Relazione

Bruniera Alvise

Calabrigo Massimo

Università degli studi di Udine

December 4, 2021

Contents

1 Introduzione

Il nostro obiettivo è creare un database postgres, per la gestione di uno studio medico. Vogliamo registrare informazioni sulle sedute e le terapie dei pazienti, sulle competenze e gli orari di lavoro dei medici, e degli altri dipendenti.

2 Requisiti e Specifiche

2.1 Workflow e fase di specificazione

Per questo progetto abbiamo deciso di utilizzare un modello iterativo incrementale. Abbiamo separato il progetto in tre processi: Requisiti e specifiche, Progettazione ed Implementazione; a loro volta divisi in sotto-processi:

1. Requisiti e Specifiche
    - Analisi: lettura del documento, evidenziando punti importanti
    - Specificazione: riassumere i punti importanti nelle specifiche
    - Validazione: controllo che le specifiche rispettino il documento, eventualmente tornando all'analisi
2. Progettazione
    - concettuale: stesura e ristrutturazione dell'ER dalle specifiche
    - logica: traduzione da ER a logico, validazione, ed implementazione su postgres

    - fisica: scelta degli indici

3. Implementazione
    - Implementazione di operazioni e viste
    - Analisi statistica dei dati

In questa sezione esponiamo il processo di "Requisiti e specifiche", e riportiamo solo i risultati finali.

Come prima cosa abbiamo letto il documento con le richieste del cliente (fornito dal professore) evidenziando i concetti principali, annotandoli nel glossario, e le richieste importanti. Poi abbiamo iniziato la prima fase di analisi dei requisiti, elencando e riordinando quello che avevamo evidenziato. Prima di passare alla fase di specificazione abbiamo riletto i requisiti e il documento delle richieste cercando incompletezze ed errori, quindi abbiamo raffinato i requisiti (seconda iterazione di requisiti). Quindi siamo passati ad una prima iterazione delle specifiche, in cui abbiamo cominciato a risolvere le ambiguità, documentando la soluzione nel glossario e semplificando la descrizione. In una prima fase di validazione abbiamo notato che non erano chiari alcuni dettagli riguardanti terapie prolungate ed appuntamenti, e se gli appuntamenti fossero da considerarsi sedute programmate; Quindi siamo tornati all'analisi dei requisiti, risolvendo questo dubbio, ed abbiamo proseguito con l'ultima fase di specificazione e validazione.

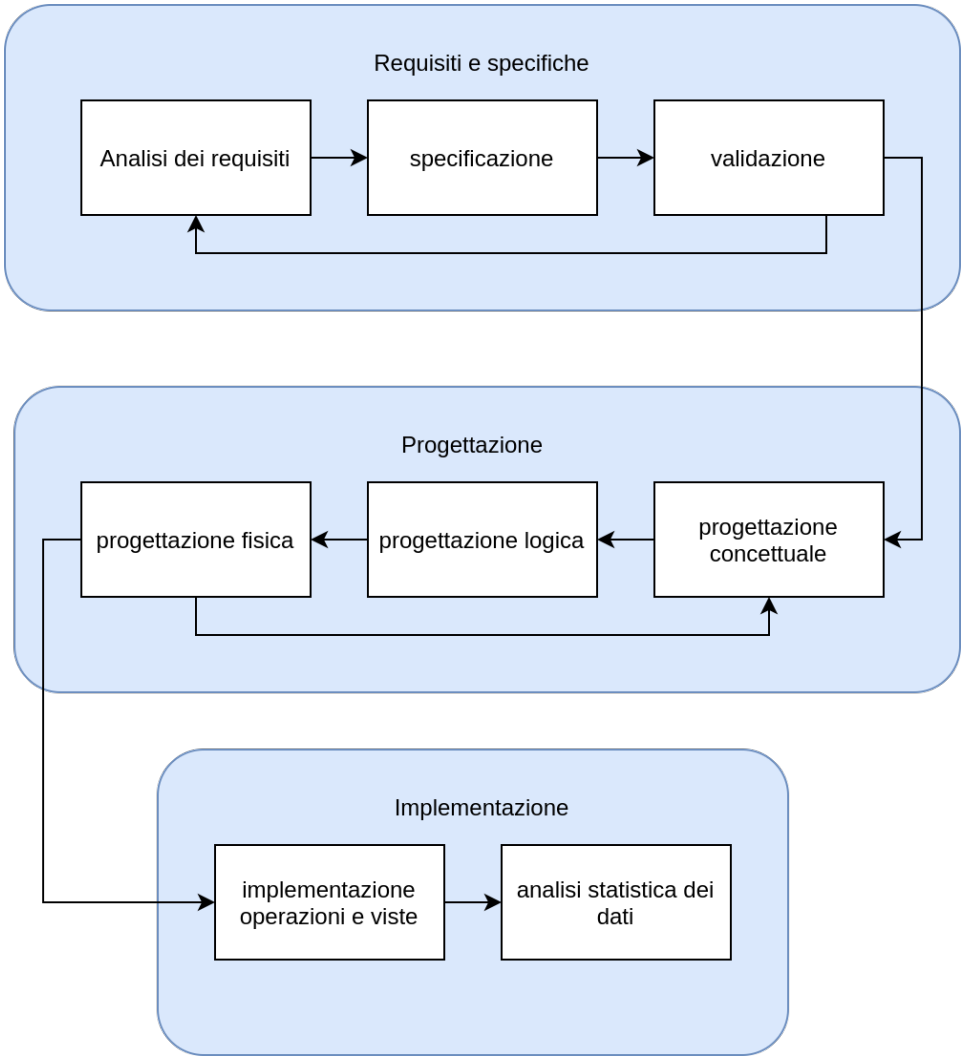

Figure 1: workflow

2.2 Glossario

Nel corso del processo di specificazione abbiamo annotato i vari termini specifici del dominio, risolvendo le ambiguità. L'elenco riportato è riferito all'ultima iterazione di requisiti e specifiche.

- Medico: Ogni medico può essere interno od esterno.
- Medico interno: Medico comproprietario dello studio medico.

- Medico esterno: Medico non comproprietario dello studio medico.
- Codice-medico: Codice che identifica univocamente un medico.
- Membro del personale ausiliario: Ogni membro può essere assistente medico, oppure amministrativo, può anche essere entrambe le cose.
- Codice-personale: Codice che identifica univocamente un membro del personale ausiliario.
- Paziente: cliente dello studio medico.
- Paziente regolare: Paziente che si sottopone ad almeno una terapia prolungata.
- Paziente occasionale: Paziente che si sottopone ad almeno una seduta per un problema urgente.
- Specializzazione: Titolo di studio acquisito da tutti medici dopo la laurea. (Oculistica, urologia, cardiologia, ...)
- Qualifica: Titoli di studio specifici dei membri del personale ausiliario. (Diploma di ragioneria, laurea in infermieristica, tecnico radiologo, ...)
- Storico: Resoconto periodico.
- Denominazione: Titolo di un corso di aggiornamento. ("Corso di aggiornamento in pneumologia", ...)
- Terapia prolungata: Trattamento prolungato a cui si sottopone il paziente.
- Seduta: Visita occasionale a cui si sottopone il paziente per motivi urgenti. La singola seduta deve risolvere il problema, altrimenti sarebbe parte di una terapia prolungata.
- Appuntamento: Visita periodica a cui si sottopone il paziente come parte di una terapia prolungata. Quando ci si presenta ad un appuntamento viene comunicato un ambulatorio ed assegnati i membri del personale ausiliario ed i medici che si occuperanno della visita.
- Appuntamento accettato: appuntamento a cui il paziente si presenta. Può essere terminato o ancora in corso.

