



Relazione progetto di Laboratorio

Corso di Basi di Dati

Università degli studi di Udine, A.A. 2024-2025

Daniele De Martin [162521@spes.uniud.it]

Massimiliano Di Marco [144714@spes.uniud.it]

Enrico Peressin [163503@spes.uniud.it]

Michele Vecchiato [162101@spes.uniud.it]

PROGETTAZIONE E IMPLEMENTAZIONE DI UNA BASE DI DATI PER LA GESTIONE DI UNA BANCA

Indice

1. Raccolta e analisi dei requisiti	4
1.1. Richiesta Originale	4
1.2. Analisi dei Requisiti	4
1.2.1. Assunzioni	4
1.2.2. Glossario	5
2. Progettazione Concettuale	6
2.1. Costruzione dello schema Entità-Relazione	6
2.1.1. Entità	6
2.1.2. Relazioni	8
2.2. Scelte particolari	10
2.3. Schema Concettuale	11
2.4. Analisi dei cicli	11
2.4.1. Ciclo <i>DIPENDENTE - FILIALE - CAPO</i> :	11
2.4.2. Ciclo <i>DIPENDENTE - CLIENTE - CONTO - FILIALE</i> :	12
2.5. Vincoli d'integrità	12
3. Progettazione Logica	13
3.1. Tabella dei volumi	13
3.1.1. Considerazioni	13
3.2. Analisi delle ridondanze	13
3.2.1. Studio dell'attributo derivato <i>Attivi di FILIALE</i>	13
Operazione 1	14
Operazione 2	14
Operazione 3	15
Operazione 4	16
3.2.2. Studio dell'attributo derivato <i>Somma rate di PRESTITO</i>	17
Operazione 1	17
Operazione 2	18
3.3. Selezione delle chiavi primarie	19
3.4. Rimozione delle specializzazioni	19
3.5. Schema ER ristrutturato	20
3.6. Schema Logico	20
3.6.1. Chiavi esterne	21
4. Popolamento del database	22
4.1. Creazione delle tabelle	22
4.2. Modalità di generazione dei dati	22
4.2.1. Dati <i>FILIALE</i>	22
4.2.2. Dati <i>DIPENDENTE</i>	22
4.2.3. Dati <i>CLIENTE</i>	23
4.2.4. Dati <i>CONTO</i>	23
4.2.5. Dati <i>POSSIEDE</i>	23
4.3. Creazione dei trigger	23
4.3.1. Trigger <i>FILIALE-DIPENDENTE</i>	23
4.3.2. Trigger <i>FILIALE-CONTO-PRESTITO-RATA</i>	23
4.3.3. Trigger <i>POSSIEDE-CONTO-FILIALE</i>	24
4.4. Inserimento tabelle e dati nel database	24

4.5. Test	24
4.6. Test Dipendente-Filiale	24
4.7. Test Prestito-Rata	25
4.8. Test Conto-Filiale	25
5. Query	26
5.1. Query 1	26
5.2. Query 2:	26
5.3. Query 3:	27
5.4. Query 4:	27
5.5. Query 5:	28
6. Analisi dei dati	30
6.1. Distribuzione dei prestiti per mensilità	30
6.2. Analisi attivi per anzianità gestori	31
6.3. Analisi conti cointestati	32
7. Conclusioni	33

1. Raccolta e analisi dei requisiti

1.1. Richiesta Originale

Si vuole progettare una base di dati di supporto ad alcune delle attività di una banca.

La banca è organizzata in un certo numero di filiali. Ogni filiale si trova in una determinata città ed è identificata univocamente da un nome (si noti che in una città vi possono essere più filiali). La banca tiene traccia dei risultati (attivi) conseguiti da ciascuna filiale.

Ai clienti della banca è assegnato un codice che li identifica univocamente. La banca tiene traccia del nome del cliente e della sua residenza. I clienti possono possedere uno o più conti e possono chiedere prestiti. A un cliente può essere associato un particolare dipendente della banca, che segue personalmente tutte le pratiche del cliente (si tenga presente che non tutti i clienti godono di tale privilegio e che ad un dipendente della banca possono essere associati zero, uno o più clienti).

I dipendenti della banca sono identificati da un codice. La banca memorizza nome e recapito telefonico di ogni dipendente, il nome delle persone a suo carico e il codice dell'eventuale capo. La banca tiene inoltre traccia della data di assunzione di ciascun dipendente e dell'anzianità aziendale di ciascun dipendente (da quanto tempo tale dipendente lavora per la banca).

La banca offre due tipi di conto: conto corrente (con la possibilità di emettere assegni, ma senza interessi) e conto di risparmio (senza la possibilità di emettere assegni, ma con interessi). Un conto può essere posseduto congiuntamente da più clienti e un cliente può possedere più conti. Ogni conto è caratterizzato da un numero che lo identifica univocamente. Per ogni conto, la banca tiene traccia del saldo corrente e della data dell'ultima operazione eseguita da ciascuno dei possessori (un'operazione può essere eseguita congiuntamente da più possessori). Ogni conto di risparmio è caratterizzato da un tasso di interesse, mentre ogni conto corrente è caratterizzato da uno scoperto accordato al cliente.

Un prestito (ad esempio, un mutuo) viene emesso da una specifica filiale e può essere attribuito a uno o più clienti congiuntamente. Ogni prestito è identificato univocamente da un codice numerico. Ogni prestito è caratterizzato da un ammontare e da un insieme di rate per la restituzione del prestito. Ogni rata di un dato prestito è contraddistinta da un numero d'ordine (prima rata, seconda rata...). Di ogni rata vengono memorizzati anche la data e l'ammontare.

1.2. Analisi dei Requisiti

1.2.1. Assunzioni

Al fine di proseguire con la progettazione concettuale, sono state effettuate le seguenti assunzioni:

- Gli **attivi** sono la somma della liquidità dei conti meno la somma delle rate non pagate dei prestiti erogati. Sono relativi alla singola filiale.
- Un **cliente** può avere conti in filiali diverse e ogni conto è associato ad una singola filiale.
- I **prestiti** sono legati al conto, non al cliente.

- Un **dipendente** non può gestire se stesso.
- Un **dipendente** lavora in una sola filiale con la possibilità di gestire clienti al di fuori della propria filiale.
- Il **capo** di un dipendente è l'unico responsabile della filiale in cui il dipendente lavora.
- Nei **conti cointestati** i clienti non possono essere seguiti da dipendenti (gestori) diversi.
- In caso di **ri-assunzione** di un dipendente, si tiene conto solo dell'ultima assunzione per il calcolo dell'anzianità.
- Tutte le **rate** di un determinato prestito hanno lo stesso ammontare e devono essere pagate in ordine.

1.2.2. Glossario

Per chiarire il significato dei termini chiave definite nei requisiti viene fornito un glossario esplicativo:

Termine	Descrizione
Filiale	Unità operativa della banca situata in una determinata città. È gestita da un unico capo.
Cliente	Persona fisica con almeno un conto aperto nella banca.
Conto	Servizio di gestione del denaro che permette diverse operazioni. Può essere esclusivamente di tipo corrente o di risparmio.
Conto corrente	Tipo di conto caratterizzato da uno scoperto.
Conto di risparmio	Tipo di conto caratterizzato da un tasso di interesse.
Dipendente	Persona fisica che lavora in una certa filiale della banca.
Gestore	Dipendente che prende in carico le pratiche di uno o più clienti.
Operazione	Transazioni bancarie effettuate su un conto da uno o più intestatari. Sono operazioni l'apertura di un conto, il prelievo, il pagamento elettronico (bancomat) e il versamento.
Prestito	Somma di denaro concessa dalla banca a un cliente.

Tabella 1: *Glossario dei termini chiave*

2. Progettazione Concettuale

2.1. Costruzione dello schema Entità-Relazione

L'analisi dei requisiti ha portato alla definizione di un insieme di entità e relazioni che costituiranno il modello concettuale della base di dati.

2.1.1. Entità

- L'entità **FILIALE** rappresenta un'unità operativa della banca situata in una determinata città. La chiave primaria è il *Nome*, mentre gli altri attributi sono *Città* e *Indirizzo*. Inoltre, per ogni filiale è presente l'attributo derivato *Attivi*, che rappresenta l'ammontare totale della liquidità della filiale e viene calcolato sulla base dei conti, prestiti e rate ad esso associati.

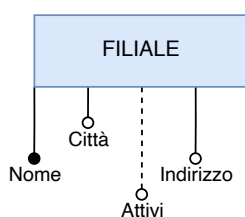


Figura 1: Entità *FILIALE*

- L'entità **CLIENTE** rappresenta una persona fisica che ha aperto nella banca almeno un conto. Essa è caratterizzata da un codice univoco (*ID*) assegnato dalla banca ad ogni cliente e dal *Codice Fiscale*, entrambi questi attributi possono essere due chiavi primarie differenti in quanto sono uniche per ogni cliente. Gli altri attributi servono per tenere traccia dell'anagrafica del cliente: *Nome*, *Cognome*, *Telefono*, *Data di nascita* e *Residenza*.

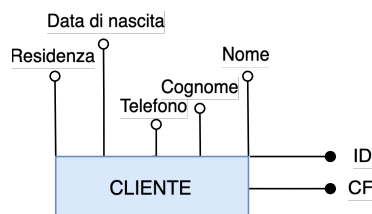


Figura 2: Entità *CLIENTE*

- L'entità **DIPENDENTE** è caratterizzata da un codice univoco *ID* che funge da chiave primaria. *Nome*, *Cognome*, *Numero di telefono*, *Data di assunzione* sono gli altri attributi che la descrivono. È stato scelto di tenere traccia dell'anzianità aziendale sulla base della data di assunzione. La qualifica di capo viene descritta da una specializzazione parziale di *DIPENDENTE*, chiamata **CAPO**.

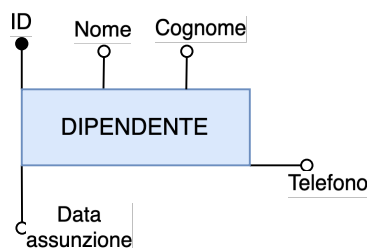


Figura 3: Entità *DIPENDENTE*

- L'entità **CAPO** rappresenta il capo di una filiale. Essendo una specializzazione dell'entità **DIPENDENTE**, eredita tutti gli attributi di quest'ultima. Un capo è unico per ogni filiale.

