo *costoMedioMq* all'entità *zona*, poichè ritenuta una proprietà intrinseca a ciascun quartiere ed adatta alla formulazione di statistiche proiettate nel livello applicativo. Il dato in questione, tuttavia, si sarebbe potuto ottenere andando ad analizzare tutti gli *immobili* situati nella *zona* di interesse, facendo la media delle divisioni tra il costo di ciascuna abitazione per la sua metratura. Quest'ultima manovra, già in origine percepita come eccessivamente onerosa a livello computazionale, verrà messa a confronto con l'alternativa adottata, nelle due operazioni coinvolte.

OP 2: Pubblicare un annuncio immobiliare (vendita o affitto)

Con ridondanza

Concetto	Costrutto	Accessi	Tipo
Annuncio	E	1	Scrittura
Riguarda	R	1	Scrittura
Immobile	E	1	Scrittura
Situato	R	1	Scrittura
Zona	E	2	Scrittura
		Totale: 6.000 al giorno	

Senza ridondanza

Concetto	Costrutto	Accessi	Tipo
Annuncio	E	1	Scrittura
Riguarda	R	1	Scrittura
Immobile	E	1	Scrittura
		Totale: 3.000 al giorno	

OP 14: Comparare il prezzo di un immobile al mq con quello degli immobili nella stessa zona

Con ridondanza

Concetto	Costrutto	Accessi	Tipo
Immobile	E	1	Lettura
Situato	R	1	Lettura
Zona	E	1	Lettura
		Totale: 30.000 al giorno	

Senza ridondanza

Concetto	Costrutto	Accessi	Tipo
Immobile	E	50	Lettura
Situato	R	50	Lettura
Zona	E	1	Lettura
		Totale: 1.010.000 al giorno	

Il paragone del numero di letture e scritture necessarie alle operazioni nei due diversi scenari dimostra la correttezza della scelta di mantenere la ridondanza. È importante notare tuttavia come la maggiore efficienza sia resa possibile dal campo *numeroImmobili* di *zona* che ci esonera dalla rilettura di tutti gli immobili di una determinata zona nel caso in cui dovessimo andare a modificare il *costoMedioMq* in occasione della pubblicazione di un nuovo annuncio (ci basterà infatti moltiplicare costoMedioMq per numeroImmobili, sommare il costo al mq del nuovo immobile e dividere per il nuovo numeroImmobili, andando ad aggiornare quest'ultimo).

3.6 Traduzione di entità e associazioni in relazioni

stati(<u>nome</u>, superficie, popolazione)

citta(<u>idCitta</u>, nome, popolazione, superficie, regione*, nomeStato)

FK: nomeStato REFERENCES **stati**

citta_anni(<u>idCitta</u>, <u>anno</u>, punteggioAmbiente, punteggioTrasporto, punteggioEconomia, punteggioSanità, punteggioIstruzione, hashAmbiente, hashTrasporto, hashEconomia, hashSanita, hashIstruzione)

FK: idCitta REFERENCES **citta**

FK: anno REFERENCES **anni**

FK: hashAmbiente REFERENCES **ambiente**

FK: hashTrasporto REFERENCES **trasporto**

FK: hashEconomia REFERENCES **economia**

FK: hashSanita REFERENCES **sanita**

FK: hashIstruzione REFERENCES **istruzione**

anni(anno)

ambiente(hashAmbiente, pm25media, percentualeSpazioVerdeUrbano)

trasporto(hashTrasporto, percorrenzaMediaPendolare, autoProCapite)

economia(hashEconomia, PILProCapite, stipendioMedio, tassoDisoccupazione)

sanita(hashSanita, aspettativaVita, postiLettoProCapite)

istruzione(hashIstruzione, percentualeLaureati, percentualeDiplomati, numeroUniversita)

zone(idZona, idCitta, nome, costoMedioMq, numeroImmobili)

immobili(idImmobile, idZona, metriQuadri, classeEnergetica*, annoCostruzione*, via, numeroCivico, numeroInterno*, piano*, numeroStanze*, numeroConquilini*, tipoImmobile)

FK: idZona REFERENCES **zone**

annunci_utente(idAnnuncio, idImmobile, email, statoAnnuncio, dataCreazione, tipoAnnuncioUtente, costoMensile*, prezzo*)