- Appuntamento saltato: appuntamento al quale il paziente non si è presentato.
- Appuntamento programmato: appuntamento fissato per una data e ora future.

2.3 Requisiti

Abbiamo riportato requisiti finali, risultato dell'ultima (terza) iterazione di analisi dei requisiti. Le iterazioni precedenti dei requisiti si trovano nel documento allegato "Specifiche.docx"

- I *medici* possono essere interni od esterni. Un medico è identificato univocamente da un codice-medico, ed ha un nome, un cognome, un indirizzo, un recapito telefonico, ed una o più *specializzazioni*.
  - I medici interni sono comproprietari ed hanno diritto su una percentuale degli incassi
  - I medici esterni hanno una tariffa oraria
- Ogni *membro del personale ausiliario* è identificato univocamente da un codice-personale, ed ha un nome, cognome, indirizzo, recapito telefonico (uno), ed una o più qualifiche.
  - Gli assistenti medici possono seguire dei corsi di aggiornamento.
  - Amministrativi.
- Ogni *corso di aggiornamento* è identificato univocamente dalla denominazione, dal luogo dove si svolge, dalla data in cui si svolge. Due o più corsi di aggiornamento con la stessa denominazione non possono svolgersi nello stesso luogo alla stessa data.
- Ogni mese viene memorizzato *uno storico delle ore lavorative* ordinarie e straordinarie dei medici e dei membri del personale ausiliario.
- I *pazienti* possono essere regolari od occasionali. Un paziente è identificato univocamente dal codice fiscale, ed ha un nome, un cognome, un indirizzo, un recapito telefonico, ed una data di nascita.
  - I pazienti occasionali si presentano allo studio per un problema urgente da risolvere in una seduta.

  - I pazienti regolari si sottopongono ad una o più terapie prolungate. Un paziente regolare può essere anche occasionale per un problema urgente estraneo alla terapia prolungata.

- Ogni *seduta* è caratterizzata dalle persone coinvolte (un paziente, uno o più medici, uno o più membri del personale ausiliario), dalla data, l'ora, e l'ambulatorio in cui si svolge la seduta.

- Ogni *terapia prolungata* è caratterizzata dal paziente, da uno specifico tipo di medico e da una data di fine. Una terapia prolungata può essere aperta o chiusa, inizialmente è aperta e quando termina diventa chiusa. Ad un paziente in terapia prolungata aperta possono essere associati uno o più appuntamenti programmati, mentre ad una terapia prolungata chiusa solo appuntamenti accettati o saltati.

  - Gli appuntamenti possono essere programmati. In seguito, se il paziente si presenta all'ora e alla data dell'appuntamento programmato, l'appuntamento diventerà accettato, altrimenti diventerà saltato. Degli appuntamenti programmati o saltati non sono noti ambulatorio, medici e membri del personale ausiliario.

- Ogni *appuntamento* è caratterizzato dal paziente, dai medici e dai membri del personale ausiliario coinvolti, dalla data, l'ora e l'ambulatorio in cui si svolge.

- Lo studio medico dispone di un certo numero di *ambulatori*, dove ogni ambulatorio è identificato univocamente da una lettera.

2.4 Specifiche

Come per i requisiti, abbiamo riportato solo l'ultima iterazione (seconda) delle specifiche. Sullo stesso file ("Specifiche.docx") si trova anche la prima iterazione.

- Il medico è identificato univocamente dal codice-medico ed è caratterizzato da nome, cognome, indirizzo, un unico recapito telefonico e una o più specializzazioni. I medici interni hanno diritto a una percentuale degli incassi e i medici esterni hanno una tariffa oraria. Un medico si occupa di zero o più appuntamenti accettati. Il medico si occupa di zero o più sedute.

- Un medico ha una o più specializzazioni.

- Le specializzazioni sono: Oculistica, urologia, pneumologia, ...
- Il membro del personale ausiliario è identificato univocamente da codice-personale ed è caratterizzato da nome, cognome, indirizzo, da un unico recapito telefonico e da una o più qualifiche. Il membro del personale ausiliario può essere amministrativo, assistente medico, od entrambi. Il membro del personale ausiliario partecipa a zero o più appuntamenti accettati.
- Gli assistenti medici possono seguire nessuno o più corsi di aggiornamento.
- Le qualifiche sono: Diploma di ragioneria, laurea in infermieristica, tecnico radiologo, ...
- Un corso di aggiornamento è identificato univocamente dalla denominazione, dal luogo e data in cui si svolge.
- Lo storico mantiene per ogni mese il numero di ore ordinarie e straordinarie dei medici e dei membri del personale ausiliario.
- Il paziente è identificato univocamente dal codice fiscale, ed è caratterizzato da nome, cognome, indirizzo, un unico recapito telefonico e dalla data di nascita. Il paziente è occasionale, regolare o entrambi. Il paziente regolare si sottopone ad una o più terapie prolungate aperte, mentre il paziente occasionale si sottopone ad una o più sedute. Il paziente è sia regolare che occasionale se ha almeno una terapia prolungata aperta e si sottopone a una seduta.
- Ogni seduta è caratterizzata dal paziente, da uno o più medici, da uno o più membri del personale ausiliario, dalla data, dall'ora, e dall'ambulatorio in cui si svolge la seduta.
- Ogni terapia prolungata è caratterizzata dal paziente, da uno specifico tipo di medico da una data di inizio; e da una data di fine (quest'ultima è inserita alla fine della terapia prolungata). Una terapia prolungata può essere aperta o chiusa, inizialmente è aperta e quando termina diventa chiusa. Ad un paziente in terapia prolungata aperta possono essere associati uno o più appuntamenti programmati, tra cui almeno uno programmato.
- Ad una terapia prolungata chiusa possono essere associati appuntamenti accettati o appuntamenti saltati.

- Ogni appuntamento è caratterizzato dalla terapia prolungata, dalla data, l'ora e l'ambulatorio in cui si svolge. Un appuntamento può essere programmato, accettato o saltato.
  - È programmato quando è fissato per una data e ora future.
  - È accettato quando la data e l'ora sono passate, e il paziente si è presentato, e gli vengono assegnati uno o più medici e uno o più membri del personale ausiliario
  - È saltato accettato quando la data e l'ora sono passate, e il paziente non si è presentato
- Lo studio medico dispone di un certo numero di ambulatori, dove ogni ambulatorio è identificato univocamente da una lettera.
- Quando si assegna un medico a un appuntamento tra le specializzazioni del medico ci deve essere quella del tipo di specializzazione di terapia.

2.5 Diagramma dei Casi d'Uso

Per facilitare la progettazione abbiamo anche abbozzato un diagramma dei casi d'uso, che rappresenta alcune operazioni comuni che ci aspettiamo dal sistema gestito da questa base di dati.

