Università di Catania

Basi di Dati A.A.2023/2024

Database per la gestione delle release software di un’azienda

Il progetto si propone di sviluppare un database dedicato alla gestione e all'organizzazione delle release di un’azienda software. Questo database offre un sistema centralizzato per monitorare tutte le versioni dei prodotti software sviluppati dall'azienda. Inoltre, il sistema comprenderà un modulo dedicato alla gestione dei ticket per ciascuna release rilasciata, consentendo un'organizzazione più accurata delle attività correlate allo sviluppo e alla distribuzione dei software.

PATRICK NICOLOSI **X81000928**

1. Progettazione concettuale

1.1. Specifiche sui dati

L'azienda sviluppa vari software, per ognuno di questi verranno presi in considerazione il nome, il sistema operativo ed un ID. Più release, sono associate ad un software e per ognuna di queste è importante considerare la data di rilascio, il codice versione e note di rilascio. Ad ogni versione sono associati una lista di ticket. Ogni ticket può essere di tre tipi Aperto, Chiuso o Non assegnato. Ogni tipologia di ticket presenta un ID, data di apertura, descrizione. Un ticket chiuso contiene un ulteriore informazione che è la data di chiusura. Ogni ticket aperto fa riferimento ad almeno un developer che curerà lo sviluppo e chiusura. Un ticket non assegnato è un ticket aperto ma al quale non è stato ancora assegnato alcun developer. Per ogni developer verranno prese in considerazione nome, cognome, codice fiscale.

Per il nostro progetto supponiamo ci siano 20 software, 30 release per software, 50 ticket per release di cui 10 aperti, 1 non assegnato, 39 chiusi. Ogni ticket aperto è curato da circa 2 developer. L'azienda conta in totale di 10 dipendenti.

1.2. Glossario dei termini

Termini	Descrizione	Sinonimi	Termini collegati
Software	Prodotto dall'azienda.	Applicativo, programma, prodotto	Release
Release	Una versione specifica di un software resa disponibile per l'utilizzo da parte degli utenti finali o dei clienti. Una release può includere nuove funzionalità, correzioni di bug, miglioramenti.	Versione, rilascio	Software, Ticket
Ticket	Richiesta di miglioramento o risoluzione legata ad un particolare aspetto o funzionalità di un software.	Issue, task	Aperto, Chiuso, Non Assegnato, Release
Aperto	Ticket che è stato aperto e assegnato ad un Developer per essere implementato		Developer, Release
Chiuso	Ticket che è stato lavorato e cui sviluppo è terminato.		Release

Non Assegnato	Ticket che è stato aperto ma al quale non è stato associato nessun Developer		Release
Developer	Dipendente dell'azienda	Programmatore, Dipendente, Sviluppatore	Aperto

1.3. **Operazioni sui dati**

Indice	**Operazione**
1	Aggiungere un nuovo software
2	Aggiungere una nuova release
3	Creare un nuovo ticket
4	Assegnare un ticket creato ad un developer
5	Chiudere un issue
6	Visualizzare tutte le release dell'applicativo "Ricevimenti App" ordinate per data
7	Visualizzare l'ultima release di ogni applicativo
8	Visualizzare tutte le issue aperte di una release ordinate per date di apertura
9	Visualizzare tutte le issue chiuse di una release ordinate per date di chiusura
10	Aggiungere un developer
11	Eliminare un developer
12	Visualizzare tutte le issue non assegnate di una release ordinate per date di apertura