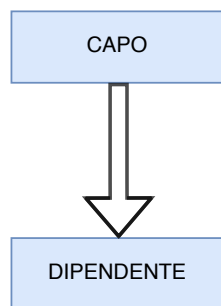


Figura 4: Entità **CAPO**

- L'entità **CONTO** serve per identificare un servizio della banca messo a disposizione per il cliente. Ogni entità viene identificata univocamente da un attributo *IBAN*, un attributo *Saldo* tiene traccia dell'ammontare di denaro presente sul conto. La banca inoltre mette a disposizione due tipi di conto, quindi **CONTO** è stato specializzato in due sottoentità: **CONTO CORRENTE** e **CONTO DI RISPARMIO**. La specializzazione è totale e disgiunta.
 - L'entità **CONTO CORRENTE** è una specializzazione dell'entità **CONTO** pertanto ne eredita tutti gli attributi e tutte le relazioni, la chiave primaria è quindi quella dell'entità **CONTO**. L'attributo che lo caratterizza è *Scoperto* che indica quanto la banca può concedere di debito nei confronti del cliente.
 - L'entità **CONTO DI RISPARMIO** è una specializzazione dell'entità **CONTO** pertanto ne eredita tutti gli attributi e tutte le relazioni, la chiave primaria è quindi quella dell'entità di **CONTO**. L'attributo che lo caratterizza è *Tasso d'interesse* che indica il valore di rendita mensile del conto.

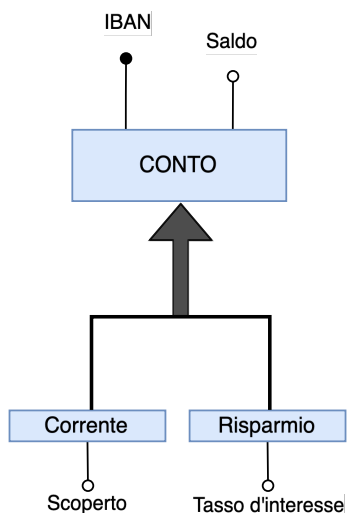


Figura 5: Entità **CONTO**

- L'entità **PRESTITO** costituisce il servizio creditizio della banca. Essa è caratterizzata da un codice univoco che funge da chiave primaria. L'attributo *Ammontare* fornisce l'informazione relativa alla somma di denaro prestata, mentre l'attributo *Inizio* registra la data in cui il prestito ha avuto origine. L'attributo *Somma rate* è un attributo derivato, che tiene traccia dell'importo saldato dal cliente. L'attributo *Mensilità* indica il numero di rate complessive del prestito.

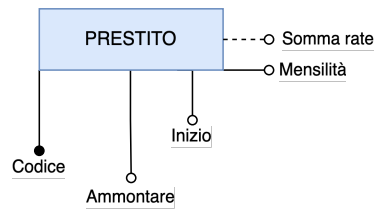


Figura 6: Entità *PRESTITO*

- L'entità **RATA** è un'entità debole che ha il compito di rappresentare ogni singolo pagamento periodico associato a un determinato prestito. L'identificazione univoca di ciascuna rata è garantita da una chiave primaria composta, costituita dal suo numero (indicante la “posizione” della rata nella sequenza dei pagamenti) e dalla chiave esterna che fa riferimento all'entità *PRESTITO*. Tra gli attributi figurano inoltre la *Data scadenza*, ossia il giorno entro cui la rata deve essere corrisposta, e la *Data pagamento*, che riporta il momento in cui il versamento è stato effettuato. Infine, l'attributo *Ammontare* specifica l'importo dovuto per quella singola rata.

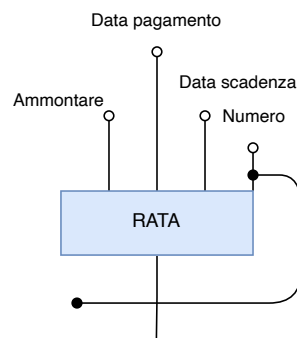


Figura 7: Entità *RATA*

2.1.2. Relazioni

- La relazione **È capo** collega *CAPO* e *FILIALE*, definendo il legame tra il capo di una filiale e la filiale stessa. Le cardinalità di (1,1) indicano che ogni filiale ha uno e un solo capo e non esiste un capo che non faccia riferimento a una sola filiale.



Figura 8: Relazione *È capo*

- La relazione **Lavora** collega *DIPENDENTE* e *FILIALE*. La cardinalità di (1,1) tra la relazione e l'entità *DIPENDENTE* indica che ogni dipendente lavora in una e in una sola filiale, mentre la cardinalità di (1,N) tra la relazione e l'entità *FILIALE* indica che in una filiale lavorano uno o più dipendenti.



Figura 9: Relazione *Lavora*

- La relazione **Di** collega **DIPENDENTE** e **CAPO**. La cardinalità di (1,N) tra la relazione e l'entità **CAPO** indica che un capo dirige uno o più dipendenti, mentre la cardinalità di (1,1) tra la relazione e l'entità **DIPENDENTE** indica che un dipendente ha uno e un solo capo.

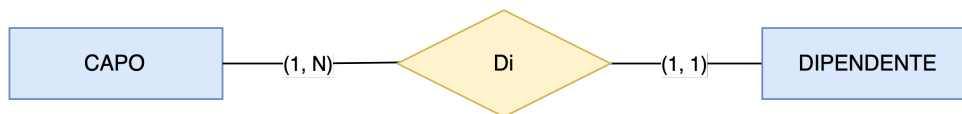


Figura 10: Relazione *Di*

- La relazione **È composto** collega **PRESTITO** e **RATA**, dando forma al legame logico tra un finanziamento e i singoli pagamenti previsti per il suo rimborso. Dal lato di **RATA**, la cardinalità è di (1,1), poiché ogni rata è necessariamente associata ad uno e un solo prestito, essendo **RATA** un'entità debole. Dal lato di **PRESTITO**, invece, la cardinalità è di (1,N), poiché un singolo prestito può essere suddiviso in una o più rate.

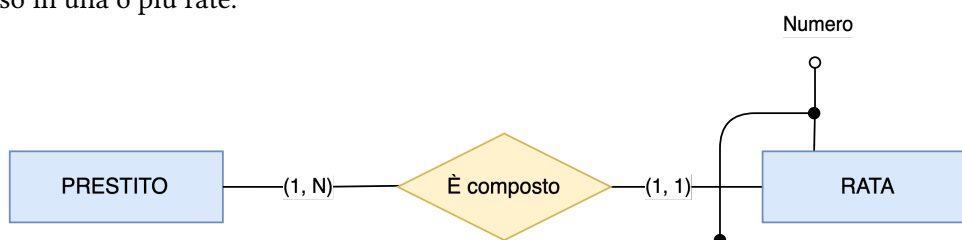


Figura 11: Relazione *È composto*

- La relazione **È associato** collega **CONTO** e **PRESTITO**, definendo il legame tra esso e il conto bancario a cui è associato. Dal lato di **PRESTITO**, la cardinalità è (1,1), poiché ogni prestito deve fare riferimento a un solo conto bancario. Dal lato di **CONTO** la cardinalità è (0,N), infatti un conto non necessariamente ha prestiti associati e può averne più di uno.



Figura 12: Relazione *È associato*

- La relazione **Possiede** collega **CLIENTE** e **CONTO**. Un cliente deve possedere almeno un conto e più clienti possono possedere lo stesso conto (caso di conto cointestato), da cui deriva la cardinalità (1,N) della relazione sul lato di **CLIENTE**. D'altro canto, un **CONTO** deve essere posseduto da almeno un cliente e più conti possono fare riferimento allo stesso cliente (caso in cui uno stesso cliente ha aperto più conti con la banca), da cui deriva la cardinalità (1,N) della relazione sul lato di **CONTO**. Gli attributi *Operazione* e *Data* sulla relazione indicano l'ultima operazione svolta e la data in cui è stata effettuata da quel particolare cliente su quel particolare conto. Nel caso di operazione congiunta di più clienti gli attributi *Operazione/Data* vengono aggiornati per tutti.

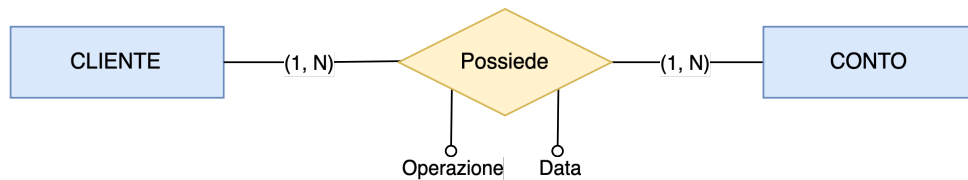


Figura 13: *Relazione Possiede*

- La relazione **Gestisce** collega **DIPENDENTE** e **CLIENTE**. Un sottoinsieme dei dipendenti può seguire le pratiche di un certo numero di clienti della banca, da cui ne deriva la cardinalità $(0, N)$ della relazione sul lato di **DIPENDENTE**. D'altro canto un **CLIENTE** può avere al più un solo gestore che segue le sue attività nella banca, da cui ne deriva la cardinalità $(0, 1)$ della relazione sul lato di **CLIENTE**.



Figura 14: *Relazione Gestisce*

- La relazione **Contiene** collega **FILIALE** e **CONTO** in quanto ogni conto deve fare riferimento ad una e una sola filiale. Una filiale può contenere uno o più conti (anche zero se la filiale è appena stata aperta), da cui ne deriva la cardinalità $(0, N)$ della relazione sul lato di **FILIALE**. D'altro canto un conto deve essere associato ad una e una sola filiale, da cui ne deriva la cardinalità $(1, 1)$ della relazione sul lato di **CONTO**.



Figura 15: *Relazione Contiene*

2.2. Scelte particolari

- La specializzazione non totale **CAPO - DIPENDENTE** ci permette di inserire la molteplicità $(1, 1)$ nella relazione *È capo* e di non dover tenere la molteplicità $(0, 1)$ nel caso in cui **DIPENDENTE** non avesse avuto la specializzazione. Favorisce inoltre una maggiore chiarezza nella relazione *Di*.
- La specializzazione totale di **CONTO** è dovuta alla presenza dei diversi attributi che caratterizzano le due specializzazioni.
- La scelta di assegnare il ruolo di entità a **RATA** è dovuta alla numerosità degli attributi e alla gestione dell'ammontare dei prestiti. Avendo un numero seriale non univoco, è necessario che una parte della chiave sia il codice del prestito.