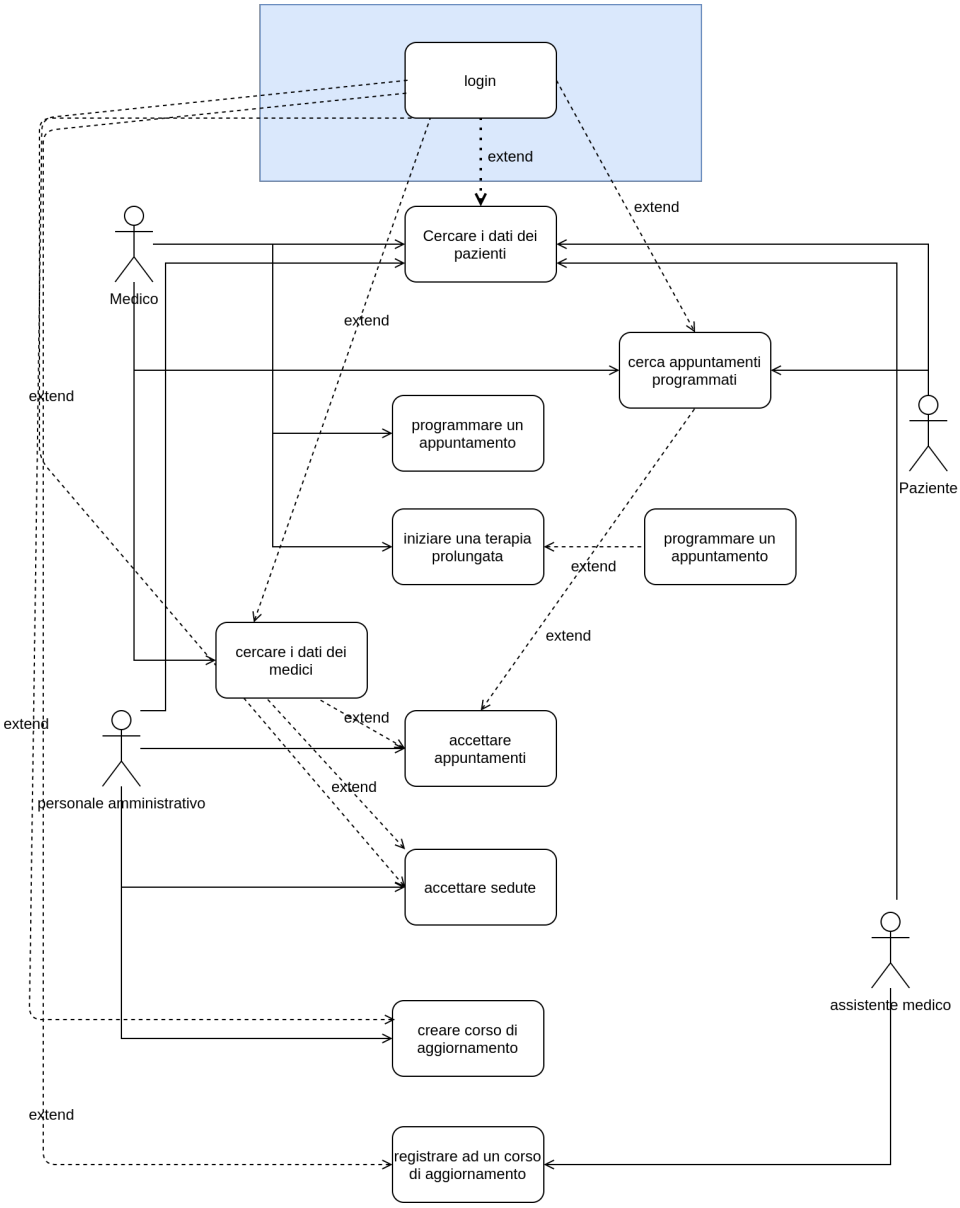

Figure 2: Casi d'Uso

3 ER e relazionale

3.1 ER

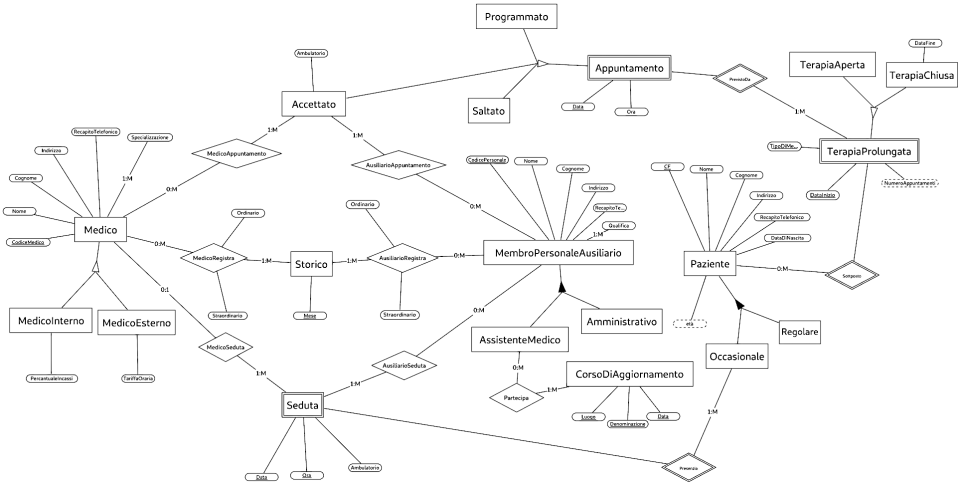

Figure 3: ER

3.1.1 Stesura

Per la stesura dell'ER abbiamo seguito un approccio misto di progettazione. Inizialmente abbiamo associato ad ogni punto delle specifiche, una entità, tranne che In alcuni casi particolari in cui un'entità era descritta da più punti (esempio: i primi tre punti descrivono due entità). Quindi ci siamo divisi le entità tra i membri del gruppo, ed individualmente le abbiamo raffinate e messe in relazione tra di loro.

Finito il lavoro individuale, abbiamo aggregato quanto prodotto individualmente in un unico ER aggiungendo le relazioni tra le due parti, e raffinato ulteriormente le entità, in particolare i collegamenti tra le parti sviluppate individualmente. Abbiamo poi verificato, che l'ER rispettasse le specifiche, anche analizzando i cicli; ad esempio consideriamo il ciclo: Specializzazioni - terapia prolungata - appuntamento accettato - Medico - Specializzazioni, detto che ad ogni terapia prolungata è associata una specializzazione, ed ogni medico può avere più specializzazioni (od anche nessuna), il medico che si occupa di un appuntamento, associato ad una terapia, deve essere specializzato *almeno* nella stessa specializzazione associata alla terapia. Molti cicli non erano problematici, quelli che lo erano sono stati risolti con un trigger 5.1.

3.1.2 Ristrutturazione

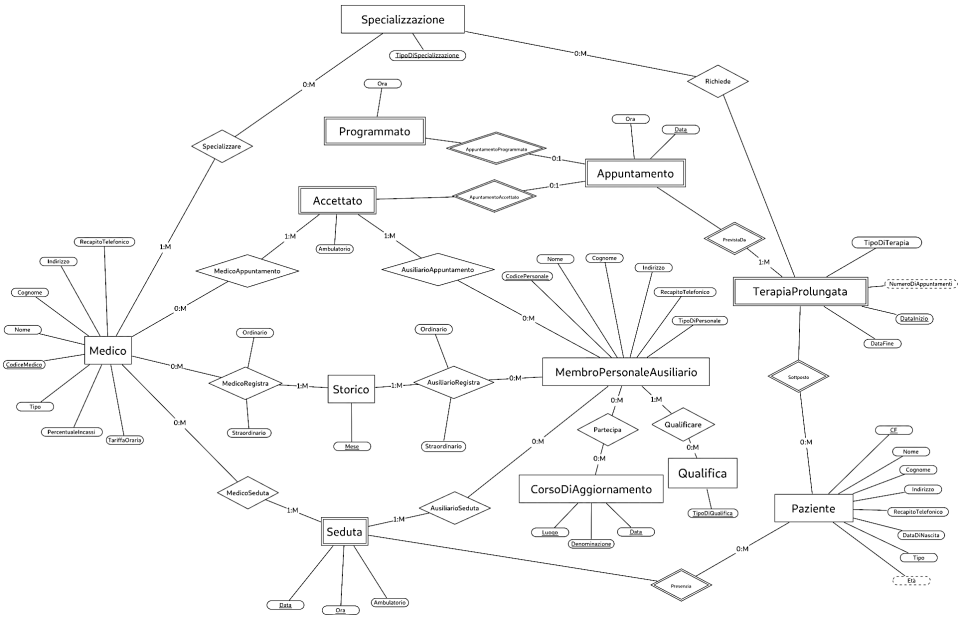

Figure 4: ER Ristrutturato

In questa fase abbiamo risolto le specializzazioni, gli attributi composti, ed altri costrutti non rappresentabili direttamente in relazionale (come gli attributi con cardinalità multipla). Al contrario della fase di stesura, la fase di ristrutturazione è stata svolta direttamente in coppia, invece che separando i compiti, questo sia per poter decidere meglio la soluzione più adatta quando un costrutto poteva essere risolto in più modi diversi, che per permettere a tutto il gruppo di prendere "familiarità" anche con le parti dello schema ER di cui non si è occupato personalmente. Come per i sottoprocessi precedenti, dopo la prima fase di ristrutturazione è seguita una fase di validazione, in cui abbiamo riesaminato l'intero schema, per controllare che fosse adeguato; quindi, insoddisfatti di come era stata trasformata la generalizzazione dell'entità "Appuntamento", abbiamo corretto l'errore in una seconda iterazione, prima di dichiarare conclusa la ristrutturazione.

