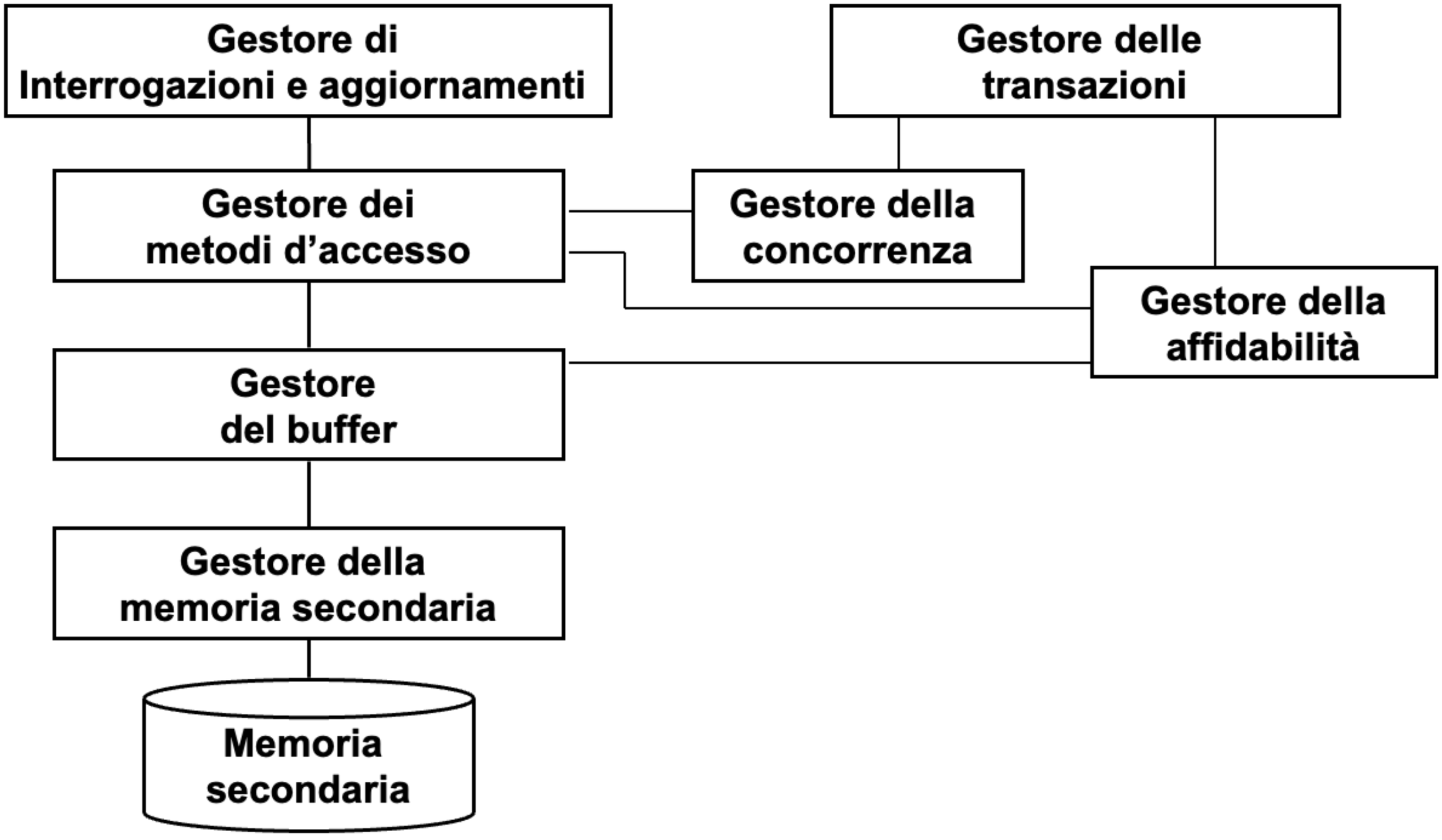
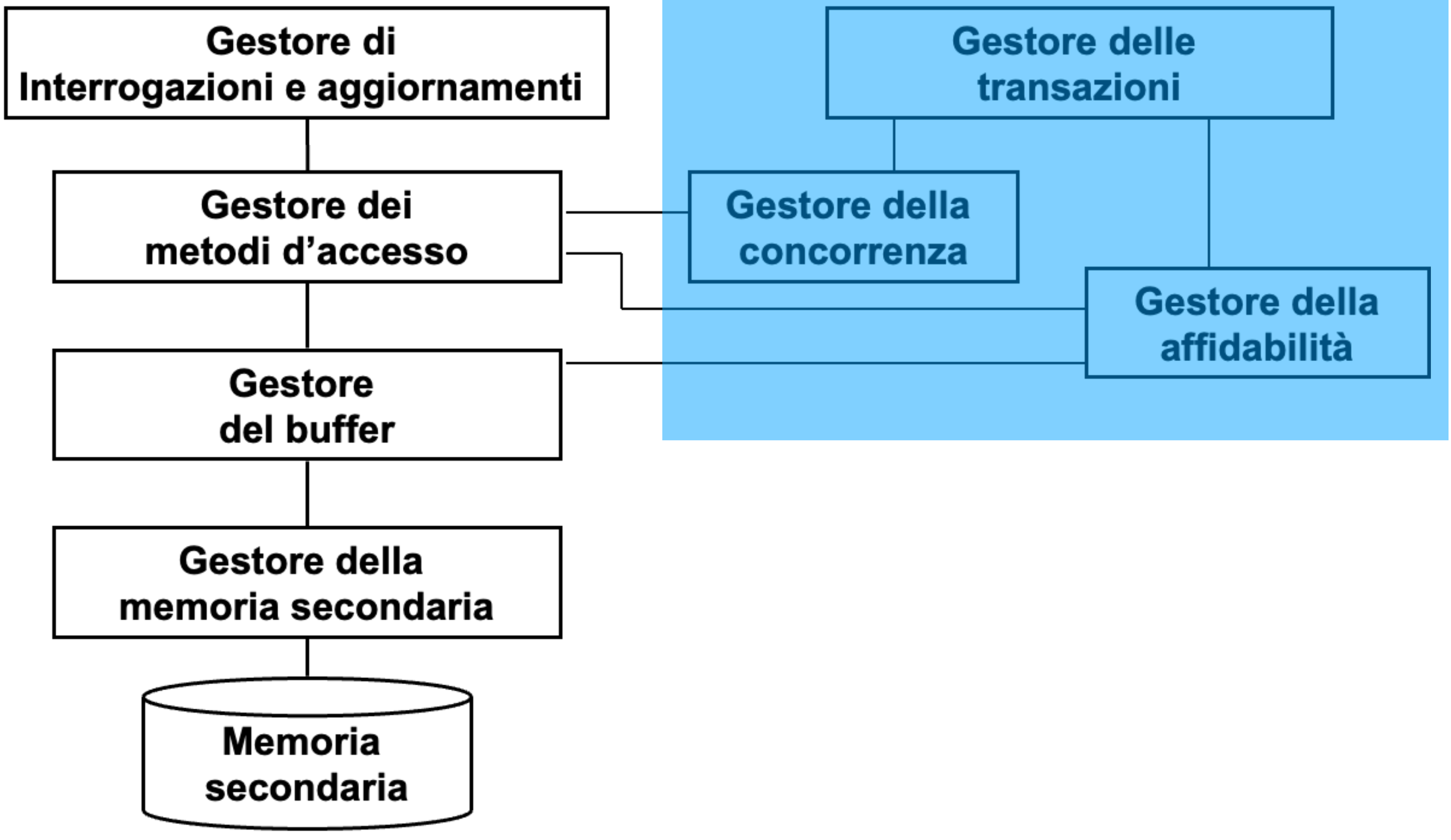
Gestione delle Transazioni