FK: idImmobile REFERENCES **immobili**

FK: email REFERENCES **utenti**

aste(idAsta, idImmobile, email, prezzoMinimo, rialzoMinimo, depositoCauzionale, dataFine, dataCreazione)

FK: idImmobile REFERENCES **immobili**

FK: email REFERENCES **giudici_esecuzione**

rialzi(prezzoAttuale, idAsta, email, dataRialzo)

FK: idAsta REFERENCES **aste**

FK: email REFERENCES **utenti**

giudici_esecuzione(email, telefono*, nome)

utenti(email, telefono*, nome)

messaggi(email, idAnnuncio, timestamp, testo, mittente)

FK: idAnnuncio REFERENCES **annunci_utente**

FK: email REFERENCES **utenti**

3.7 Costruzione delle tabelle del DB in linguaggio SQL

ambiente

```
CREATE TABLE ambiente (
    hashAmbiente int NOT NULL,
    pm25media float NOT NULL,
    percentualeSpazioVerdeUrbano float NOT NULL,
    PRIMARY KEY(hashAmbiente)
)
```

anni

```
CREATE TABLE Anni (
    anno int NOT NULL,
    PRIMARY KEY(anno)
)
```

annunci_utente

```
CREATE TABLE annunci_utente (
    idAnnuncio int NOT NULL AUTO_INCREMENT,
    idImmobile int NOT NULL FOREIGN KEY REFERENCES immobili.idImmobile,
    email varchar(25) NOT NULL FOREIGN KEY REFERENCES utenti.email,
    statoAnnuncio varchar(20) NOT NULL
        CHECK(statoAnnuncio = 'Attivo' OR 'Inattivo'),
    dataCreazione datetime NOT NULL,
    tipoAnnuncioUtente varchar(20) NOT NULL
        CHECK(tipoAnnuncioUtente = 'Vendita' OR 'Affitto')
    costoMensile int DEFAULT NULL CHECK (costoMensile > 0),
    prezzo int DEFAULT NULL CHECK (prezzo > 0),
```

```
    PRIMARY KEY(idAnnuncio, idImmobile, email)
)
```

aste

```
CREATE TABLE aste (
    idAsta int NOT NULL AUTO_INCREMENT,
    idImmobile int DEFAULT NULL FOREIGN KEY REFERENCES immobiliidImmobile,
    email varchar(25) NOT NULL FOREIGN KEY REFERENCES giudici_esecuzioneemail,
    dataCreazione datetime NOT NULL,
    prezzoMinimo int NOT NULL CHECK (prezzoMinimo > 0),
    rialzoMinimo int NOT NULL CHECK (rialzoMinimo > 0),
    depositoCauzionale int NOT NULL
        CHECK (depositoCauzionale > 0),
    dataFine datetime NOT NULL,
    PRIMARY KEY(idAsta, idImmobile, email)
)
```

citta

```
CREATE TABLE citta (
    idCitta int NOT NULL AUTO_INCREMENT,
    nomeStato char(25) NOT NULL FOREIGN KEY REFERENCES stati.nome,
    nome char(50) NOT NULL,
    popolazione int NOT NULL CHECK (popolazione > 0),
    superficie int NOT NULL CHECK (superficie > 0),
    regione char(25) DEFAULT NULL,
    PRIMARY KEY(idCitta, nomeStato)
)
```

citta_anni

```
CREATE TABLE citta_anni (
    idCitta int NOT NULL FOREIGN KEY REFERENCES citta.idCitta,
    anno int NOT NULL FOREIGN KEY REFERENCES anni.anno,
    punteggioAmbiente int NOT NULL CHECK (punteggioAmbiente > 0),
```

```
    punteggioTrasporto int NOT NULL
        CHECK (punteggioTrasporto > 0),
    punteggioEconomia int NOT NULL CHECK (punteggioEconomia > 0),
    punteggioSanita int NOT NULL CHECK (punteggioSanita > 0),
    punteggioIstruzione int NOT NULL
        CHECK (punteggioIstruzione > 0),
    hashAmbiente int NOT NULL FOREIGN KEY REFERENCES ambiente.hashAmbiente,
    hashEconomia bigint NOT NULL FOREIGN KEY REFERENCES economia.hashEconomia,
    hashIstruzione int NOT NULL FOREIGN KEY REFERENCES istruzione.hashIstruzione
    hashSanita int NOT NULL FOREIGN KEY REFERENCES sanita.hashSanita,
    hashTrasporto int NOT NULL FOREIGN KEY REFERENCES trasporto.hashTrasporto,
    PRIMARY KEY(idCitta, anno)
)
```