All'atto pratico, abbiamo deciso di non reificare né la specializzazione di "Medico" (in MedicoInterno e MedicoEsterno), né la specializzazione di "MembroPersonaleAusiliario" (in AssistenteMedico ed Amministrativo), ma di inserire degli attributi "Tipo" e "TipoDiPersonale" per indicare la a quale entità specializzata appartengono, oltre all'aggiunta di opportuni vincoli e

trigger. Lo stesso è stato fatto per l'entità "Paziente". La specializzazione dell'entità "Appuntamento", invece, era stata inizialmente reificata completamente nelle sue tre varianti specializzate (Programmato, Accettato, e Saltato); ma nella seconda iterazione di ristrutturazione è stata rimossa l'entità "Saltato" perché ridondante, in quanto era sufficiente controllare la data e l'ora prefissate per l'appuntamento, e l'appartenenza o meno all'entità "Accettato" per sapere se un appuntamento era programmato o saltato. L'entità "Programmato" è stata mantenuta, nonostante fosse superflua come "Saltato" in seguito all'analisi delle ridondanze 3.1.3. Inoltre, si è scelto di reificare "Specializzazione" per semplificare altre operazioni sulla base di dati, principalmente per garantire il vincolo che richiede che il medico associato ad un appuntamento possieda la specializzazione richiesta dalla terapia.

3.1.3 Analisi delle ridondanze

Nome	Tipo	#istanze
Medico	E	20
Specializzazione	E	20
Programmato	E	500
Accettato	E	20000
Storico	E	120
Seduta	E	300000
Saltato	E	100
Appuntamento	E	20600
MembroPersonaleAusiliario	E	50
CorsoDiAggiornamento	E	50
Qualifica	E	20
TerapiaProlungata	E	7000
Paziente	E	6000
Specializzare	R	30
MedicoAppuntamento	R	25000
MedicoRegistra	R	2400
MedicoSeduta	R	301000
AppppuntamentoSaltato	R	100
AppuntamentoProgrammato	R	500
AppuntamentoAccettato	R	20000
AusiliarioAppuntamento	R	20000
AusiliarioRegistra	R	6000
AusiliarioSeduta	R	20000
Richiede	R	7000
PrevistaDa	R	20600
Partecipa	R	500
Qualificare	R	70
Presenzia	R	300000
Sottoposto	R	7000

Table 1: Numero di istanze di entità e relazioni

Operazioni	Tipo	Frequenza
Cercare dati pazienti	Interattivo	60
Cercare appuntamenti programmati	Interattivo	120
Programmare un appuntamento	Interattivo	60
Accettare un appuntamento	Interattivo	60
Accettare sedute	Interattivo	100
Creare corso di aggiornamento	Interattivo	0.1
Registrarsi ad un corso di aggiornamento	Interattivo	0.1
Cercare dati dei medici	Interattivo	200
Iniziare una terapia prolungata	Interattivo	2
Segna appuntamenti saltati	Interattivo	1

Table 2: Frequenza e tipo delle operazioni

Cercare appuntamenti programmati	**Con**	**60000**	
Concetto	**Tipo**	**Accesso**	**Tipo**
Programmato	E	500	R
Cercare appuntamenti programmati	**Senza**	2472000	
Concetto	**Tipo**	**Accesso**	**Tipo**
Appuntamento	E	20600	R

Table 3: Analisi appuntamenti programmati

Segna appuntamenti saltati	**Con**	**504**	
Concetto	**Tipo**	**Accesso**	**Tipo**
Saltato	E	1	W
Programmato	E	500	R
Programmato	E	1	W
Segna appuntamenti saltati	**Senza**	502	
Concetto	**Tipo**	**Accesso**	**Tipo**
Programmato	E	500	R
Programmato	E	1	W

Table 4: Analisi appuntamenti saltati

Per l'analisi delle ridondanze abbiamo cominciato elencando la quantità di istanze di ogni entità e relazione dello schema 1, cercando di utilizzare numeri adeguati ad uno studio moderatamente grande (20 medici) e già in attività da qualche anno (almeno una decina); nella fase di popolazione 6.1

abbiamo cercato, dove possibile, di tenere in considerazione la quantità di istanze prevista in questa fase, per mantenere la validità dell'analisi.

Successivamente abbiamo trovato la frequenza delle operazioni prese dal diagramma dei casi d'uso 2, tenendo in considerazione anche il numero di istanze (ad esempio, più appuntamenti programmati vuol dire più accettazioni) previsto in precedenza. A questo punto ne abbiamo scelto alcune (ne sono state riportate due: 3 4) di queste operazioni legate a delle ridondanze dello schema, e abbiamo calcolato come cambiava il numero di accessi giornalieri alla base di dati con e senza le ridondanze. Ne è risultato che mantenere l'entità ridondante "Programmato" portava ad una notevole riduzione degli accessi alla base di dati, quindi si è deciso di mantenere l'entità; mantenere l'entità "saltato", invece, comportava una riduzione minima, quindi si è scelto di rimuovere la ridondanza.

3.2 Relazionale

Una volta ristrutturato, l'ER è pronto per essere tradotto in relazionale, e quindi le entità e le relazioni dell'ER, fanno posto alle relazioni del relazionale, e subito dopo abbiamo validato la nuova struttura, verificando che il relazionale appena prodotto rispettasse le forme normali.

3.2.1 Traduzione

Per prima cosa abbiamo iniziato identificando le entità principali dell'ER che si sarebbero trasformate in singole relazioni, e che avevamo già identificato nella fase di ristrutturazione come medico, membroPersonaleAusiliario, paziente, terapiaProlungata, ...

Poi ci siamo concentrati sulle relazioni molti a molti (dell'ER), per trasformarle in relazioni (in relazionale), anche queste erano già state identificate nella fase precedente di ristrutturazione; relazioni come: AusiliarioSeduta, MedicoRegistra, Specializzare, ...

Una volta ottenute le relazioni (del relazionale), abbiamo assegnato i vincoli di chiave primaria, secondo i campi di chiave primaria dell'ER, e le chiavi esterne, facendo attenzione a porle nelle relazioni (del relazionale) ottenute da relazioni molti a molti (dell'ER).