1.4. **Raggruppamento e struttura dei requisiti**

- **Frasi di carattere generale:** L'azienda sviluppa vari software, per ognuno di questi verranno presi in considerazione il nome, il sistema operativo di destinazione, ed un ID. Più release, sono associate ad un software e per ognuna di queste è importante considerare la data di rilascio, il codice versione e note di rilascio. Ad ogni versione sono associati una lista di ticket. Ogni ticket avrà uno stato (aperto o chiuso), una descrizione, data di apertura e chiusura ticket. Ogni ticket dovrà fare riferimento ad un developer che curerà lo sviluppo e chiusura. Per ogni developer verranno prese in considerazione nome, cognome, codice fiscale.
- **Frasi relative ai software:** L'azienda sviluppa vari software, per ognuno di questi verranno presi in considerazione il nome, il sistema operativo di destinazione, ed un ID. Più release, facendo riferimento all'ID, sono associate ad un software. Consideriamo circa 20 software.
- **Frasi relative alle release:** Più release, sono associate ad un software e per ognuna di queste è importante considerare la data di rilascio, il codice versione e note di rilascio. Ad ogni versione sono associati una lista di ticket. Consideriamo circa 30 release per software.

- **Frasi relative ai ticket**: Ad ogni versione sono associati una lista di ticket. Ogni ticket può essere di tre tipi Aperto, Chiuso o Non assegnato. Ogni tipologia di ticket presenta un ID, data di apertura, descrizione. Consideriamo circa 50 ticket per release.
- **Frasi relative ai ticket aperti**: Ogni ticket aperto è curato da 2 developer. Consideriamo 10 ticket aperti per release.
- **Frasi relative ai ticket chiusi**: Un ticket chiuso contiene un ulteriore informazione che è la data di chiusura. Consideriamo 39 ticket chiusi per release.
- **Frasi relative ai ticket non assegnati:** Un ticket non assegnato è un ticket aperto ma al quale non è stato ancora assegnato alcun developer. Consideriamo 1 ticket di questo tipo per release.
- **Frasi relative ai developer:** Per ogni developer verranno prese in considerazione nome, cognome, codice fiscale. Consideriamo circa 10 dipendenti. Ogni ticket aperto è curato da 2 developer.

1.5. Strategie di progetto e schema concettuale

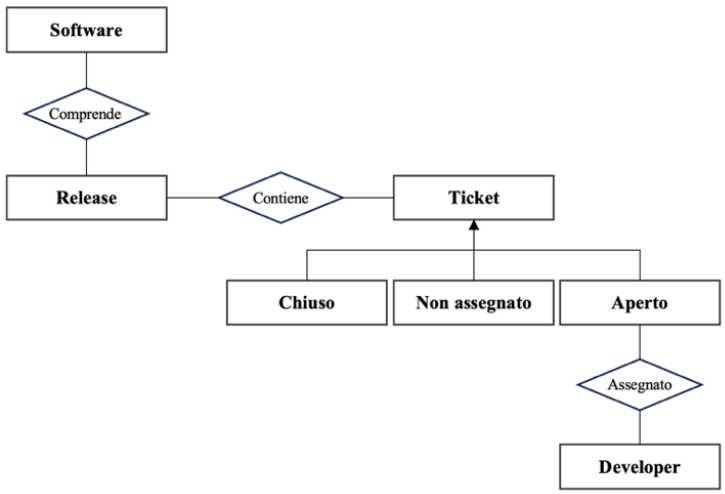
La strategia di progetto scelta per lo svolgimento del progetto è la top-down, attraverso la quale lo schema concettuale viene prodotto mediante una serie di raffinamenti effettuati su uno schema iniziale che descrive tutte le specifiche con pochi concetti molto astratti. I raffinamenti avranno lo scopo di aumentare il dettaglio dei concetti inizialmente astratti.