economia

```
CREATE TABLE economia (
    hashEconomia bigint NOT NULL,
    PILProCapite float NOT NULL CHECK (PILProCapite > 0),
    stipendioMedio int NOT NULL CHECK (stipendioMedio > 0),
    tassoDisoccupazione float NOT NULL,
    PRIMARY KEY(hashEconomia)
)
```

giudici_esecuzione

```
CREATE TABLE giudici_esecuzione (
    email varchar(30) NOT NULL,
    telefono varchar(10) CHECK (length(telefono)=10),
    nome char(25) NOT NULL,
    PRIMARY KEY(email)
)
```

immobili

```
CREATE TABLE immobili (
```

```
    idImmobile int NOT NULL AUTO_INCREMENT,
    idZona int NOT NULL FOREIGN KEY REFERENCES zone.idZona,
    metriQuadri int NOT NULL CHECK (metriQuadri > 0),
    classeEnergetica varchar(50)
        CHECK(classeEnergetica = 'A+' OR 'A' OR 'B' OR 'C' OR 'D'
OR 'E' OR
        'F' OR 'G'),
    annoCostruzione int CHECK (annoCostruzione > 0),
    via varchar NOT NULL,
    numeroCivico int NOT NULL,
    tipoImmobile varchar(50) NOT NULL
        CHECK(tipoImmobile = 'Appartamento' OR 'Villa' OR 'Stanza'),
    numeroInterno int,
    piano int,
    numeroStanze int CHECK (numeroStanze > 0),
    numeroConquilini int,
    PRIMARY KEY(idImmobile, idZona)
)
```

istruzione

```
CREATE TABLE istruzione (
    hashIstruzione int NOT NULL,
    percentualeLaureati float NOT NULL
        CHECK (percentualeLaureati > 0),
    percentualeDiplomati float NOT NULL
        CHECK (percentualeDiplomati > 0),
    numeroUniversita int NOT NULL,
    PRIMARY KEY(hashIstruzione)
)
```

messaggi

```
CREATE TABLE messaggi (
    timestamp datetime NOT NULL,
```

```
    email varchar(25) NOT NULL FOREIGN KEY REFERENCES utenti.email,
    idAnnuncio int NOT NULL FOREIGN KEY REFERENCES annunci_utente.idAnnuncio,
    testo text NOT NULL,
    mittente varchar(25) NOT NULL,
        CHECK(mittente = 'Richiedente' OR 'Venditore'),
    PRIMARY KEY(timestamp, email, idAnnuncio)
)
```

rialzi

```
CREATE TABLE rialzi (
    prezzoAttuale int NOT NULL CHECK (prezzoAttuale > 0),
    email varchar(25) NOT NULL FOREIGN KEY REFERENCES utenti.email,
    idAsta int NOT NULL FOREIGN KEY REFERENCES annunci_utente.idAsta,
    dataRialzo datetime NOT NULL,
    PRIMARY KEY(prezzoAttuale, email, idAsta)
)
```

sanita

```
CREATE TABLE sanita (
    hashSanita int NOT NULL,
    aspettativaVita float NOT NULL CHECK (aspettativaVita > 0),
    postiLettoProCapite float NOT NULL
        CHECK (postiLettoProCapite > 0),
    PRIMARY KEY(hashSanita)
)
```

stati

```
CREATE TABLE stati(
    nome char(25) NOT NULL,
    superficie int CHECK (superficie > 0),
    popolazione int CHECK (popolazione > 0),
    PRIMARY KEY(nome)
)
```

trasporto

```
CREATE TABLE trasporto (
    hashTrasporto int NOT NULL,
    percorrenzaMediaPendolare float NOT NULL
        CHECK (percorrenzaMediaPendolare > 0),
    autoProCapite float NOT NULL
        CHECK (autoProCapite > 0),
    PRIMARY KEY(hashTrasporto)
)
```

utenti

```
CREATE TABLE utenti (
    email varchar(30) NOT NULL,
    telefono varchar(10) CHECK (length(telefono)=10),
    nome char(25) NOT NULL,
    PRIMARY KEY(email)
)
```

zone

```
CREATE TABLE zone (
    idZona int NOT NULL AUTO_INCREMENT,
    idCitta int NOT NULL FOREIGN KEY REFERENCES citta.idCitta,
    nome char(25) NOT NULL,
    costoMedioMq float NOT NULL CHECK (costoMedioMq > 0),
    numeroImmobili int NOT NULL CHECK (numeroImmobili ≥ 0),
    PRIMARY KEY(idZona, idCitta)
)
```