- Medico(<u>codice medico</u>, recapito telefonico, indirizzo, cognome, nome, tipo, percentuale incassi (nullable), tariffa oraria (nullable))
- Seduta(<u>data, ora, cf (ext)</u>, ambulatorio)

- MedicoSeduta(codice medico(ext), data(ext), ora(ext), cf (ext))
- AusiliarioSeduta(codice personale(ext), data(ext), ora(ext), cf (ext))
- MembroPersonaleAusiliario(codice personale, nome, cognome, indirizzo, recapito telefonico, tipoDiPersonale)
- Storico(mese)
- AusiliarioRegistra(mese (ext), codice personale (ext), ordinario, straordinario)
- MedicoRegistra(mese (ext), codice medico (ext), ordinario, straordinario)
- CorsoDiAggiornamento(Luogo, denominazione, data)
- Partecipa(CodicePersonale, luogo, denominazione, data (ext))
- Qualifica(TipoDiQualifica)
- Qualificare(codicePersonale, tipoDiQualifica (ext))
- Specializzazione(tipoDiSpecializzazione)
- Specializzare(CodiceMedico(ext), tipoDiSpecializzazione (ext))
- Paziente(Cf, nome, cognome, indirizzo, recapito telefonico, data di nascita, tipo, età (derivato))
- TerapiaProlungata(DataDiInizio, cf (ext), dataDiFine (nullable), tipoDiTerapia, tipoDiSpecializzazione (ext), numeroAppuntamento)
- Appuntamento(data, DataDiInizio (ext), CF (ext), ora) [unique: (data, ora, CF)]
- Programmato(data (ext), DataDiInizio (ext), CF (ext), ora)
- Accettato(data(ext), DataDiInizio (ext), CF (ext), ambulatorio)
- MedicoAppuntamento(codiceMedico, data, DataDiInizio (ext), CF (ext))
- AusiliarioAppuntamento(codicePersonale, data, DataDiInizio (ext), CF (ext))

3.2.2 Validazione e forme normali

Prima forma normale: Lo schema relazionale rispetta la prima forma normale sono stati eliminati in fase di ristrutturazione gli attributi multivalore, e di conseguenza non esistono campi di riga i e colonna j di nessuna delle istanze delle relazioni dello schema che abbiano più di un valore.

Seconda forma normale: Tutti gli attributi non chiave dipendono (direttamente o tramite catene di dipendenze), dalla chiave primaria completa per ogni relazione, qui scriviamo le più articolate:

- Medico: tariffa oraria e percentuale incassi non dipendono da tipo perché per uno stesso tipo di medico, possono esserci diverse percentuali o tariffe orarie, mentre per uno stesso codice medico, ci sono le stesse percentuali incassi e tariffa oraria.
  - codice medico → recapito telefonico
  - codice medico → indirizzo
  - codice medico → nome
  - codice medico → cognome
  - codice medico → tipo
  - codice medico → percentuale incassi
  - codice medico → tariffa oraria
- Seduta:
  - Cf → data
  - Cf → ora
  - Cf → ambulatorio
- MembroPersonaleAusiliario:
  - Codice medico → nome
  - Codice medico → cognome
  - Codice medico → indirizzo
  - Codice medico → recapito telefonico
  - Codice medico → tipo di personale
- AusiliarioRegistra:

  - Mese, codice personale → ordinario
  - Mese, codice personale → straordinario

- Paziente:
  - Cf → nome
  - Cf → cognome
  - Cf → indirizzo
  - Cf → recapito telefonico
  - Cf → data di nascita
  - Cf → tipo
  - Cf → età

- TerapiaProlungata:
  - Data di Inizio, cf → data di fine
  - Data di Inizio, cf → tipo di terapia
  - Data di Inizio, cf → tipo di specializzazione
  - Data di Inizio, cf → numero appuntamenti

In realtà è facile vedere che non ci sono catene di dipendenze, quindi le relazioni saranno anche in terza forma normale.

Terza forma normale: Dall'analisi effettuata è risultato che ogni attributo non chiave dipende direttamente dalla chiave composta, e che non esistono dipendenze transitive, quindi tutte le relazioni analizzate sono in terza forma normale.

Visto che il relazionale rispetta le forme normali, lo riteniamo validato e possiamo passare alla fase successiva di creazione degli indici.

4 Progettazione fisica

4.1 Scelta degli indici

Sono stati implementati degli indici per velocizzare l'esecuzione di alcune query per cui abbiamo ritenuto valesse la pena allocare lo spazio aggiuntivo necessario.

Abbiamo aggiunto degli indici per la ricerca attraverso nome o cognome di pazienti o medici, e per la ricerca della terapia prolungata secondo le stime della frequenza di interrogazione 3

- cf terapiaProlungata → per tutte le query che cercano dati su un paziente in generale (es. tutte le terapie del paziente x);
- terapia appuntamento → per la query "cerca tutti gli appuntamenti di una determinata terapia x";
- codiceMedico medicoAppuntamento → per la query "Tutti gli appuntamenti in cui il medico ha partecipato"

```
1  create index cf_terapiaProlungata on terapiaProlungata (cf)
   ;
2  create index cf_appuntamento on appuntamento (cf);
3  create index cf_programmato on programmato (cf);
4  create index cf_accettato on accettato (cf);
5  create index cf_seduta on seduta (cf);
6  create index terapia_appuntamento on appuntamento (
   dataDiInizio,cf,tipoDiSpecializzazione);
7  create index codiceMedico_medicoAppuntamento on
   medicoAppuntamento (codiceMedico);
8  create index codiceMedico_medicoSeduta on medicoSeduta (
   codiceMedico);
9  create index nome_paziente on paziente (nome);
10 create index nome_medico on medico (nome);
11 create index cognome_paziente on paziente (cognome);
12 create index cognome_medico on medico (cognome);
```

Sono stati considerati anche degli indici secondari sulle tabelle programmato e accettato, ma abbiamo deciso di non metterli perchè nelle fasi precedenti di stima della quantità di queries, queste tabelle risultavano interrogatemolto più raramente.

5 Alcuni Trigger e Query

5.1 Trigger

Nelle fasi precedenti (diagramma casi d'uso, frequenza e tipo operazioni)sono stati nominati diversi trigger, di seguito riportiamo l'implementazione di alcuni triggers esemplificativi.

```
--quando aggiungiamo un appuntamento accettato, la specializzazione richiesta dalla terapia deve essere tra le specializzazioni del medico che fa l'appuntamento

create or replace function controlla_specializzazione()
return trigger
language plpgsql as $$
begin
perform *
from specializzare
where codiceMedico = new.codiceMedico and new.tipoDiSpecializzazione = tipoDiSpecializzazione;
if found then
    return new;
else
    raise exception 'Il medico inserito non ? specializzato per l appuntamento';
    return null;
endif;
end;
$$;

create trigger inserisci_accettato()
before insert on medicoAppuntamento
for each row
execute procedure controlla_specializzazione();
```

```
--quando inseriamo una terapia prolungata, non ci devono essere, per quel paziente, terapie prolungate aperte con lo stesso tipo di specializzazione

create or replace function controlla_specializzazione_terapieProlungate()
return trigger
language plpgsql as $$
begin
perform *
from terapiaProlungata
where new.tipoDiSpecializzazione = tipoDiSpecializzazione and tipoDiTerapia = 'aperta' and new.cf = cf and new.tipoDiTerapia = 'aperto';
if found then
    raise exception 'Non puoi inserite 2 terapie aperte con la stessa specializzazione';
    return null;
else
    return new;
endif;
end;
```

```
$$;

create trigger inserisci_terapiaProlungata()
before insert on terapiaProlungata
for each row
execute procedure
controlla_specializzazione_terapieProlungate();
```

```
--quando inseriamo un nuovo appuntamento nella tabella
appuntamento, se e' un appuntamento futuro lo inseriamo
anche in nella tabella Programmato

create or replace function
check_programma_appuntamento_futuro()
returns trigger
language plpgsql as $$
begin
    if ((new.data > CURRENT_DATE) or (new.data =
CURRENT_DATE and new.ora > extract(hour from CURRENT_TIME))
) then
        insert into programmato (data, cf, dataDiInizio,
tipoDiSpecializzazione, ora)
        values (new.data, new.cf, new.dataDiInizio, new.
tipoDiSpecializzazione, new.ora);
    end if;
    return new;
end;
$$;

create trigger programma_appuntamento_futuro after insert
or update
on appuntamento for each row
execute procedure check_programma_appuntamento_futuro();
```

```
--Verichiamo che solo gli assistenti medici possano
iscriversi ai corsi di aggiornamento

create or replace function
check_non_partecipa_amministrativo()
returns trigger
language plpgsql as $$
begin
    perform * from membroPersonaleAusiliario
        where codicePersonale = new.codicePersonale and
tipo = 'amministrativo';
    if found then
        raise exception 'Solo gli assistenti medici possono
 essere iscritti ai corsi di aggiornamento';
        return null;
```

```
        else
            return new;
        end if;
    end;
    $$;

    create trigger non_partecipa_amministrativo before insert
    or update
    on partecipa for each row
    execute procedure check_non_partecipa_amministrativo();
```