2.3. Schema Concettuale

A seguito delle analisi effettuate, lo schema concettuale nel modello Entità-Relazione risultante è il seguente:

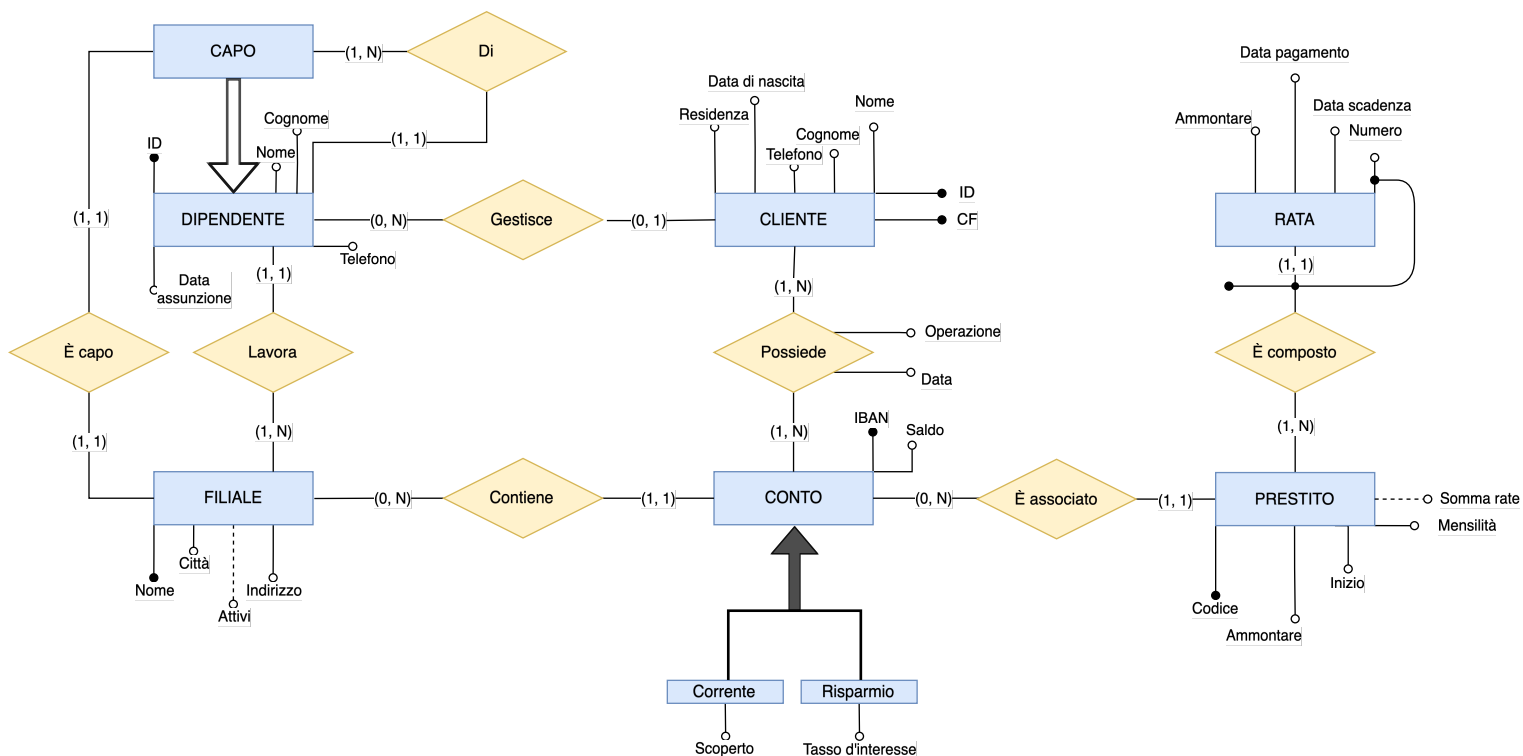


Figura 16: Schema concettuale nel modello Entità Relazione

2.4. Analisi dei cicli

2.4.1. Ciclo *DIPENDENTE - FILIALE - CAPO*:

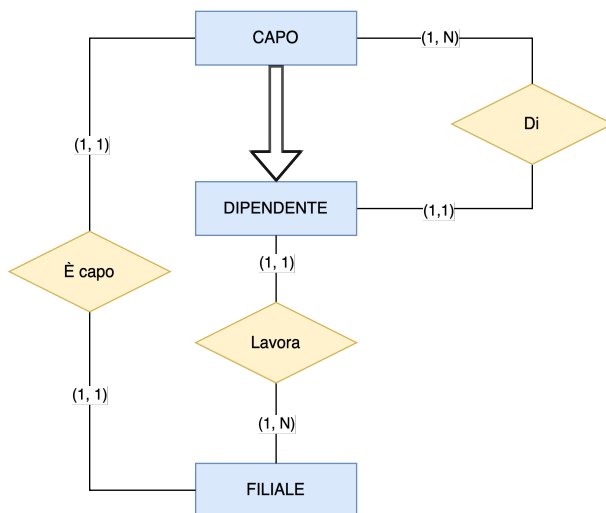


Figura 17: Ciclo *DIPENDENTE - FILIALE - CAPO*

Questo ciclo è problematico in quanto potrebbe accadere che il capo di una filiale non lavori presso la filiale di cui è responsabile. È necessario imporre dei vincoli di integrità per evitare che ciò accada.

2.4.2. Ciclo *DIPENDENTE - CLIENTE - CONTO - FILIALE*:

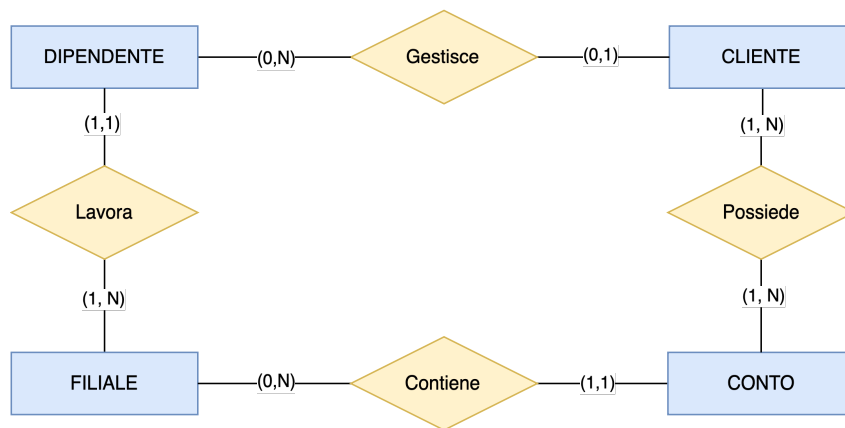


Figura 18: *Ciclo DIPENDENTE - CLIENTE - CONTO - FILIALE*

Questo ciclo non genera problemi di inconsistenza, in quanto a un cliente è permesso avere un gestore che lavora presso una certa filiale e avere più conti aperti in filiali diverse.

2.5. Vincoli d'integrità

Alcuni vincoli non possono essere catturati tramite il modello ER, vengono riportati di seguito e saranno tenuti in considerazione nella sezione di implementazione fisica:

- Il capo di una filiale deve lavorare nella filiale in cui è responsabile.
- Due clienti che hanno gestori differenti non possono avere un conto condiviso.
- Un dipendente non può gestire se stesso.
- Le rate vanno pagate in ordine cronologico, in base al *Numero*.
- La somma dell'importo delle rate deve corrispondere all'ammontare del prestito.

3. Progettazione Logica

Nel processo di ottimizzazione delle prestazioni, nell'analisi delle ridondanze e nella semplificazione dello schema ER concettuale in vista della sua ristrutturazione, sono stati presi in considerazione volumi di dati stimati sulla base di una banca reale di riferimento: Intesa Sanpaolo S.p.A.

3.1. Tabella dei volumi

Nome	Costrutto	Volume
Capo	Entità	3.000
Cliente	Entità	15.000.000
Conto	Entità	12.000.000
Conto Corrente	Entità	10.000.000
Conto Risparmio	Entità	2.000.000
Dipendente	Entità	100.000
Filiale	Entità	3.000
Prestito	Entità	7.000.000
Rata	Entità	84.000.000
Contiene	Relazione	12.000.000
Di	Relazione	100.000
È associato	Relazione	7.000.000
È capo	Relazione	3.000
È composto	Relazione	84.000.000
Gestisce	Relazione	10.000.000
Lavora	Relazione	100.000
Possiede	Relazione	19.000.000

Tabella 2: *Tabella dei volumi*

3.1.1. Considerazioni

Il numero di clienti, conti, dipendenti e filiali è stato stimato sulla base dei dati reali di Intesa Sanpaolo. Abbiamo ipotizzato un numero di prestiti sulla base di una proporzione realistica rispetto ai conti e un numero di rate pagate (in media 12 per prestito). Per distinguere tra conti correnti e conti di risparmio, abbiamo utilizzato la percentuale media nazionale italiana, applicandola al numero totale di conti. I volumi per le relazioni sono stati calcolati tenendo conto delle cardinalità e della natura dei legami tra le entità coinvolte, in modo da mantenere coerenza con il modello concettuale.

3.2. Analisi delle ridondanze

3.2.1. Studio dell'attributo derivato *Attivi* di *FILIALE*

Il primo blocco di operazioni coinvolge l'attributo derivato *Attivi* che produce una ridondanza ed è derivabile da altre entità, nel nostro caso da *CONTO*, *PRESTITO* e *RATA*. Ipotizziamo delle operazioni e le loro relative frequenze che vanno a coinvolgere questo attributo e osserviamo se è conveniente eliminarlo o mantenerlo.

Operazione 1

Interrogazione per leggere il valore *Attivi* di ogni filiale con frequenza di una volta al giorno

Con attributo *Attivi*:

Nome	Costrutto	Accessi	Tipo
Filiale	Entità	3000	Lettura

Tabella 3: *Operazione 1*

$$\text{op1: } (1 \text{ lettura } \{\text{Filiale}\}) \cdot 3.000$$

$$\text{op1} = 3000$$

Per leggere il valore *Attivi* di ogni filiale, è necessario eseguire una lettura della tabella *FILIALE* e leggere l'attributo derivato *Attivi*.