3.8 Schema relazionale finale

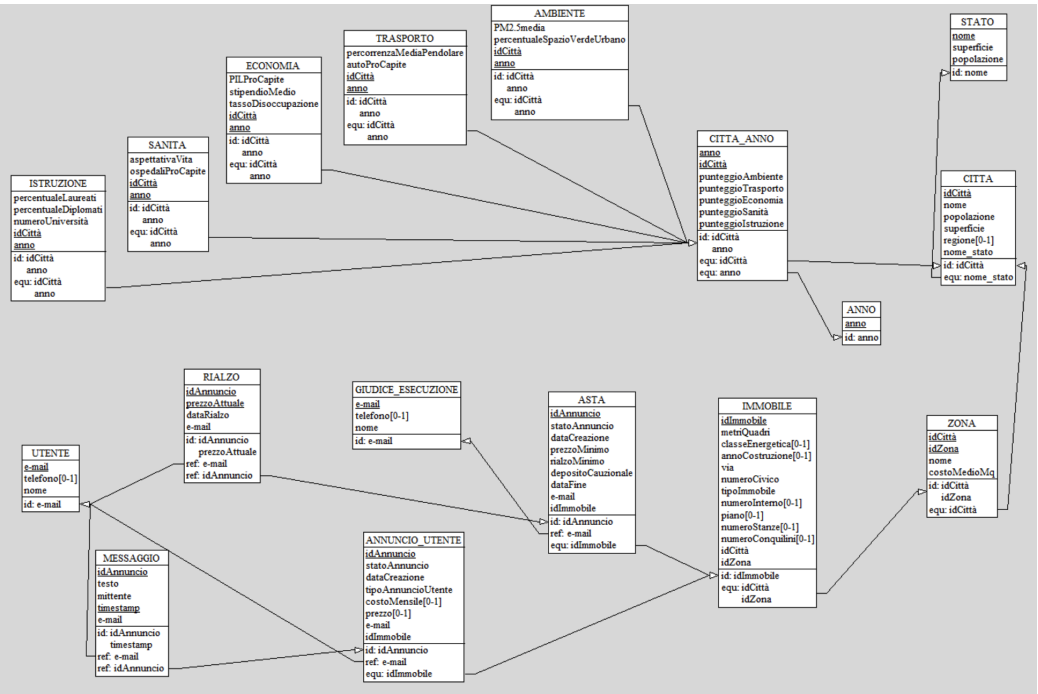

Figure 3.1: Schema relazionale finale

3.9 Traduzione delle operazioni in query SQL

OP 1: Registrare un nuovo account

```
INSERT INTO utenti (email, telefono, nome)
VALUES (?, ?, ?)
```

OP 2: Pubblicare un annuncio immobiliare (vendita o affitto)

```
IF (SELECT COUNT(*)
    FROM immobili
    WHERE via = ?  AND numeroCivico = ?  AND idCitta = ?  > 0,
    TRUE, FALSE)
```

Se tale condizione risulta soddisfatta, procediamo all'inizializzazione di *annuncio_utente* con i parametri appena trovati. Se non dovesse essere soddis-

fatta, vengono inseriti i dati di input. In maniera analoga viene eseguita la query di inserimento di aste da parte dei giudici

```
INSERT INTO annunci_utente (idAnnuncio, idImmobile, email, statoAnnuncio,
dataCreazione, tipoAnnuncioUtente, costoMensile, prezzo)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)
```

OP 3: Visualizzare annunci immobiliari attivi di una città in ordine cronologico

```
SELECT A.*, Z.idCitta, Z.idZona
FROM (immobili I JOIN annunci_utente A ON (A.idImmobile = I.idImmobile))
JOIN zone Z ON (I.idZona = Z.idZona AND Z.idCitta = ?  AND A.statoAnnuncio
= 'Attivo')
ORDER BY A.dataCreazione DESC
```