1.5.1. Generalizzazioni

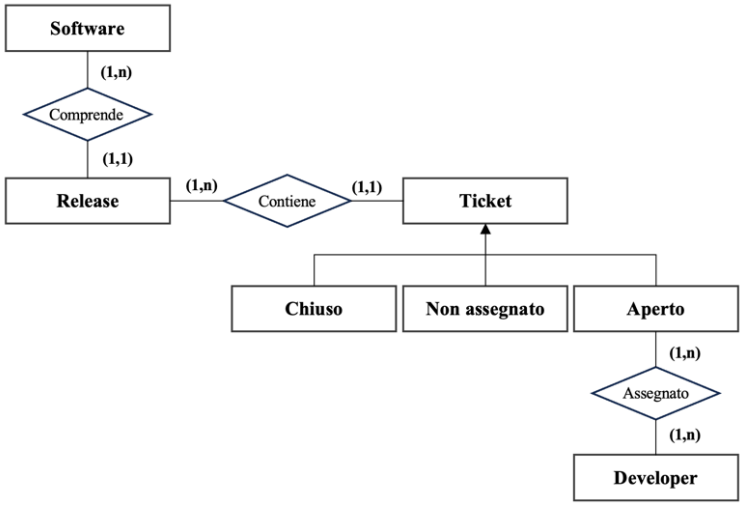
L'unica generalizzazione può essere fatta sulla tipologia del Ticket, infatti le tipologie Aperto, Chiuso, Non assegnato possono essere viste come entità figlie di Ticket. La generalizzazione è, in questo caso, totale poiché ogni occorrenza di *Ticket* è occorrenza di almeno una delle entità figlie, ed esclusiva poiché ogni occorrenza di *Ticket* è occorrenza di al più una delle entità figlie.

1.5.2. Schema concettuale

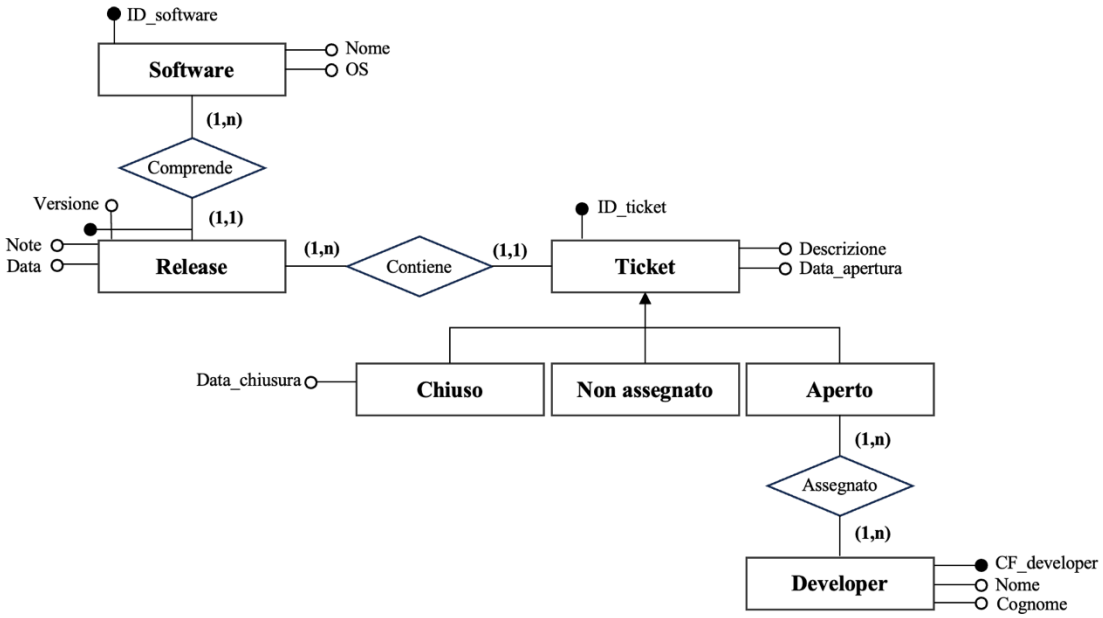
Di seguito è riportato uno schema scheletro semplificato che illustra le entità e le relazioni tra i dati derivati dalla descrizione generale (2.4)