Senza attributo *Attivi*:

Nome	Costrutto	Accessi	Tipo
Filiale	Entità	3000	Lettura
Contiene	Relazione	12.000.000	Lettura
Conto	Entità	12.000.000	Lettura
È associato	Relazione	7.000.000	Lettura
Prestito	Entità	7.000.000	Lettura

Tabella 4: *Operazione 1*

$$\text{op1: } \left((3 \text{ letture } \{\text{Filiale, Contiene, Conto}\} \cdot 4000) + (2 \text{ letture } \{\text{È associato, Prestito}\} \cdot 2333) \right) \cdot 3000$$

$$\text{op1} = 49.998.000$$

Senza l'attributo *Attivi*, per calcolare gli attivi di ogni filiale vengono lette le 3.000 righe della tabella *FILIALE*. Poi, per ogni filiale, si risale ai conti che possiede: in media sono circa 4.000. Vengono poi effettuati altri 4.000 accessi alla tabella *CONTO* per ottenere i saldi.

Lo stesso vale per i prestiti: per ogni filiale si leggono in media 2.333 righe nella tabella *È associato* e poi si accede a *PRESTITO* per recuperarne l'importo delle rate pagate.

In totale quindi, come si vede dalla tabella, bisognerà leggere interamente le relazioni *Contiene*, *È associato* e tutte le entità *CONTO* e *PRESTITO*.

Operazione 2

Inserimento di un conto nella base di dati con frequenza 150 volte al giorno

Con attributo *Attivi*:

Nome	Costrutto	Accessi	Tipo
Conto	Entità	150	Scrittura
Contiene	Relazione	150	Scrittura
Possiede	Relazione	150	Scrittura
Filiale	Entità	150	Lettura
Filiale	Entità	150	Scrittura

Tabella 5: *Operazione 2*

op2: (4 scritture{Conto, Contiene, Possiede, Filiale} + 1 lettura{Filiale}) · 150

$$\text{op2} = 1350$$

Per inserire un conto bisogna scrivere in *CONTO* e nelle due relazioni *Contiene* e *Possiede*, poiché un conto deve avere un cliente che lo possiede e il conto deve essere contenuto da una filiale. Infine bisogna leggere e scrivere nell'entità *FILIALE* per aggiornare l'attributo *Attivi* con il saldo del conto appena inserito.

Senza attributo *Attivi*:

Nome	Costrutto	Accessi	Tipo
Conto	Entità	150	Scrittura
Contiene	Relazione	150	Scrittura
Possiede	Relazione	150	Scrittura

Tabella 6: *Operazione 2*

op2: (3 scrittura{Conto, contiene, possiede}) · 150

$$\text{op2} = 900$$

La logica è come quella vista sopra, con l'eccezione che non serve aggiornare l'attributo *Attivi*, che non è presente.

Operazione 3

Inserimento di un'operazione in *Possiede* con frequenza 1.000.000 al giorno.

Con attributo *Attivi*:

Nome	Costrutto	Accessi	Tipo
Possiede	Relazione	1.000.000	Lettura
Possiede	Relazione	1.000.000	Scrittura
Conto	Entità	1.000.000	Lettura
Conto	Entità	1.000.000	Scrittura
Filiale	Entità	1.000.000	Lettura
Filiale	Entità	1.000.000	Scrittura
Contiene	Relazione	1.000.000	Lettura

Tabella 7: *Operazione 3*

op3: (3 scrittura{Possiede, Conto, Filiale} + 4 letture{Possiede, Conto, Filiale, Contiene}) · 1.000.000

$$\text{op3} = 10.000.000$$

Poichè la relazione *Possiede* contiene l'attributo operazione, ogni volta che un'operazione viene eseguita bisogna aggiornare l'attributo, ciò comporta una lettura e una scrittura. Dopodiché, bisogna anche in questo caso aggiornare il saldo del conto che fa riferimento a quella tupla in possiede, dopodiché solamente leggere la relazione *contiene* per individuare la filiale in cui quel conto ha sede e aggiornare quindi l'attributo *Attivi* della filiale.

Senza attributo *Attivi*:

Nome	Costrutto	Accessi	Tipo
Possiede	Relazione	1.000.000	Scrittura
Possiede	Relazione	1.000.000	Lettura
Conto	Entità	1.000.000	Scrittura
Conto	Entità	1.000.000	Lettura

Tabella 8: *Operazione 3*

$$\text{op3: } (2 \text{ scritture}\{\text{Possiede, Conto}\} + 2 \text{ letture}\{\text{Possiede, Conto}\}) \cdot 1.000.000$$

$$\text{op3} = 6.000.000$$

La logica è la stessa di prima, ma non serve aggiornare l'attributo *Attivi* della filiale, quindi non serve leggere e scrivere nell'entità *FILIALE*.

Operazione 4

Aggiornamento di tutti i prestiti con frequenza di una volta al mese.

Con attributo *Attivi*:

Nome	Costrutto	Accessi	Tipo
Rata	Entità	233.333	Scrittura
È composto	Relazione	233.333	Lettura
Prestito	Entità	233.333	Lettura
Prestito	Entità	233.333	Scrittura
È associato	Relazione	233.333	Lettura
Contiene	Relazione	233.333	Lettura
Filiale	Entità	233.333	Lettura
Filiale	Entità	233.333	Scrittura

Tabella 9: *Operazione 4*

$$\text{op4: } (3 \text{ scritture}\{\text{Rata, Prestito, Filiale}\} +$$

$$5 \text{ Letture}\{\text{È composto, Prestito, È associato, Contiene, Filiale}\}) \cdot 7.000.000 \cdot \frac{1}{30}$$

$$\text{op4} = 2.566.667$$

Abbiamo considerato l'aggiornamento mensile delle rate e quindi questo comporta: la scrittura della rata che viene saldata in quel mese, poi bisogna risalire al prestito a cui essa fa riferimento tramite la relazione *È composto*, aggiornare il prestito di riferimento, dopodiché tramite la relazione *È associato*

ricavare l'IBAN del conto a cui è associato, poter quindi leggere in *Contiene* la filiale in cui quel prestito fa riferimento e quindi operare un aggiornamento dell'attributo attivi della filiale.

Senza attributo attivi:

Nome	Costrutto	Accessi	Tipo
Rata	Entità	233.333	Scrittura
È composto	Relazione	233.333	Lettura
Prestito	Entità	233.333	Lettura
Prestito	Entità	233.333	Scrittura

Tabella 10: *Operazione 4*

$$\text{op4: } (2 \text{ scritture}\{\text{Rata, Prestito}\} + 2 \text{ Letture}\{\text{È composto, Prestito}\}) \cdot 7.000.000 \cdot \frac{1}{30}$$

$$\text{op4} = 1.400.000$$

Anche in questo caso la logica rimane la stessa, ma non serve aggiornare l'attributo *Attivi* della filiale, quindi non serve leggere e scrivere in *FILIALE* e nelle relazioni *È associato* e *Contiene*.

Totale con attributo attivi : 12.571.017

Totale senza attributo attivi : 52.398.900

Questa analisi ci suggerisce che la conservazione dell'attributo derivato *Attivi* sia utile e quindi lo manterremo nel nostro schema ER ristrutturato.

3.2.2. Studio dell'attributo derivato *Somma rate di PRESTITO*

Il secondo blocco di operazioni riguarda la ridondanza introdotta dall'attributo derivato *Somma rate* dell'entità *PRESTITO* che misura il numero di rate che sono state pagate. Anche in questo caso si tratta di un attributo derivato secondo funzioni aggregative e le entità coinvolte sono *RATA* e *PRESTITO*. Possiamo considerare due operazioni (per coerenze con lo studio precedente riportiamo il numero di operazioni giornaliere):

Operazione 1

Inserimento di una rata una volta al mese per ogni prestito della banca

Con attributo *Somma rate*:

Nome	Costrutto	Accessi	Tipo
Rata	Entità	7.000.000	Scrittura
Prestito	Entità	7.000.000	Lettura
Prestito	Entità	7.000.000	Scrittura
È composto	Relazione	7.000.000	Lettura

Tabella 11: *Operazione 1*

$$\text{op1: } (2 \text{ scritture}\{\text{Rata, Prestito}\} + 2 \text{ Letture}\{\text{Prestito, è composto}\}) \cdot 7.000.000 \cdot \frac{1}{30}$$

$$\text{op1} = 1.400.000$$

L'inserimento di una nuova rata comporta la scrittura di una istanza dell'entità *RATA*, seguito dalla lettura nella relazione *È composto* per risalire al prestito corrispondente. La lettura del prestito corretto comporta poi la scrittura per aggiornare l'attributo *Somma rate*.

Senza attributo *Somma rate*:

Nome	Costrutto	Accessi	Tipo
Rata	Entità	7.000.000	Scrittura

Tabella 12: *Operazione 1*

$$\text{op1: } (1 \text{ scrittura}\{\text{Rata}\}) \cdot 7.000.000 \cdot \frac{1}{30}$$

$$\text{op1} = 233.333$$

In questo caso l'operazione di inserimento di una rata comporta semplicemente la scrittura della rata, senza la necessità di leggere il prestito per aggiornare l'attributo *somma rate*.

Operazione 2

Lettura del valore della somma delle rate pagate per ogni prestito con frequenza semestrale.

Per questa analisi abbiamo dovuto introdurre un'ulteriore ipotesi, ovvero il numero medio di rate saldate presenti nella nostra base di dati per ogni prestito. Abbiamo supposto questo numero essere 12, che equivale ad un anno di rate pagate.

Con attributo *Somma rate*:

Nome	Costrutto	Accessi	Tipo
Prestito	Entità	7.000.000	Lettura

Tabella 13: *Operazione 2*

$$\text{op2: } (1 \text{ lettura}\{\text{Prestito}\}) \cdot 7.000.000 \cdot \frac{1}{180}$$

$$\text{op2} = 38.889$$

In questo caso è necessaria una semplice lettura dell'attributo dal prestito corretto, senza la necessità di leggere ogni rata ad esso associata.