OP 4: Suddividere gli annunci in vendita, affitto o asta all'interno di una città

```
SELECT A.*
FROM (immobili I JOIN annunci_utente A ON (A.idImmobile = I.idImmobile))
JOIN zone Z ON (I.idZona = Z.idZona AND Z.idCitta = ?  AND A.statoAnnuncio
= 'Attivo' AND A.tipoAnnuncioUtente = ?)
ORDER BY A.dataCreazione DESC
```

Similmente vengono proiettate le aste attive in una data città in ordine di creazione decrescente.

OP 5: Ordinare gli annunci in base a particolari filtri sull'immobile (e.g. metratura, tipoImmobile)

```
SELECT A.*, I.tipoImmobile
FROM immobili I JOIN annunci_utente A ON (I.idImmobile = A.idImmobile
AND A.statoAnnuncio = 'Attivo' AND I.tipoImmobile = ?)
WHERE EXISTS (
    SELECT *
```

```
    FROM zone Z, citta C
    WHERE C.idCitta = ?
    AND Z.idCitta= C.idCitta
    AND Z.idZona = I.idZona
)
```

OP 6: Filtrare gli annunci per zona

```
SELECT A.*, I.idZona
FROM immobili I JOIN annunci_utente A ON (I.idImmobile = A.idImmobile
AND A.statoAnnuncio = 'Attivo' AND I.idZona = ?)
ORDER BY dataCreazione DESC
```

OP 7: Ordinare gli annunci di affitto per costo mensile

```
SELECT A.*, I.idZona
FROM immobili I JOIN annunci_utente A ON (I.idImmobile = A.idImmobile
AND A.statoAnnuncio = 'Attivo' AND A.tipoAnnuncioUtente = 'Affitto')
WHERE EXISTS (
    SELECT *
    FROM zone Z, citta C
    WHERE C.idCitta = ?
    AND Z.idCitta = C.idCitta
    AND Z.idZona = I.idZona
) ORDER BY A.costoMensile DESC
```

OP 8: Mostrare aste attive in una città

```
SELECT A.*, Z.idCitta
FROM (immobili I JOIN aste A
ON (A.idImmobile = I.idImmobile)) JOIN zone Z
ON (I.idZona = Z.idZona AND Z.idCitta = ?  AND A.dataFine > NOW())
ORDER BY A.dataCreazione DESC
```

OP 9: Ordinare le aste per prezzo attuale crescente

```
SELECT R.idAsta, MAX(R.prezzoAttuale) AS prezzo_attuale
```

```
FROM rialzi R, aste A
WHERE R.idAsta = A.idAsta
AND A.dataFine > DATE(NOW())
GROUP BY A.idAsta
ORDER BY prezzo_attuale
```

OP 10: Effettuare un rialzo all'interno di un'asta (controllando la sua validità)

```
IF ((SELECT MAX(R.prezzoAttuale)
    FROM rialzi R
    WHERE idAsta = ?)  <=
    ?  - (SELECT rialzoMinimo
    FROM aste
    WHERE idAsta = ?),
THEN INSERT INTO rialzi(prezzoAttuale, email, idAsta, dataRialzo)
VALUES(?, ?, ?, ?),
END IF;
```

OP 11: Contattare un utente in merito ad un annuncio creato

```
INSERT INTO messaggi (timestamp, email, idAnnuncio, testo, mittente)
VALUES (CURRENT_TIMESTAMP(), ?, ?, ?, ?)
```

OP 12: Ricostruire una conversazione tra due utenti

```
SELECT testo, mittente, timestamp
FROM messaggi
WHERE email = ?
AND idAnnuncio = ?
ORDER BY timestamp
```

OP 13: Andamento del prezzo di un immobile in funzione del tempo

```
SELECT dataCreazione, prezzo, costoMensile
FROM annunci_utente
```

```
WHERE idImmobile = ?
ORDER BY dataCreazione DESC
```