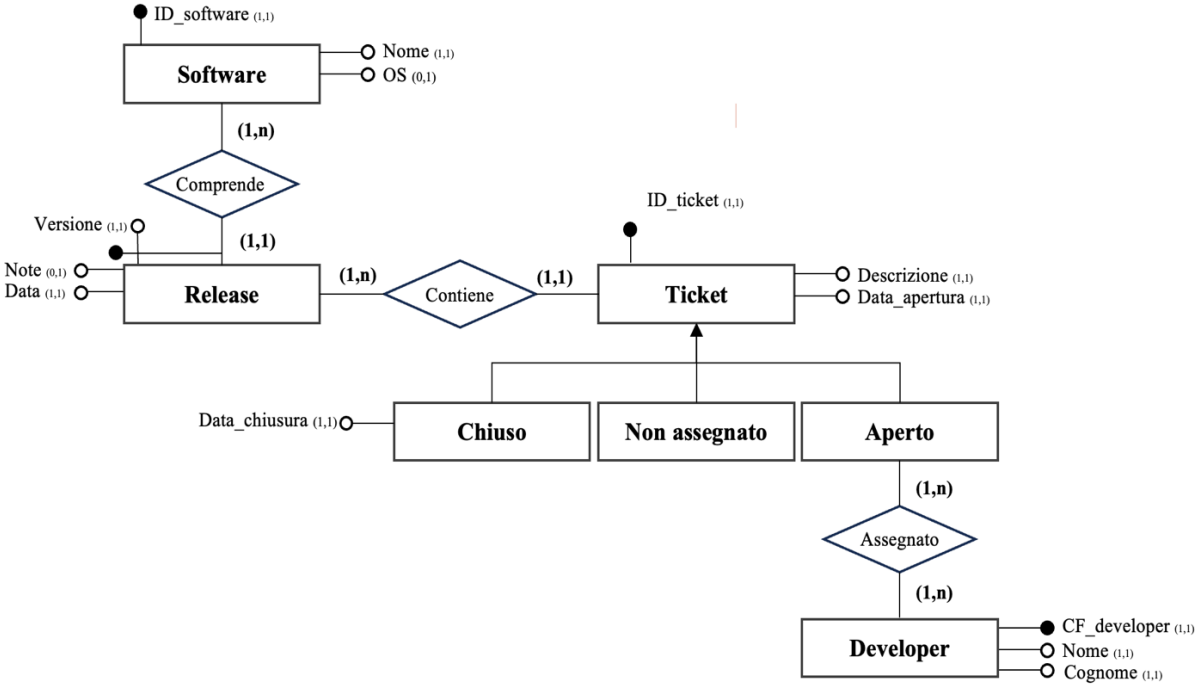
Un primo raffinamento è legato all'aggiunta delle cardinalità relative alle relazioni *Assegnato*, *Contiene*, *Comprende*.



Un secondo raffinamento è legato all'aggiunta di attributi e chiavi per ogni entità.



Attraverso un terzo raffinamento di aggiunta cardinalità degli attributi otteniamo lo schema scheletro completo.



1.6. Vincoli non esprimibili dallo schema e dati derivabili

1.6.1. Vincoli non esprimibili nel diagramma

Rispetto il diagramma concettuale non esiste nessun vincolo non esprimibile.

1.6.2. Dati derivabili nel diagramma

Rispetto il diagramma concettuale non esiste nessun dato derivabile.

1.7. Dizionario dei dati

Entità	Descrizione	Attributi	Identificatori
Software	Prodotto dall'azienda.	ID_software, Nome, OS	ID_software
Release	Una versione specifica di un software resa disponibile per l'utilizzo da parte degli utenti finali o dei clienti. Una release può includere nuove funzionalità, correzioni di bug, miglioramenti.	Note, Data, Versione	Versione, ID_software

Ticket	Richiesta di miglioramento o risoluzione legata ad un particolare aspetto o funzionalità di un software.	ID_ticket, Descrizione, Data_apertura	ID_ticket
Aperto	Ticket che è stato aperto e assegnato ad un Developer per essere implementato	ID_ticket, Descrizione, Data_apertura	ID_ticket
Chiuso	Ticket che è stato lavorato e cui sviluppo è terminato.	ID_ticket, Descrizione, Data_apertura, Data_chiusura	ID_ticket
Non assegnato	Ticket che è stato aperto ma al quale non è stato associato nessun Developer	ID_ticket, Descrizione, Data_apertura	ID_ticket
Developer	Dipendente dell'azienda	CF_developer, Nome, Cognome	CF_developer

1.8. **Dizionari delle relazioni**

Associazione	**Partecipanti**	**Descrizione**	**Attributi**
Comprende	Software, Release	Ogni software *Comprende* diverse release ed almeno una (1,n) ma ad ogni release corrisponde un software (1,1).	
Contiene	Release, Ticket	Ogni Release *Contiene* più Ticket ed ogni Ticket fa riferimento ad una Release.	
Assegnato	Aperto, Developer	Ogni ticket Aperto è assegnato ad uno o più developer (1,n). Ogni developer ha assegnato più ticket.	