Senza attributo *Somma rate*:

Nome	Costrutto	Accessi	Tipo
Prestito	Entità	7.000.000	Lettura
È composto	Relazione	84.000.000	Lettura
Rata	Entità	84.000.000	Lettura

Tabella 14: *Operazione 2*

$$\text{op2: } (1 \text{ lettura}\{\text{Prestito}\} + 12 \cdot 2 \text{ letture}\{\text{Rata}, \text{È composto}\}) \cdot 7.000.000 \cdot \frac{1}{180}$$

$$\text{op2} = 972.222$$

Senza l'attributo ridondante oltre alla lettura del prestito corretto bisogna leggere anche nella relazione *è associato* per ottenere tutte le rate associate al mio prestito. Tra le rate associate mediamente 12

sono state pagate, è da aggiungere quindi una media di ulteriori 12 letture per risalire all'ammontare effettivo già pagato.

Totale con ridondanza di Somma rate: 1.438.889

Totale senza ridondanza di Somma rate: 1.205.555

Per questa ridondanza abbiamo concluso quindi che l'attributo somma rate possa essere rimosso e non essere utilizzato nello schema ER ristrutturato.

3.3. Selezione delle chiavi primarie

In *CLIENTE* abbiamo scelto come chiave primaria l'attributo *ID* rispetto a *Codice Fiscale* per mantenere una linearità con *DIPENDENTE* il quale è identificato a sua volta da un codice univoco. In tutti gli altri casi la chiave candidata a essere primaria era unica.

3.4. Rimozione delle specializzazioni

Per le analisi fatte in precedenza siamo giunti alla conclusione che il blocco *CAPO-Di-DIPENDENTE* può essere «compresso», riducendo la complessità visiva e pratica del problema, eliminando la specializzazione *CAPO* e la relativa relazione *Di*, sostituendo il tutto con un nuovo attributo derivato posto nell'entità *DIPENDENTE*: *Capo*. Di conseguenza viene anche cambiato il riferimento della relazione *È CAPO* che non farà più riferimento all'entità *CAPO* in quanto è stata eliminata ma bensì a *DIPENDENTE* richiedendo un cambio di cardinalità dal lato di *DIPENDENTE*.

Non c'è perdita di informazione in quanto il nuovo attributo *Capo* viene ricavato dalle relazioni *LAVORA* ed *È CAPO*. Per ricavare il capo di un certo dipendente posso andare a vedere la filiale in cui lavora (che è unica per le cardinalità della relazione), tale filiale sarà gestita da uno e un solo capo (deducibile dalle cardinalità della relazione è capo). Si può quindi, in maniera univoca, ricavare il capo di un certo dipendente passando attraverso le relazioni e salvare il dato di interesse nell'attributo *Capo*.

Successivamente la specializzazione di *CONTO* è stata ristrutturata aggiungendo due nuove relazioni: *Tipo-Corrente* e *Tipo-Risparmio* che legano rispettivamente le entità *CORRENTE* e *RISPARMIO* a *CONTO*.

Gli attributi delle tre entità coinvolte nella specializzazione sono rimasti invariati. Le cardinalità delle due nuove relazioni sono (0,1) dal lato di *CONTO*. Dal lato di *CORRENTE* e *RISPARMIO* sono (1,1) in quanto i due tipi di conto sono associati a uno e un solo conto.

Questa scelta non ci permette di catturare tutti i requisiti su conto, il quale deve essere esclusivamente un conto corrente o un conto di risparmio. Ciò va tenuto in considerazione aggiungendolo ai vincoli di integrità.

Le chiavi primarie di *CORRENTE* e di *RISPARMIO* sono delle chiavi primarie legate alla relazione con *CONTO*, ne ereditano quindi la chiave primaria *IBAN*. Da notare il fatto che l'insieme degli *IBAN* di *CORRENTE* deve essere disgiunto dall'insieme degli *IBAN* di *RISPARMIO* (non esiste un conto che è sia corrente che di risparmio in quanto la specializzazione originariamente era disgiunta).

3.5. Schema ER ristrutturato

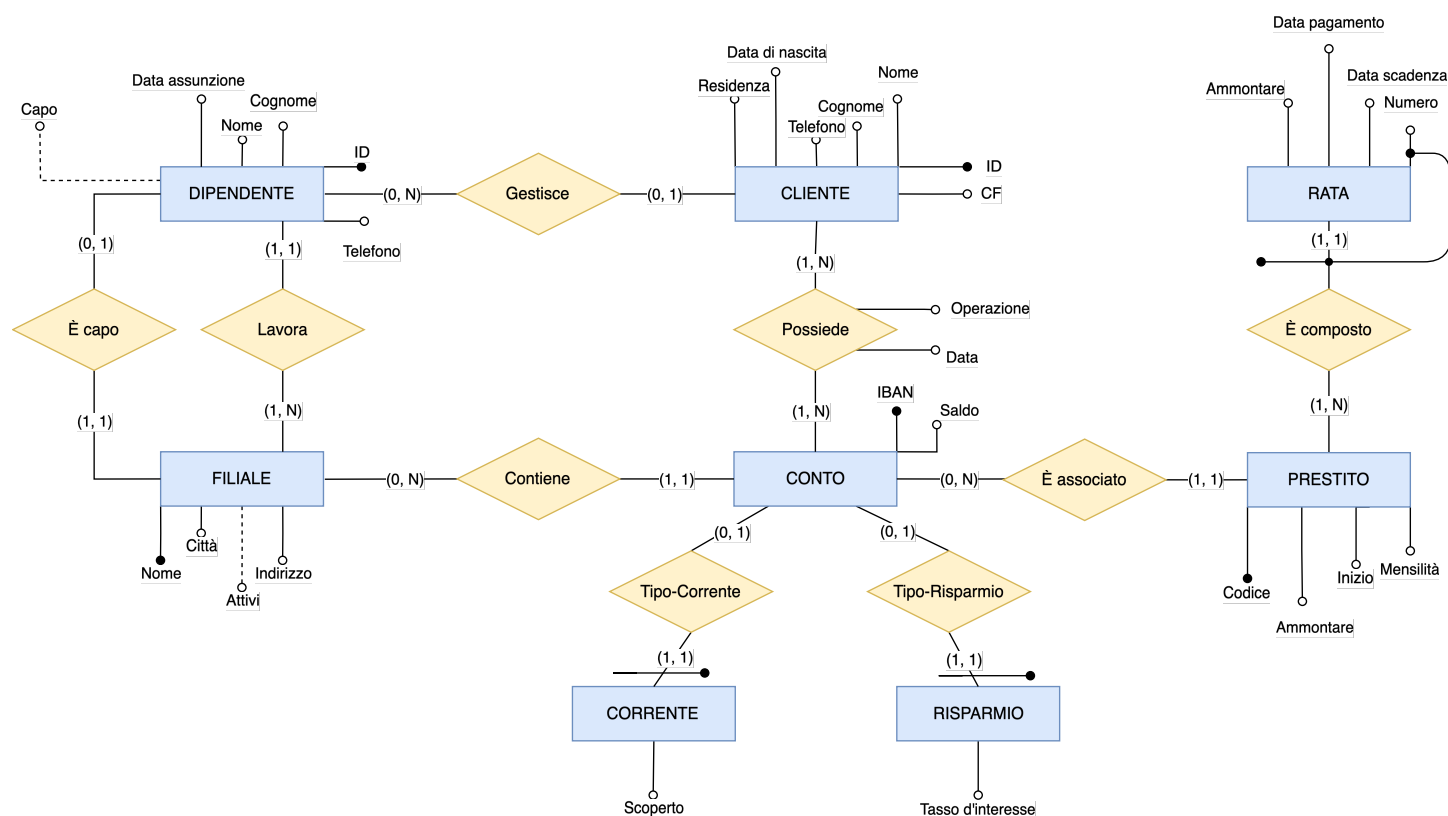


Figura 19: Schema ER ristrutturato

3.6. Schema Logico

CLIENTE (ID, CF, Residenza, DataDiNascita, Telefono, Cognome, Nome, *Gestore*)

- CF: UNIQUE

CONTO (IBAN, Saldo, *Filiale*)

- *FilialeAppartenenza*: NOT NULL

CONTO_CORRENTE (IBAN, Scoperto)

- IBAN: UNIQUE

CONTO_RISPARMIO (IBAN, TassoInteresse)

- IBAN: UNIQUE

POSSIEDE (Cliente, Conto, Operazione, Data)

PRESTITO (Codice, Ammontare, Inizio, Mensilità, *ContoAssociato*)

- *ContoAssociato*: NOT NULL

RATA (Numero, CodicePrestito, Ammontare, DataPagamento, DataScadenza)

- *CodicePrestito*: NOT NULL

FILIALE (Nome, Città, Attivi (derivato), Indirizzo, *Capo*)

- *Capo*: NOT NULL

DIPENDENTE (ID, Nome, Cognome, Telefono, DataAssunzione, *Capo* (derivato), *Filiale*)

- *Filiale*: NOT NULL
- *Capo*: NOT NULL

Legenda: Le chiavi primarie sono sottolineate e le chiavi esterne sono in corsivo.

La traduzione delle relazioni è stata fatta secondo i canoni standard seguendo le cardinalità di esse.

3.6.1. Chiavi esterne

- *Gestore* è chiave esterna di *CLIENTE* rispetto a *DIPENDENTE*
- *Filiale* è chiave esterna di *CONTO* rispetto a *FILIALE*
- *IBAN* è chiave esterna di *CONTO CORRENTE*, *CONTO RISPARMIO* rispetto a *CONTO*
- *Conto* chiave esterna di *POSSIEDE* rispetto a *CONTO*
- *Cliente* è chiave esterna di *POSSIEDE* rispetto a *CLIENTE*
- *ContoAssociato* è chiave esterna di *PRESTITO* rispetto a *CONTO*
- *Capo* è chiave esterna di *FILIALE* rispetto a *DIPENDENTE*
- *CodicePrestito* è chiave esterna di *RATA* rispetto a *PRESTITO*
- *Capo* è chiave esterna di *DIPENDENTE* rispetto a *DIPENDENTE*

4. Popolamento del database

4.1. Creazione delle tabelle

Per ogni entità è stata creata una tabella nello schema fisico dove gli attributi dell'entità corrispondono ai campi della tabella. I campi della tabella sono stati opportunamente dichiarati in base al tipo di dato e aggiunti eventuali controlli sul loro valore per avere coerenza logica con quanto richiesto tramite l'utilizzo della condizione `CHECK()`.