OP 14: Comparare il prezzo di un immobile al mq con quello degli immobili nella stessa zona

```
SELECT A.idAnnuncio, ((prezzo/metriQuadri)-costoMedioMq) * 100/costoMedioMq
AS perc_diff
FROM (immobili I JOIN annunci_utente A
ON (A.idImmobile = I.idImmobile)) JOIN zone Z
ON (I.idZona = Z.idZona)
WHERE idImmobile = ?
AND statoAnnuncio = 'Attivo'
AND tipoAnnuncioUtente = 'Vendita'
```

OP 15: Comparare il prezzo di un immobile al mq con quello degli immobili nella stessa città

```
SET @costoMedioMqCitta = (
    SELECT SUM(costoMedioMq*numeroImmobili)
    FROM zone
    WHERE idCitta = ?
    ) / SELECT SUM(numeroImmobili)
    FROM zone
    WHERE idCitta = ?
SELECT ((prezzo/metriQuadri)-@costoMedioMqCitta) * 100/@costoMedioMqCitta
FROM (immobili I JOIN annunci_Utente A
ON (A.idImmobile = I.idImmobile)) JOIN Zone Z
ON (I.idZona = Z.idZona) JOIN Città C
ON (Z.idCittà = C.idCittà)
WHERE idImmobile = ?
AND tipoAnnuncioUtente = 'Vendita'
AND A.statoAnnuncio = 'Attivo'
```

OP 16: Comparare il prezzo medio al mq di una zona con quello

della città

```
SELECT *
FROM (
    SELECT idZona, (costoMedioMq-SUM(costoMedioMq * numeroImmobili)/
    SUM(numeroImmobili)) * 100 / SUM(costoMedioMq * numeroImmobili)/
    SUM(numeroImmobili)
    FROM zone
    WHERE idCitta = ?
    GROUP BY idZona
) AS percZonaCitta
WHERE percZonaCitta.idZona = ?
```

OP 17: Ordinare le zone di una città per costo medio al mq

```
SELECT *
FROM zone
WHERE idCitta = ?
ORDER BY costoMedioMq DESC
```

OP 18: Stilare una top 5 città per una o più categorie

```
SELECT TOP(5) WITH TIES
FROM citta_anni A
WHERE A.anno = ?
ORDER BY A.punteggioAmbiente*?+
A.punteggioTrasporto*?+A.punteggioEconomia*?+
A.punteggioSanità*?+A.punteggioIstruzione*?  DESC
```

OP 19: Classificare città in base all'evoluzione in una categoria rispetto all'anno precedente

```
SET @perc_sviluppo = (
    (SELECT punteggioAmbiente
    FROM citta_anni
    WHERE idCitta = 1
    AND anno = 2020) -
```

```
    (SELECT punteggioAmbiente
    FROM citta_anni
    WHERE idCitta = 1
    AND anno = 2020-1)
    ) *100 /
    (SELECT punteggioAmbiente
    FROM citta_anni
    WHERE idCitta = 1
    AND anno = 2020-1);
SELECT @perc_sviluppo
```

OP 20: Ordinare le città in base a valori specifici di una categoria

```
SELECT A.percentualeSpazioVerdeUrbano, C.anno, C.idCitta
FROM ambiente A, citta_anni C
WHERE A.hashAmbiente = C.hashAmbiente
AND C.anno = ?
ORDER BY A.percentualeSpazioVerdeUrbano DESC
```

OP 21: Calcolare performance di uno stato in una categoria (da ogni città)

```
SELECT AVG (A.punteggio*)
FROM (stati S JOIN citta C
ON (S.nome = C.nomeStato))
JOIN citta_anni A
ON (A.idCitta = C.idCitta AND anno = ?)
WHERE C.nomeStato = ?
```

OP 22: Aggiornamento annuale dei punteggi di una città (dati i campi delle categorie)

```
newHashAmbiente = CAST(pm25media*10 AS int)*1001 + CAST(percentualeSpazioVerdeUr
AS int)

IF ((SELECT COUNT(hashAmbiente)
FROM ambiente
```

```
WHERE hashAmbiente = newHashAmbiente) > 0,
NULL,
INSERT INTO ambiente(hashAmbiente, pm25media, percentualeSpazioVerdeUrbano)
VALUES (newHashAmbiente, ?, ?))
INSERT INTO citta_anni(punteggioAmbiente, newHashAmbiente)
```

Chapter 4

Progettazione dell'applicazione

4.1 Descrizione dell'architettura dell'applicazione realizzata

asdasd