Tutti i trigger implementati sono nel file “relazionale.sql” in allegato

5.2 query

Qui riportiamo alcune delle queries riportate nella tabella di “frequenza e tipo di operazioni” 2 e altre queries più complicate.

```
    -- tutti i medici che hanno visitato il paziente ABCDEF
    select codiceMedico, nome, cognome
    from medico m
    where codiceMedico = any (select codiceMedico
        from medicoSeduta
        where cf = 'ABCDEF')
    or codiceMedico = any (select codiceMedico
        from medicoAppuntamento
        where cf = 'ABCDEF');
```

```
    -- per ogni paziente, i medici che lo hanno visitato piu'
    spesso per problemi occasionali
    create view pazienteSedutaMedico as
    select cf, p1.nome nomePaziente, p1.cognome cognomePaziente
    , m1.codiceMedico codiceMedico, m1.nome nomeMedico, m1.
    cognome cognomeMedico, count(*) sedute
    from ((paziente p1 natural join seduta) natural join
    medicoSeduta) join medico m1 on medicoSeduta.codiceMedico =
     m1.codiceMedico
    group by p1.cf, m1.codiceMedico;

    select *
    from pazienteSedutaMedico s1
    where not exists (select *
        from pazienteSedutaMedico s2
        where s1.cf = s2.cf
        and s1.codiceMedico = s2.codiceMedico
        and s2.sedute > s1.sedute);
```

```
    -- medici che hanno seguito solo ed almeno una terapie che
    richiedevano la specializzazione ABCDEF
```

```
    select codiceMedico, nome, cognome
    from (medico m1 natural join medicoAppuntamento) natural
    join terapiaProlungata t1
    where t1.tipoDiSpecializzazione = 'ABCDEF'
    and not exists (select *
        from (medico m2 natural join medicoAppuntamento)
    natural join terapiaProlungata t2
        where t2.tipoDiSpecializzazione <> 'ABCDEF'
        and m1.codiceMedico = m2. codiceMedico);
```

Tutte le queries sono implementate nelle "query.sql" in allegato

5.3 Viste

Abbiamo aggiunto anche due viste al database: una per aggiungere l'età alla tabella "Paziente" (derivata dalla data di nascita), ed una che aggiunge il numero di appuntamenti accettati alle terapie prolungate. Sono riportate entrambe.

```
create view paziente_eta as
select *, age(datadinascita) eta from paziente;

create view terapia_naccettati as
select dataDiInizio, cf, dataDiFine, tipoDiTerapia,
    tipoDiSpecializzazione, count(*) nAppuntamentiAccettati
from terapiaprolungata natural join accettato
group by dataDiInizio, cf, dataDiFine, tipoDiTerapia,
    tipoDiSpecializzazione;
```

6 Popolazione ed analisi

6.1 Popolazione

Per avere dei dati su cui eseguire l'analisi, quindi provare query ed il comportamento dei vari vincoli e trigger, abbiamo popolato in modo casuale il database utilizzando uno script in R, che si collega al database e, partendo da dei campioni (elenchi casuali di nomi, cognomi, indirizzi, date, etc. recuperati da Internet) genera dei dati casuali da memorizzare sulla base di dati. Per maggiore coerenza con il resto del progetto, abbiamo scelto il numero di righe generate, tenendo conto del numero di istanze ipotizzato per l'analisi delle ridondanze 1, ma (escluse alcune tabelle come "Accettato", il cui numero di righe è casuale) generalmente generandone una quantità maggiore.

Il lavoro si è svolto in due fasi ed anche questa volta è stato diviso tra i membri del gruppo. Nella prima fase abbiamo separato le tabelle corrispondenti alle entità dell'ER, queste tabelle sono state suddivise in due gruppi di simili dimensioni di cui ciascun membro si è occupato individualmente producendo uno script; ci siamo poi incontrati per mettere insieme i due script in uno unico che popolasse tutte le entità. Nella seconda fase abbiamo gestito allo stesso modo le tabelle che rappresentavano le relazioni dell'ER, eventualmente leggendo dal database dati scritti dagli script della prima fase per assicurarsi di rispettare alcuni vincoli di integrità.

Lo script è stato scritto in modo da generare dati che rispettassero i vincoli implementati nella base di dati, questo anche a discapito dell'efficienza, sapendo che anche così i dati vengono generati in pochi secondi ed è sufficiente farlo una volta sola. Tuttavia a volte può generare delle chiavi ripetute per le tabelle "Seduta" e "TerapiaProlungata"; nonostante la quantità di dati generati, questo evento si verifica poco spesso, ed in quel caso è sufficiente cancellare i dati già scritti e riprovare. Visto che si trattava (considetata la frequenza del problema, e l'utilizzo di un tale script) di un problema marginale, abbiamo deciso di non risolverlo ma sottolinearne semplicemente l'eventualità.

6.1.1 Snippets

Riportiamo alcuni snippets interessanti del codice che genera i dati di popolazione, il codice completo dello script è disponibile nel file "popolazione.r".

Generazione sedute

```
## Omessa la generazione di cf (contenente i codici fiscali di
    tutti i pazienti) e cf_terapia (contenente i codici dei
    pazienti in terapia prolungata)

# Seduta
# Questo campionamento e' la causa delle eventuali chiavi
    ripetute
cf_seduta <- sample(cf, 20000, replace=T)
ora <- sample(1:24, 20000, replace=T)
ambulatorio <- sample(LETTERS[1:26], 20000, replace=T)
data_seduta <- sample(dat, 20000, replace=T)

# Controllo che tutti i pazienti siano regolari od occasionali,
    e memorizzo l'informazione su un array
tipo = c()
for(i in 1:length(cf)) {
```

```
    regolare <- cf[i]%in%cf_terapia
    occasionale <- cf[i]%in%cf_seduta
    if(regolare && occasionale) {
        tipo[i] <- "entrambi"
    } else {
        if(regolare) {
            tipo[i] <- "regolare"
        } else {
            if(occasionale) {
                tipo[i] <- "occasionale"
            } else {
                # Se il paziente non e' ancora ne regolare ne occasionale, aggiungo una seduta rendendolo occasionale
                tipo[i] <- "occasionale"
                append(cf_seduta, cf[i])
                append(ora, sample(1:24, 1, replace=T))
                append(ambulatorio, sample(LETTERS[1:26], 1, replace=T))
                append(data_seduta, sample(dat, 1, replace=T))
            }
        }
    }
}
```