Campi particolari che richiedevano di essere ad esempio chiave primaria, unici, o non nulli sono stati settati tramite gli appositi comandi.

Le tabelle sono state create in un ordine preciso; in particolare i vincoli di chiave esterna sono stati aggiunti solo quando tutte le tabelle coinvolte erano esistenti.

La tabella possiede è stata creata in quanto corrisponde alla relazione molti a molti tra l'entità *CONTO* e l'entità *CLIENTE*.

4.2. Modalità di generazione dei dati

Riportiamo di seguito la tabella dei volumi debitamente proporzionata sulla quale abbiamo creato i dati per il nostro database.

Nome	Volume
Cliente	30.000
Conto	24.000
Conto Corrente	20.000
Conto Risparmio	4.000
Dipendente	200
Filiale	6
Prestito	14.000
Rata*	1.764.000
Possiede	38.000

Tabella 15: *Tabella dei volumi proporzionata*

* Abbiamo incluso tutte le rate, pagate e non, con una media di 126 rate per prestito.

4.2.1. Dati *FILIALE*

Questi dati non richiedevano particolari attenzioni poiché non soggetti a nessun tipo di vincolo particolare. Per comodità è stato scelto di nominare le filiali con numeri interi crescenti, e per la logica di popolamento e vincoli di cardinalità, per ogni record di filiale, il codice del suo manager è pari al suo nome.

4.2.2. Dati *DIPENDENTE*

I dipendenti sono composti da dati che per ciò che concerne gli attributi *Nome*, *Cognome*, *Data di assunzione*, *Telefono* sono stati generati e assegnati casualmente, mentre più delicato è il nome (numero) di filiale che per i primi 6 dipendenti è stato assegnato progressivamente per mantenere la logica dei

dati di *FILIALE* mentre per i restanti in maniera casuale, inoltre il campo *Capo* è stato inizializzato a -1 per poi essere assegnato correttamente durante il popolamento.

Abbiamo stabilito per comodità che solo i primi 100 dipendenti sono gestori.

4.2.3. Dati *CLIENTE*

Il campo *Gestore* è il più delicato. I clienti con *ID* da 1 a 4.000 e da 24.001 a 28.000 hanno nell'ordine lo stesso gestore (il cliente con *ID* = 1 ha lo stesso gestore del cliente con *ID* = 24.001, quello con *ID* = 2 ha lo stesso gestore del cliente con *ID* = 24.002 e così via). I clienti con *ID* da 4001 a 16.000 hanno un gestore (eventualmente ripetuto) e i rimanenti ne sono privi.

4.2.4. Dati *CONTO*

Per assegnare gli *IBAN* in maniera coerente, i primi 20.000 sono stati assegnati ai conti correnti, i restanti 4.000 ai conti di risparmio.

4.2.5. Dati *POSSIEDE*

Per garantire la coerenza e i vincoli di integrità, i primi 4.000 *IBAN* sono stati assegnati ai primi 4.000 clienti cointestati con i clienti con *ID* compreso tra 24.001 e 28.000, in maniera tale da rispettare il vincolo sui gestori (entrambi hanno lo stesso gestore).

I successivi 2.000 *IBAN* sono cointestati ma solo uno dei clienti ha il gestore.

I restanti *IBAN* hanno un unico proprietario, alcuni con gestore e altri senza.

4.3. Creazione dei trigger

4.3.1. Trigger *FILIALE-DIPENDENTE*

Sono stati creati dei trigger per gestire le problematiche tra dipendente e filiale che non è stato possibile catturare con i vincoli tramite lo schema relazionale.

Il manager di una filiale deve fare riferimento alla filiale che gestisce, pertanto non deve essere possibile cambiare la filiale di un manager. Il trigger controlla che su ogni inserimento o modifica nella tabella dipendente venga rispettato il vincolo appena descritto, sollevando un'eccezione in caso di problemi bloccando di conseguenza l'inserimento o la modifica.

Un altro trigger simile controlla che una volta assegnato il manager in una filiale esso lavori effettivamente in quella filiale.

4.3.2. Trigger *FILIALE-CONTO-PRESTITO-RATA*

La creazione delle rate di un prestito è stata gestita in modo automatico da un trigger il quale dopo l'inserimento di un prestito, calcola l'importo mensile di ogni rata in base all'ammontare e il numero di mensilità, creando le rate (tutte con lo stesso importo mensile) e mettendo la data di scadenza in modo coerente e sequenziale.

Un altro trigger controlla la possibilità di poter pagare una rata bloccando l'aggiornamento in caso la rata fosse già stata pagata, in caso di pagamento concesso, il trigger si occupa anche di aggiornare gli attivi della filiale corrispondente.

In modo analogo un trigger aggiorna gli attivi della filiale ogni volta che un nuovo prestito viene creato.

4.3.3. Trigger *POSSIEDE-CONTO-FILIALE*

Il calcolo degli attivi, analogamente a quanto avviene per prestiti e rate, viene fatto in automatico da un trigger ogni volta che è aggiornato il saldo di un conto.

Per le scelte fatte nessun IBAN in conto corrente deve comparire in conto di risparmio e viceversa ma tutti gli IBAN di *CONTO CORRENTE* e di *CONTO RISPARMIO* devono comparire in *CONTO*, tale vincolo viene rispettato da due opportuni trigger.

La coerenza delle operazioni eseguibili su un determinato conto anch'essa è verificata da due appositi trigger. Viene controllato che l'operazione sia sensata sul conto (non posso aprire un conto due volte e non posso fare operazioni sul conto di risparmio) e in caso di prelievo un trigger si occupa di verificare il saldo rimanente e di aggiornarlo.

4.4. Inserimento tabelle e dati nel database

Per creare il database richiesto, popolarlo e testare le query assegnate è stata seguita una particolare logica affinché tutto venisse inserito correttamente.

Per prima cosa è stato creato il database, assegnando i volumi dei dati con valori proporzionati alla tabella dei volumi precedentemente proposta.

Vengono poi caricati nel sistema tutti i trigger utilizzati e temporaneamente disabilitati per possibili inconsistenze momentanee nell'inserimento dei dati. La modalità di generazione casuale dei dati è stata pensata in modo tale che, al termine degli inserimenti iniziali, tutto sia coerente e non ci siano errori.

Le prime tabelle popolate sono *FILIALE* e *DIPENDENTE*. Al termine del popolamento vengono eseguiti forzatamente due trigger in maniera tale da assegnare automaticamente i manager (che non erano stati inseriti) e verificare la presenza di eventuali errori (di base i dati sono stati generati consistentemente).

Estratti dei possibili gestori vengono inseriti i clienti, successivamente la tabella *CONTO* con le relative *CONTO CORRENTE* e *CONTO DI RISPARMIO*. Una volta inseriti questi dati è possibile procedere al popolamento della tabella *POSSIEDE* che gestisce tutte le connessioni tra i clienti e i loro conti.

Per la macrocategoria dei prestiti, una volta generati quest'ultimi e le relative rate andiamo, tramite apposito script, a pagare le rate che hanno una data di scadenza antecedente a quella odierna. Inseriti tutti i prestiti aggiorniamo l'attributo *attivi* della tabella *FILIALE* in maniera automatica sui dati inseriti e al termine riattiviamo tutti i trigger.

Gli script utilizzati non potevano essere sempre sostituiti dai trigger, infatti non era possibile tenerli tutti attivi e inserire tutti i valori in maniera ordinata e raggruppati per tabelle, ma avremmo dovuto fare attenzione volta per volta. Degli esempi di inserimenti di record sono presentati più avanti.

4.5. Test

Finito di popolare tutto il database ci assicuriamo tramite dei test che tutto sia perfettamente funzionante, che rispetti i requisiti che ci siamo imposti e che ci dia i risultati attesi. Questa verifica viene effettuata confrontando il risultato ottenuto dalle operazioni con i risultati attesi.

4.6. Test Dipendente-Filiale

1. Tentiamo di modificare la filiale di riferimento di un manager senza aggiornare il ruolo di manager. Il trigger ci protegge e ci vieta l'inserimento (un dipendente non può lavorare nella filiale *A* ed essere manager della filiale *B*).

2. Simile al precedente, proviamo ad assegnare il ruolo di manager di una filiale a un dipendente che lavora presso una filiale diversa. Il trigger blocca l'azione e ci restituisce l'errore (la modifica non viene effettuata).
3. Inseriamo un nuovo dipendente: non è necessario specificare il campo manager in quanto il trigger apposito si occupa di ricercare l'id del manager nella filiale dove lavora il nuovo dipendente e assegnare il campo corrispondente.
4. Come il caso (3) ma con l'aggiunta che questo dipendente diventi manager della filiale in cui lavora. Il trigger che viene innescato sulla modifica del campo manager (che passa da -1 [non manager] a un id di filiale valido) provvede ad aggiornare il campo manager di tutti i dipendenti che lavorano nella filiale dove è appena stato modificato il manager.
5. Controlliamo una semplice operazione di rimozione di un dipendente che non è manager.

4.7. Test Prestito-Rata

1. Inseriamo un nuovo prestito. Le rate relative verranno generate in maniera automatica dal trigger che si occupa di andare a recuperare il valore di *Mensilità* e generare altrettanti record nella tabella *RATA* riempiendo in maniera adeguata tutti i campi.
2. Modifichiamo la data di pagamento di una rata, portandola da NULL a una data valida. Il controllo del trigger sarà di verificare che non ci siano rate precedenti ancora da pagare.