2. Progettazione logica

2.1. **Stime**

Per il nostro progetto supponiamo ci siano 20 software, 30 release per software, 50 ticket per release di cui 10 aperti, 1 non assegnato, 39 chiusi e 10 dipendenti. Ogni ticket aperto è curato da circa 2 developer.

2.2. **Tabella dei volumi**

Concetto	**Tipo**	**Volume**
Software	E	**20**
Release	E	Considerato 30 release per software il volume totale è **600**
Ticket	E	Considerando 50 ticket per release il volume totale è 600*50=**30000**
Aperto	E	Considerando 10 ticket aperti per release il volume totale è 600*10=**6000**
Chiuso	E	Considerando 39 ticket chiusi per release il volume totale è 600*39=**23400**
Non Assegnato	E	Considerando 1 ticket non assegnato per release il volume totale è 600*1=**600**
Developer	E	**10**
Comprende	R	**600**
Contiene	R	**30000**
Assegnato	R	Considerando 2 developer a ticket il volume è **12000**

2.3. **Tabella delle frequenze**

Indice	**Tipo**	**Frequenza**
Op. 1	I	**1/Anno**
Op. 2	I	Vengono rilasciate per ogni software una release al mese, considerando 20 software, abbiamo una frequenza di **20/Mese**
Op. 3	I	Vengono creati per ogni software 3 ticket al mese, considerando 20 software, abbiamo una frequenza di **60/Mese**

Op. 4	I	In media ogni ticket aperto viene subito assegnato, quindi **60/Mese**
Op. 5	I	In media tutte le issue aperte nel mese vengono chiuse, quindi **60/Mese**
Op. 6	I	Una volta al mese per software, quindi **20/Mese**
Op. 7	I	10 volte al mese per software, quindi **200/Mese**
Op. 8	I	1 volta al giorno per software, quindi **600/Mese**
Op. 9	I	**200/Mese**
Op. 10	I	**1/Anno**
Op. 11	I	**1/Anno**
Op. 12	I	Una volta al mese per software, quindi **20/Mese**

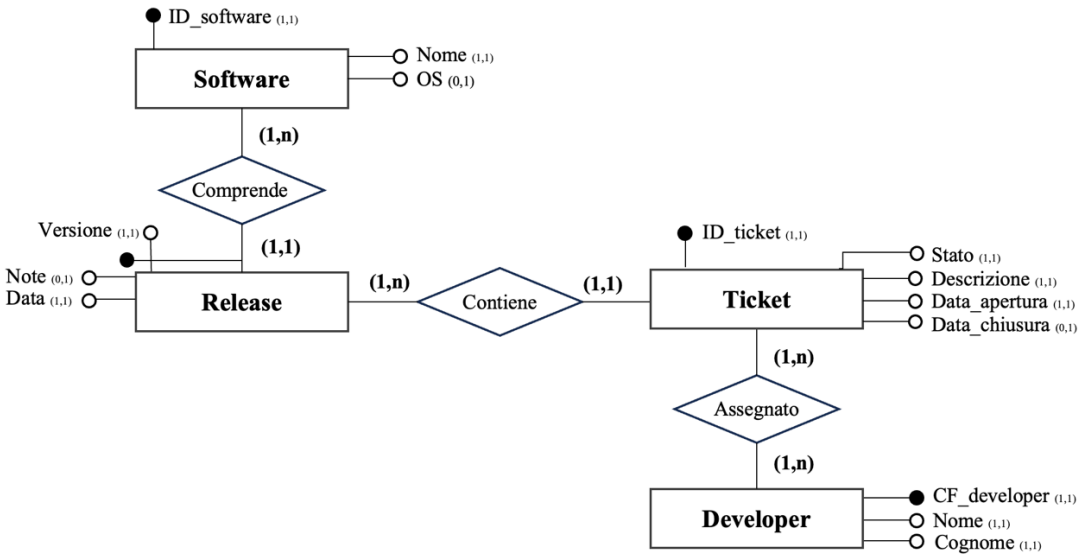
2.4. **Analisi delle ridondanze**

Non ci sono ridondanze nello schema concettuale.