Generazione terapie prolungate

```
# Terapie aperte (frame di appoggio per controllare i vincoli)
aperte <- data.frame(matrix(ncol = 2, nrow=0))
colnames(aperte) <- c("cf", "spec")
# Terapia prolungata
cf_terapia <- sample(cf, 5000, replace=T)
tipodispecializzazione_terapia <- sample(tipodispecializzazione, 5000, replace=T)
datadiinizio <- sample(dat, 5000, replace=T)
datadifine <- sample(dat, 5000, replace=T)
tipoditerapia <- c()
for(i in 1:length(datadifine)) {
    # Se la data di fine e' invalida, la si cancella e si mantiene aperta la terapia
    if(datadifine[i] < datadiinizio[i]) {
        # Controlla che non ci sia una terapia aperta dello stesso tipo per lo stesso paziente
        record <- data.frame(cf = c(cf_terapia[i]), spec = c(tipodispecializzazione_terapia[i]))
        if(!duplicated(rbind(aperte, record))[nrow(aperte)+1]) {
            datadifine[i]  <- NA
            tipoditerapia[i] <- "aperta"
            aperte <- rbind(aperte, record)
        } else {
```

```
            # Se non rispetta il vincolo si sceglie una fine valida
            datadifine[i] <- datadiinizio[i] + 1
            tipoditerapia[i] <- "chiusa"
        }
    } else {
        tipoditerapia[i] <- "chiusa"
    }
}
```

Generazione della relazione AusiliarioAppuntamento

```
# Campione di personale
personale <- dbGetQuery(con, "select * from membropersonaleausiliario")

codicepersonale = c()
data_ac = c()
datadiinizio = c()
cf = c()
tipodispecializzazione = c()
for(i in 1:nrow(accettato)) {
    app = accettato[i,]
    # Scelgo quanti membri del personale ausiliario assegnare all'appuntamento
    n = sample(1:3, 1)
    medici <- sample(personale$codicepersonale, n, replace=F)
    codicepersonale <- append(codicepersonale, medici)
    data_ac <- append(data_ac, rep(app$data, n))
    datadiinizio <- append(datadiinizio, rep(app$datadiinizio, n))
    cf <- append(cf, rep(paste(app$cf), n))
    tipodispecializzazione <- append(tipodispecializzazione, rep(paste(app$tipodispecializzazione), n)) #$
}
```

6.2 Analisi e grafici

Sui dati generati casualmente abbiamo effettuato delle analisi utilizzando R, e riportiamo di seguito i grafici ottenuti. Dell'ultimo 8 (in cui rappresentiamo un grafico a linee delle ore cumulative di tutti i medici e tutti gli ausiliari, nel corso dei mesi) abbiamo riportato anche il codice R da cui viene generato. Il codice completo è disponibile nel file "analisi.r"