4.8. Test Conto-Filiale

1. Simuliamo un versamento e un prelievo, quindi andiamo a modificare il valore del saldo dei conti. A questo punto dei trigger controllano (solo nel secondo caso) che il prelievo possa essere effettuato, quindi che il saldo sia un numero valido (non minore dello scoperto), dopodiché in entrambi i casi vengono automaticamente aggiornati gli attivi delle filiali. Lo scopo del test è comunque di verificare che il saldo venga correttamente modificato.
2. Controlliamo che il trigger che controlla la validità dei saldi funzioni, forzando la modifica di un saldo a un valore non valido. Ci attendiamo un errore.
3. Simile al primo test con il focus sull'aggiornamento degli attivi della filiale di riferimento.
4. Proviamo a inserire un IBAN valido nella tabella *CONTO* (necessario per i vincoli di chiave esterna) e poi nella tabella *CONTO CORRENTE*. Questo non dovrebbe generare problemi. Proviamo a inserire l'IBAN anche in *CONTO RISPARMIO*, il trigger vieta tale operazione e, dato che siamo all'interno di una transazione, tutti e tre gli inserimenti vengono rimossi (rollback).
5. Test di consistenza dei gestori diversi su conti cointestati

5. Query

Dopo aver verificato il corretto funzionamento del database e dei trigger con i test sopra esposti, andiamo a sviluppare le query richieste.

5.1. Query 1

Restituire il numero medio di rate dei prestiti associati a conti nelle filiali di Udine.

sql

Query 1

```
SELECT AVG(mensilità) AS media_rate
FROM prestito, conto, filiale
WHERE prestito.conto = conto.iban
AND conto.filiale = filiale.nome
AND filiale.città = 'Udine';
```

La richiesta è immediata con l'utilizzo della funzione aggregata `AVG()`.

5.2. Query 2:

Restituire i clienti con solo conti di risparmio in filiali che hanno tra i 30 e i 32 dipendenti.

sql

Query 2

```
CREATE OR REPLACE VIEW filiali_3032 AS
SELECT filiale, COUNT(*) AS n_dip
FROM dipendente
GROUP BY filiale
HAVING COUNT(*) BETWEEN 30 AND 32;

SELECT cliente.id
FROM cliente, possiede, conto, filiali_3032
WHERE cliente.id = possiede.cliente
AND possiede.conto = conto.iban
AND conto.filiale = filiali_3032.filiale
AND NOT EXISTS (
    SELECT 1
    FROM contocorrente
    WHERE contocorrente.iban = conto.iban
);
```

Per comodità è stata creata una vista dove è stata fatta una selezione sulla tabella *FILIALE*, tenendo solamente quelle che rispettavano il vincolo sul numero dei dipendenti.

La query sfrutta questa vista per cercare i clienti che hanno almeno un conto di risparmio in queste filiali e che non hanno nessun conto corrente associato.

5.3. Query 3:

Restituire i capi che gestiscono almeno 3 clienti che possiedono almeno 100.000€.

sql

Query 2

```
CREATE OR REPLACE VIEW clienti_ricchi AS
SELECT cliente.id, SUM(conto.saldo) AS soldi, cliente.gestore
FROM cliente, possiede, conto
WHERE cliente.id = possiede.cliente
  AND conto.iban = possiede.conto
GROUP BY cliente.id, cliente.gestore
HAVING SUM(conto.saldo) > 100000;

SELECT DISTINCT capo
FROM dipendente
WHERE EXISTS (
  SELECT *
  FROM clienti_ricchi c1, clienti_ricchi c2, clienti_ricchi c3
  WHERE c1.gestore = dipendente.id
    AND c2.gestore = dipendente.id
    AND c3.gestore = dipendente.id
    AND c1.id < c2.id AND c2.id < c3.id
);
```

La vista creata è una restrizione sui clienti che rispettano il vincolo. È stata effettuata con l'utilizzo della funzione SUM() poiché il saldo era relativo a tutti i conti posseduti. Per validare un capo è stato fatto il prodotto cartesiano triplo della vista e, dopo essere state selezionati solamente le righe con gestore uguale, è stato controllato che i clienti fossero tutti e tre diversi.

5.4. Query 4:

Restituire i dipendenti non capo che gestiscono esattamente 2 clienti, uno con solo conti correnti e uno con solo conti di risparmio.

sql

Query 4

```
CREATE OR REPLACE VIEW clienti_correnti AS
SELECT possiede.cliente, cliente.gestore
FROM possiede, cliente
WHERE possiede.cliente = cliente.id
  AND NOT EXISTS (
    SELECT 1
    FROM contorisparmio
    WHERE contorisparmio.iban = possiede.conto
  );

CREATE OR REPLACE VIEW clienti_risparmio AS
SELECT possiede.cliente, cliente.gestore
FROM possiede, cliente
WHERE possiede.cliente = cliente.id
```

```

AND NOT EXISTS (
    SELECT 1
    FROM contocorrente
    WHERE contocorrente.iban = possiede.conto
);

SELECT id
FROM dipendente
WHERE capo <> id
AND EXISTS (
    SELECT *
    FROM clienti_correnti cc1
    WHERE cc1.gestore = dipendente.id
    AND NOT EXISTS (
        SELECT *
        FROM clienti_correnti cc2
        WHERE cc2.gestore = dipendente.id
        AND cc1.cliente <> cc2.cliente
    )
)
AND EXISTS (
    SELECT *
    FROM clienti_risparmio cr1
    WHERE cr1.gestore = dipendente.id
    AND NOT EXISTS (
        SELECT *
        FROM clienti_risparmio cr2
        WHERE cr2.gestore = dipendente.id
        AND cr1.cliente <> cr2.cliente
    )
);

```

La prima (seconda) vista seleziona solamente i clienti che hanno almeno un conto corrente (di risparmio) e che non hanno nessun conto di risparmio (corrente). La query seleziona i dipendenti non capo (con la verifica *ID <> Capo*) e poi controlla che esista un unico cliente nella prima vista e un unico cliente nella seconda vista.

5.5. Query 5:

Restituire il cliente con il prestito più alto nella filiale di Roma che non ha come gestore un dipendente con meno di 3 anni di esperienza.

sql

Query 5

```

CREATE OR REPLACE VIEW clienti_gestiti_3 AS
SELECT cliente.id
FROM cliente, dipendente
WHERE cliente.gestore = dipendente.id
    AND dipendente.data_assunzione < DATE '', data_limite, '';

CREATE OR REPLACE VIEW candidati AS
SELECT cliente.id, prestito.ammontare

```

```

FROM cliente, clienti_gestiti_3, possiede, prestito, conto, filiale
WHERE cliente.id = clienti_gestiti_3.id
  AND cliente.id = possiede.cliente
  AND possiede.conto = prestito.conto
  AND prestito.conto = conto.iban
  AND conto.filiale = filiale.nome
  AND filiale.città = 'Roma';

SELECT id, ammontare
FROM candidati c1
WHERE NOT EXISTS (
  SELECT 1
  FROM candidati c2
  WHERE c2.ammontare > c1.ammontare
);

```

La prima vista ci restringe i possibili clienti a quelli che hanno un gestore assunto da almeno 3 anni. La seconda vista, a partire dalla prima, fa un ulteriore filtro prendendo i clienti solo della filiale di Roma. La query si occupa di verificare, per ogni cliente, che tra i clienti della seconda vista non ce ne sia qualcuno con saldo maggiore del proprio, in tal caso stampa il cliente.

6. Analisi dei dati

Di seguito sono descritte le analisi eseguite sul database, per estrarre informazioni riguardanti i clienti, i loro conti, i prestiti e le rate pagate.

È importante sottolineare che, in quanto i dati sono stati generati in modo casuale, le tendenze, correlazioni e distribuzioni osservate non riflettono necessariamente situazioni reali.

6.1. Distribuzione dei prestiti per mensilità

Viene esaminata la distribuzione delle mensilità dei prestiti associati a conti con saldo superiore a 50.000€ gestiti da un gestore.

sql

Query

```
CREATE OR REPLACE VIEW clienti_gestiti AS
SELECT cliente.id, cliente.gestore
FROM cliente, dipendente
WHERE cliente.gestore = dipendente.id;

SELECT mensilità, COUNT(*)
FROM clienti_gestiti, possiede, conto, prestito
WHERE clienti_gestiti.id = possiede.cliente
AND possiede.conto = conto.iban
AND conto.saldo > 50.000
AND possiede.conto = prestito.conto
GROUP BY mensilità
```

Distribuzione dei prestiti per mensilità

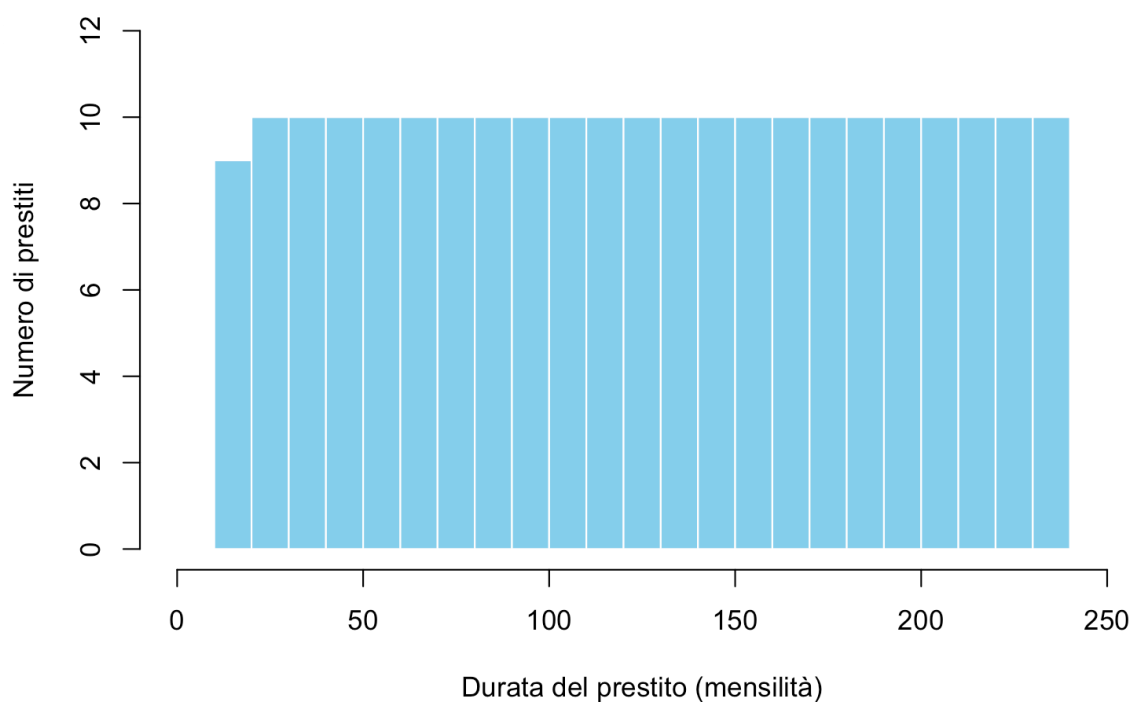


Figura 20: *Distribuzione dei prestiti per mensilità*

6.2. Analisi attivi per anzianità gestori

L'obiettivo è analizzare la relazione tra l'anzianità dei gestori (ricavata dalla data di assunzione) e l'ammontare totale dei conti gestiti.

sql

Query

```
CREATE OR REPLACE VIEW dipendenti_gestori AS
SELECT dipendente.data_assunzione, dipendente.id
FROM cliente, dipendente
WHERE cliente.gestore = dipendente.id;

SELECT SUM(conto.saldo) as skey, dipendenti_gestori.data_assunzione
FROM clienti_gestiti, possiede, conto, dipendenti_gestori
WHERE clienti_gestiti.id = possiede.cliente
AND possiede.conto = conto.iban
AND dipendenti_gestori.id = clienti_gestiti.gestore
GROUP BY dipendenti_gestori.data_assunzione
```

Viene creata una vista che contiene i gestori e la loro data di assunzione. La query finale calcola la somma dei saldi dei conti gestiti da ciascun gestore, raggruppando i risultati per data di assunzione. I dati vengono visualizzati in un grafico a dispersione.