2.5. **Accorpamento delle entità**

L'obiettivo è quello di eliminare le generalizzazioni presenti nello schema. L'unica generalizzazione è quella composta dalle entità *Ticket→Aperto, Chiuso, Non Assegnato*. In questo caso, operiamo secondo un accorpamento delle entità figlie della generalizzazione dell'entità padre, motivo per cui aggiungiamo un attributo Stato, necessario a distinguere il tipo di ticket (Aperto, Chiuso o Non assegnato) e data di chiusura.

2.6. **Schema concettuale ristrutturato**



Conseguenza della ristrutturazione dello schema E/R è l'introduzione di un vincolo non esprimibile con il diagramma. Infatti, Stato ha come vincolo quello di dover assumere uno dei seguenti valori *Aperto*, *Chiuso* o *Non assegnato*. Un'altra conseguenza è quella di aver introdotto una ridondanza con l'attributo Stato in *Ticket*. Infatti, notiamo che:

- Un ticket aperto non è che un ticket che ha una data di apertura, non ha data di chiusura ed ha un numero di developer assegnati maggiore di 0.
- Un ticket chiuso è un ticket che ha una data di chiusura.
- Un ticket è non assegnato se ha una data di apertura, non ha data di chiusura ed ha un numero di developer assegnati pari a 0.

Lo stato è per cui derivabile da altri dati.

2.7. **Analisi delle ridondanze dello schema ristrutturato**

Procediamo con l'analisi delle ridondanze, effettuate secondo la seguente equivalenza: $1S = 2L$. Nello specifico verranno analizzate le operazioni 3, 4, 5, 8, 9, 11, 12. Le uniche a far uso del dato ridondante *Stato* della entità *Ticket*.

Per l'operazione 3.

	Con ridondanza	**Senza ridondanza**
Lista operazioni	1. Scrivo in *Ticket* (1S)	1. Scrivo in *Ticket* (1S)
Costo totale	2L * 60 /Mese, quindi **120L /Mese**	2L * 60 /Mese, quindi **120L /Mese**

Per l'operazione 4.

	Con ridondanza	**Senza ridondanza**
Lista operazioni	1. Scrivo in *Ticket* per aggiornare lo stato da "Non assegnato" ad "Assegnato" (1S) 2. Assegnare due developer al ticket (1S*2)	1. Assegnare due developer al ticket (1S*2)
Costo totale	(1S+1S+1S) * 60 /Mese = (2L+2L+2L) * 60 /Mese, quindi **360L /Mese**	(1S+1S) * 60 /Mese = 4L * 60 /Mese, quindi **240L /Mese**

Per l'operazione 5.

	Con ridondanza	**Senza ridondanza**
Lista operazioni	1. Scrivo in *Ticket* per aggiungere la data di chiusura (1S)	1. Scrivo in *Ticket* per aggiungere la data di chiusura (1S)

	2. Scrivo in *Ticket* per aggiornare lo stato (1S) 3. Rimuovo i 2 developer assegnati (1S*2)	2. Rimuovo tutti i developer assegnati (1S*2)
Costo totale	(1S+1S+1S+1S) * 60 /Mese = (2L+2L+2L+2L) * 60 /Mese, quindi **480L /Mese**	(1S+1S+1S) * 60 /Mese = (2L+2L+2L) * 60 /Mese, quindi **360L /Mese**

Per l'operazione 8.