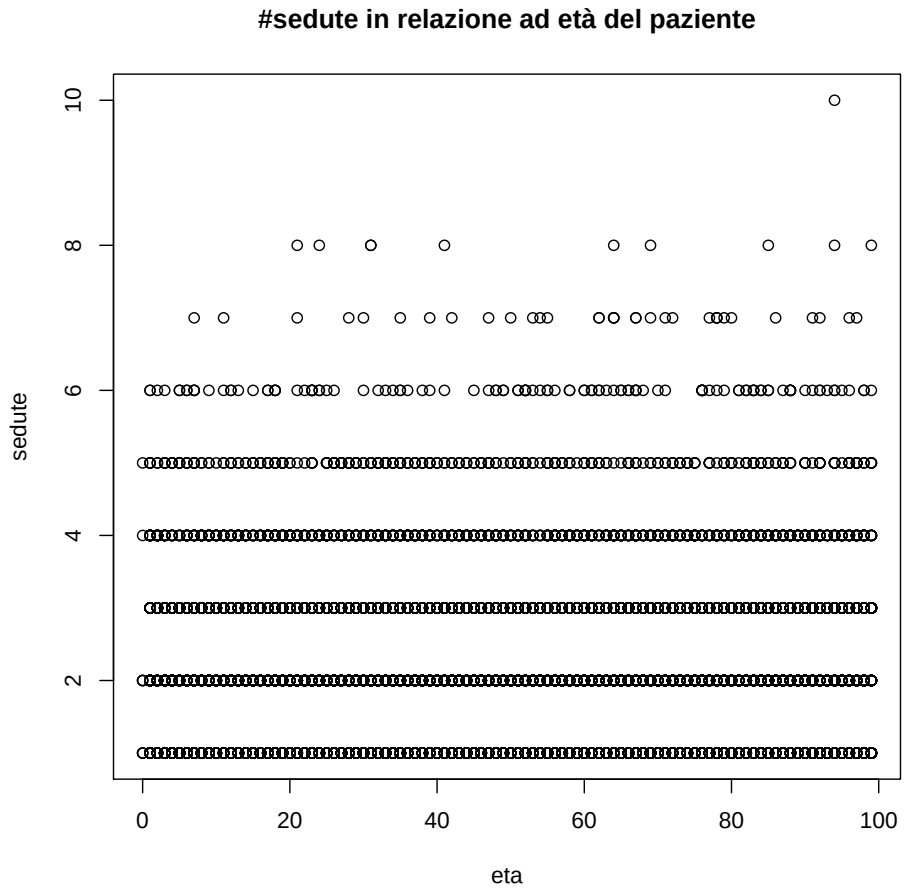

Figure 5: Età vs #sedute

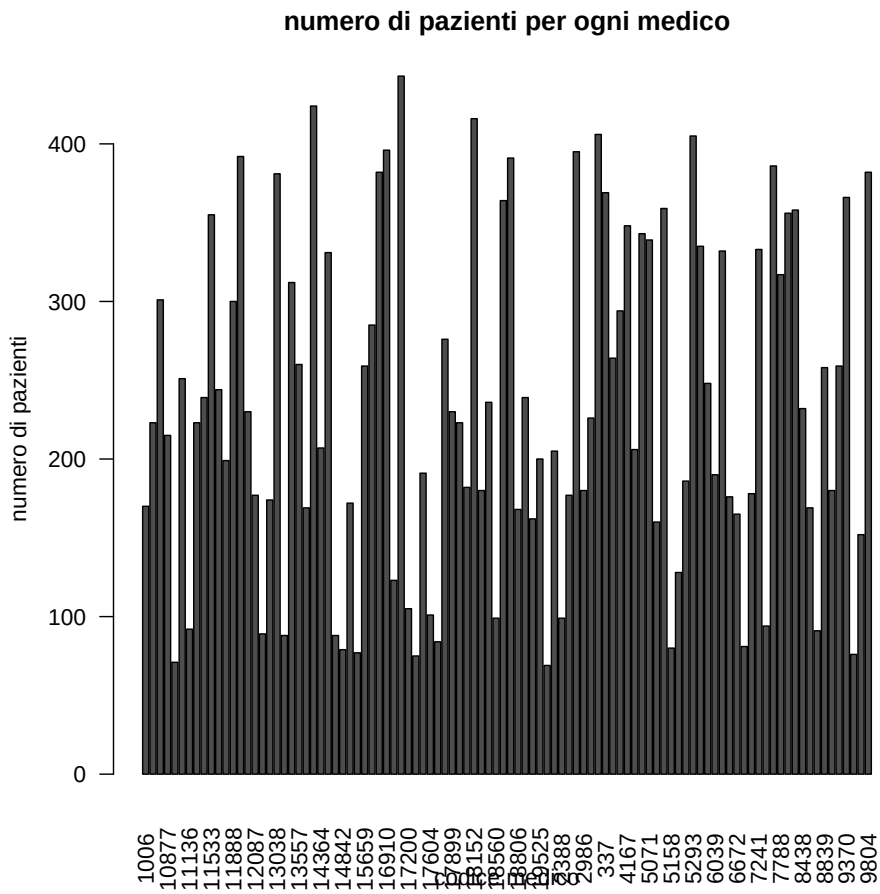

Figure 6: Pazienti per medico

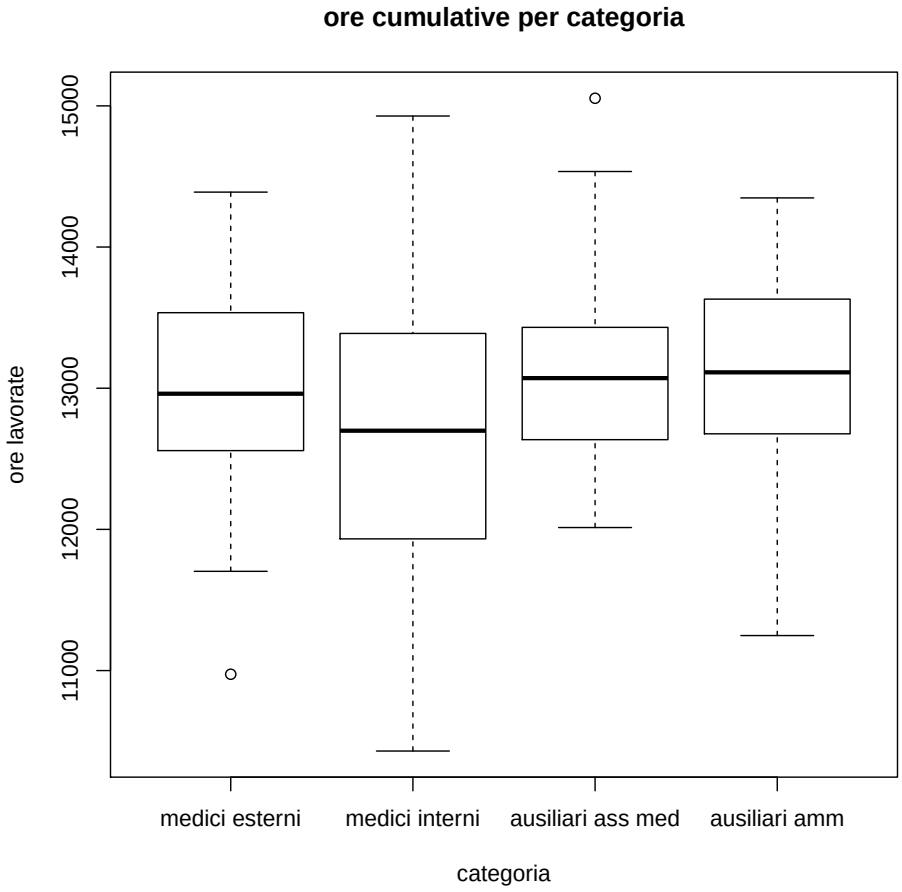

Figure 7: Ore per categoria

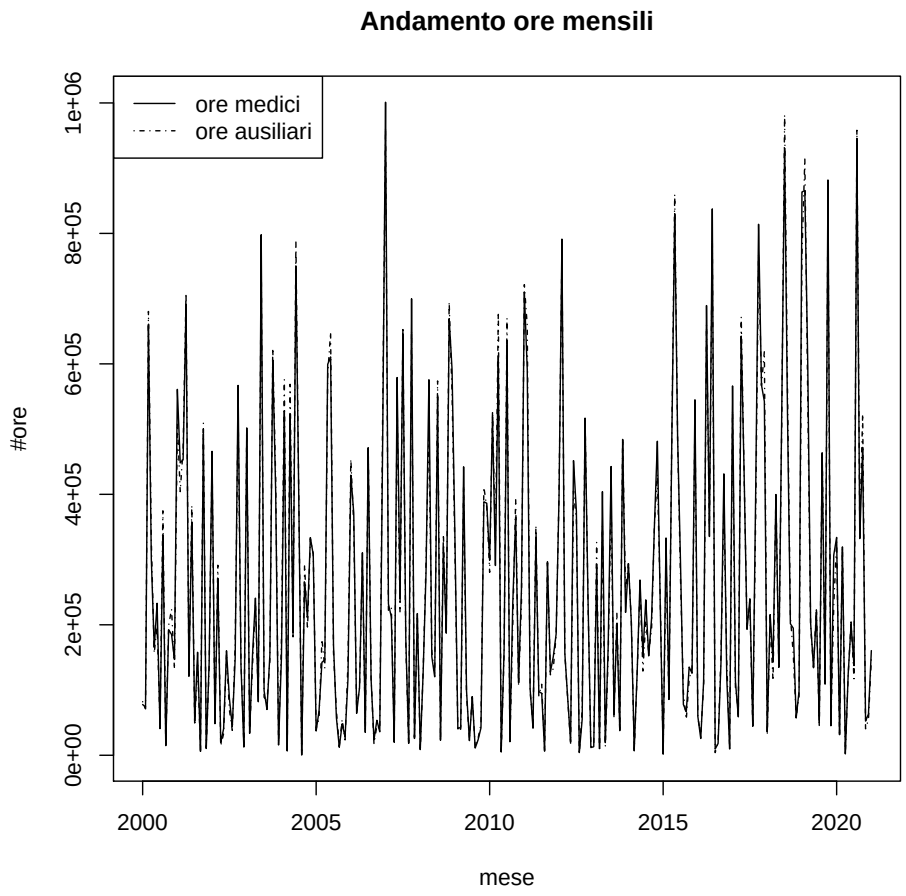

Figure 8: Andamento ore

```
#!/usr/bin/env Rscript
library("RPostgreSQL")
drv <- dbDriver("PostgreSQL")
con <- dbConnect(drv,
  dbname="studiomedico",
  host="localhost",
  port=5432,
  user="postgres",
  password="postgres")
silent <- dbGetQuery(con, "set search_path to studio;")

pdf("andamento_ore.pdf")
```

```
andamento_ore <- dbGetQuery(con, "select
                                    extract(year from s.mese) +
    ((extract(month from s.mese) - 1) / 12) mese,
                                    sum(a.ordinario + a.
    straordinario) ore_ausiliario,
                                    sum(m.ordinario + m.
    straordinario) ore_medico
                                  from (storico as s join
    ausiliarioRegistra as a on s.mese = a.mese) join
    medicoRegistra as m  on s.mese = m.mese
                                  group by s.mese
                                  order by s.mese;")
plot(andamento_ore$mese, andamento_ore$ore_medico, lty = 1,
    type = "l", main = "Andamento ore mensili")
lines(andamento_ore$mese, andamento_ore$ore_ausiliario, lty =
    4, type = "l")
legend("topleft", legend = c("ore medici", "ore ausiliari"),
    lty = c(1, 4))

dev.off()
```

7 Conclusioni

Abbiamo ottenuto un database per la gestione di uno studio medico mediamente grande. Abbiamo implementato e testato diverse query, viste, e trigger su una popolazione casuale di grandi dimensioni, ed abbiamo testato le funzionalità di base, come inserimenti e la coerenza tra le tabelle; ad esempio verificando che i medici assegnati ad un appuntamento possedessero la specializzazione richiesta dalla terapia 5.1.

Il database è quasi completo, tuttavia alcuni vincoli non sono stati implementati, ma solo individuati; per esempio non verifichiamo che il tipo di paziente sia coerente con la presenza di terapie aperte o sedute. Come sviluppo futuro si potrebbero completare i trigger per i vincoli individuati. Possiamo ipotizzare che, in un primo rilascio, il software gestionale da cui il personale dello studio accede al database, permetterebbe solo le operazioni che non causano incoerenze; e man mano che nei rilasci successivi vengono implementati ulteriori vincoli, il gestionale aggiungerà altre funzionalità.