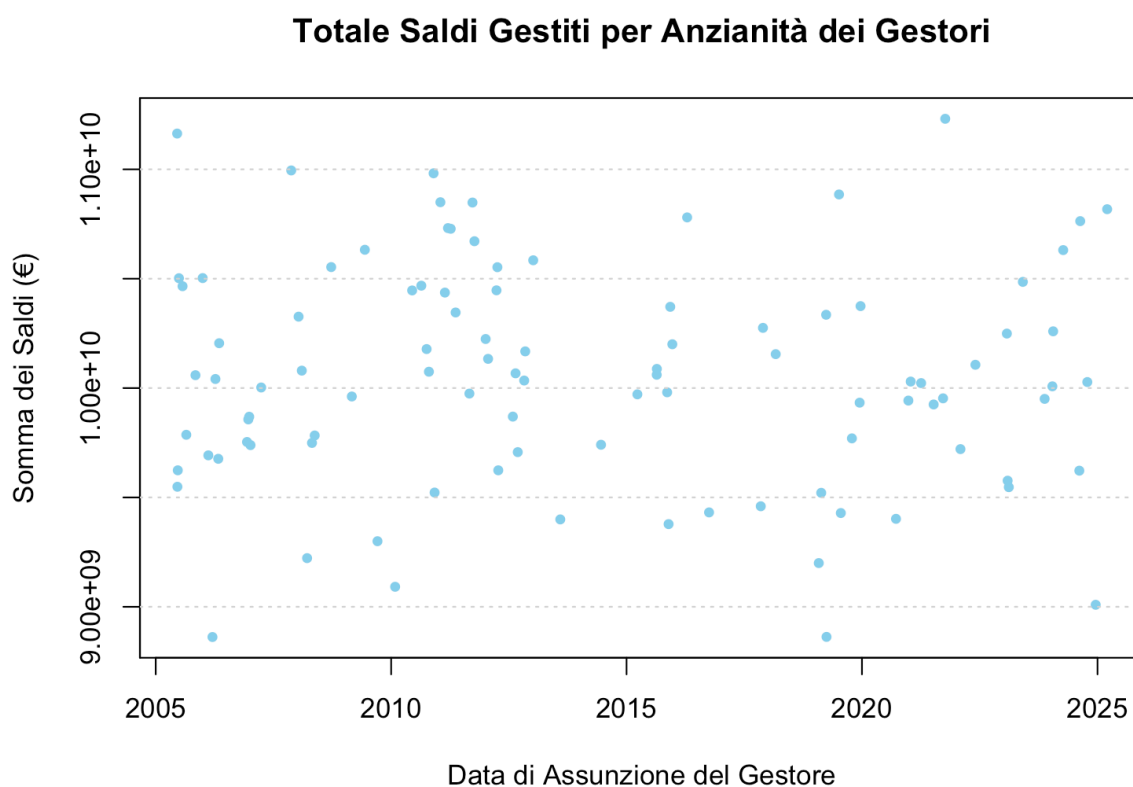


Figura 21: *Analisi attivi per anzianità gestori*

6.3. Analisi conti cointestati

L'obiettivo è determinare il numero di conti cointestati che hanno un prestito associato, raggruppandoli per filiale.

sql

Query

```
CREATE VIEW conti_cointestati AS
SELECT p1.conto, conto.filiale
FROM possiede AS p1, conto
WHERE p1.conto = conto.iban AND EXISTS (
  SELECT *
  FROM possiede AS p2
  WHERE p1.conto = p2.conto AND p1.cliente < p2.cliente)

SELECT filiale, COUNT(*) AS n_conti
FROM conti_cointestati, prestito
WHERE conti_cointestati.conto = prestito.conto
AND ammontare > 50.000
GROUP BY filiale
```

La vista `conti_cointestati` contiene i conti cointestati, raggruppati per filiale. La query finale conta il numero di conti cointestati con prestiti associati, filtrando per ammontare maggiore di 50.000€. I risultati vengono visualizzati in un grafico a barre, che mostra il numero di conti cointestati con prestito per filiale.

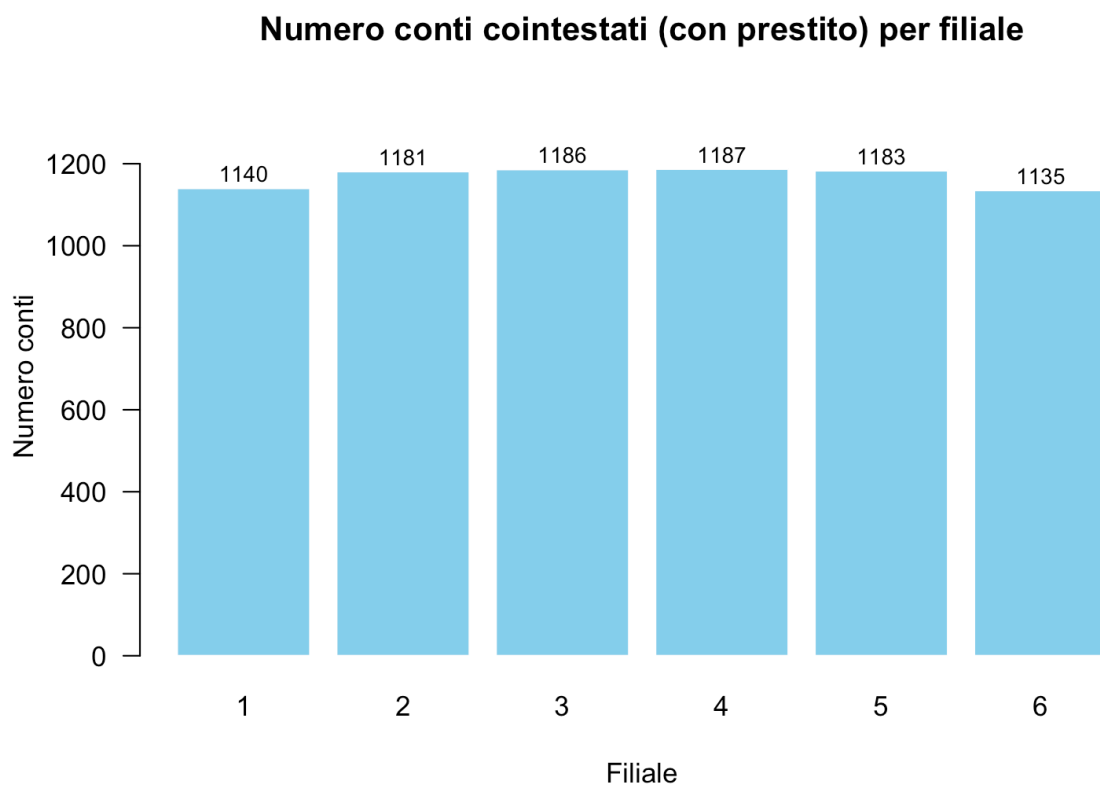


Figura 22: *Analisi conti cointestati*

7. Conclusioni

Fare un'analisi dei requisiti reale ha evidenziato la difficoltà reale di avere una documentazione completa, non ambigua e che rimanesse coerente con se stessa. Certi requisiti erano facilmente deducibili, altri sono stati «imposti» da noi, altri ancora ci siamo ritrovati a specificarli man mano perché non erano stati tenuti in considerazione sin dall'inizio.

Le progettazioni logiche e concettuali rimarcavano l'importanza di una scelta accurata di quali informazioni avessero il ruolo di entità e quali di semplici attributi.

Altrettanto importante la scelta delle relazioni e delle relative molteplicità, molte volte dettate dai vincoli. Ciò non catturato dallo schema ER (vincoli di integrità) andava comunque documentato per implementare dei trigger nella progettazione fisica.

Una buona parte del lavoro si è incentrata sulla generazione dei dati (anche del fatto che fossero coerenti tra loro e che rispettassero i vincoli imposti dal problema) e sul lavoro di popolamento tramite R. La creazione del database e delle tabelle è un lavoro su cui porre attenzione, in particolare l'ordine di creazione delle tabelle e l'assegnamento di chiavi primarie e/o esterne è fondamentale.

Per testare la funzionalità del database ci siamo serviti di test che miravano a verificare dei casi particolari dei vincoli e della funzionalità dei trigger sia per garantire la coerenza, sia per l'aggiornamento automatico di attributi derivati.

Le query invece, sono servite per capire come ragiona un database dietro a delle maschere semplificate dei software in circolazione, dove l'utente semplicemente scrive in linguaggio quasi naturale ciò che gli serve e poi viene tradotto in linguaggio SQL.

I grafici finali sfruttano la potenzialità delle query per analizzare dei dati che, con R, sarebbero stati recuperati in maniera meno semplice.

In conclusione, questo progetto ci ha consentito di mettere in campo tutte le conoscenze teoriche (e non) acquisite durante il corso, e di acquisirne di nuove.