	Con ridondanza	**Senza ridondanza**
Lista operazioni	1. Leggo da *Ticket* le issue di una release che hanno come stato "Aperto". Queste saranno circa 10. (1L*10)	1. Leggo da *Ticket* le issue di una release che non hanno data di chiusura. Queste saranno circa 11 perché verranno considerate la somma tra le issue aperte (10) e quelle non assegnate (1). (1L*11) 2. Contiamo il numero di developer assegnati. Se questo è positivo, allora significa che la specifica release è aperta. Per ogni Ticket della release che non ha data di chiusura, abbiamo in media due developer assegnati. (11L * 2) Il conteggio deve essere effettuato obbligatoriamente perché non è detto, senza ridondanza, che un ticket che non presenta data di chiusura sia aperto, infatti può anche essere non assegnato.
Costo totale	10L* 600/Mese, quindi **6000L /Mese**	(1L*11)+(11L*2) = (11L+22L) * 600/Mese, quindi **19800L /Mese**.

Per l'operazione 9.

	Con ridondanza	**Senza ridondanza**
Lista operazioni	1. Leggo da *Ticket* le issue di una release che	1. Leggo da *Ticket* le issue che hanno indicato la

	hanno come stato “Chiuso” (circa 39) (1L*39)	data di chiusura (circa 39) (1L*39)
Costo totale	39L * 200 /Mese, quindi **7800L /Mese**	39L * 200/Mese, quindi **7800L/ Mese**

Per l’operazione 12.

	Con ridondanza	**Senza ridondanza**
Lista operazioni	1. Leggo da *Ticket* le issue di una release che hanno come stato “Non assegnato” (1L)	1. Leggo da *Ticket* le issue di una release che non hanno data di chiusura. Queste saranno circa 11 perché verranno considerate la somma tra le issue aperte (10) e quelle non assegnate (1). (1L*11) 2. Contiamo il numero di developer assegnati. Se questo è nullo, allora significa che la specifica release è aperta. Per ogni Ticket della release che non ha data di chiusura, abbiamo in media due developer assegnati. (11L * 2)
Costo totale	1L * 20 /Mese, quindi **20L /Mese**	(11L+22L) * 20/Mese, quindi **660L /Mese**

Con ridondanza

120L+360L+480L+6000L+7800L+20L=14780L /Mese

Senza ridondanza

120L+240L+360L+19800L+7800L+660L=28980L /Mese

Si deduce che conviene mantenere la ridondanza.

Nel calcolo abbiamo escluso l’operazione 11 perché troppo poco frequente rispetto le altre e quindi non influente per nell’analisi delle ridondanze.

2.8. **Schema logico**

Iniziamo la traduzione di entità ed associazioni per arrivare alla formazione dello schema logico.

- **Associazioni uno a molti**
  - Creiamo una "tabella" *Release* aggiungendo tutti gli attributi della relazione *Release* e la chiave ID_software della relazione *Software*.
  - Creiamo una "tabella" *Ticket* aggiungendo tutti gli attributi della relazione *Ticket* e la chiave della relazione *Release*.
- **Associazioni molti a molti**
  - Creiamo una "tabella" *Ticket_Assegnati* aggiungendo la chiave di *Ticket* e la chiave di *Developer*. Questa tabella andrà a tradurre la relazione molti a molti *Assegnato*.

Lo schema logico finale sarà

Software(**ID_software**, Nome, OS)

Release(**ID_software, Versione**, Note, Data)

Ticket (**ID_ticket,** ID_software, Versione, Stato, Descrizione, Data_apertura, Data_chiusura)

Ticket_Assegnato (**CF_Developer, ID_ticket**)

Developer(**CF_developer**, Nome, Cognome)

In grassetto le chiavi delle relazioni e sottolineate le chiavi esterne.

3. Progettazione fisica

In conclusione, procediamo con la traduzione in linguaggio SQL di:

- Tabelle, definite secondo il precedente schema logico e presenti nel file *tables.sql*
- Dati, necessari al testing e presenti nel file *datas.sql*
- Trigger, presenti nel file *triggers.sql*
- Operazioni, presenti nel file *operations.sql*