Gestore di
Interrogazioni e aggiornamenti
Gestore delle
transazioni
Gestore dei
metodi d'accesso
Gestore della
concorrenza
Gestore della
affidabilità
Gestore
del buffer
Gestore della
memoria secondaria
Memoria
secondaria


Gestore di
Interrogazioni e aggiornamenti
Gestore dei
metodi d'accesso
Gestore
del buffer
Gestore della
memoria secondaria
Memoria
secondaria
Gestore delle
transazioni
Gestore della
concorrenza
Gestore della
affidabilità

Sistemi Multiutente

- Un criterio per classificare un sistema di basi di dati è il **numero degli utenti** che possono fruirne **simultaneamente**
- Un DBMS è **monoutente** se il sistema può essere usato al massimo da un utente alla volta
- Un DBMS è **multiutente** se invece può essere usato contemporaneamente da più utenti, generando accessi concorrenti alla base di dati
- La maggior parte dei DBMS è multiutente

Multitasking

- L'accesso alla base di dati e più in generale l'utilizzo dei sistemi di elaborazione da parte di **più utenti contemporaneamente** è possibile grazie al concetto di **multitasking**
- Il **multitasking** consente al calcolatore di eseguire più programmi (o meglio, **processi**) nello stesso momento
- Se **esiste una sola CPU** questa è in realtà in grado di eseguire al massimo **un processo alla volta**, tuttavia i sistemi operativi multitasking eseguono alcuni comandi di un processo, poi lo **sospendono** per eseguirne altri, e così via
- **L'esecuzione** di un processo viene **ripresa** nel punto in cui era stata sospesa ogniqualvolta il processo torna ad usare la CPU
- **L'esecuzione** concorrente dei processi risulta quindi **alternata** (*interleaved*)
- Se il sistema è dotato di **più CPU** è possibile realizzare una qualche forma di **elaborazione parallela** di più processi

Transazione

- Una transazione identifica una **unità elementare di lavoro** svolta da un'applicazione, cui si vogliono associare **particolari caratteristiche** di **correttezza**, **robustezza** e **isolamento**

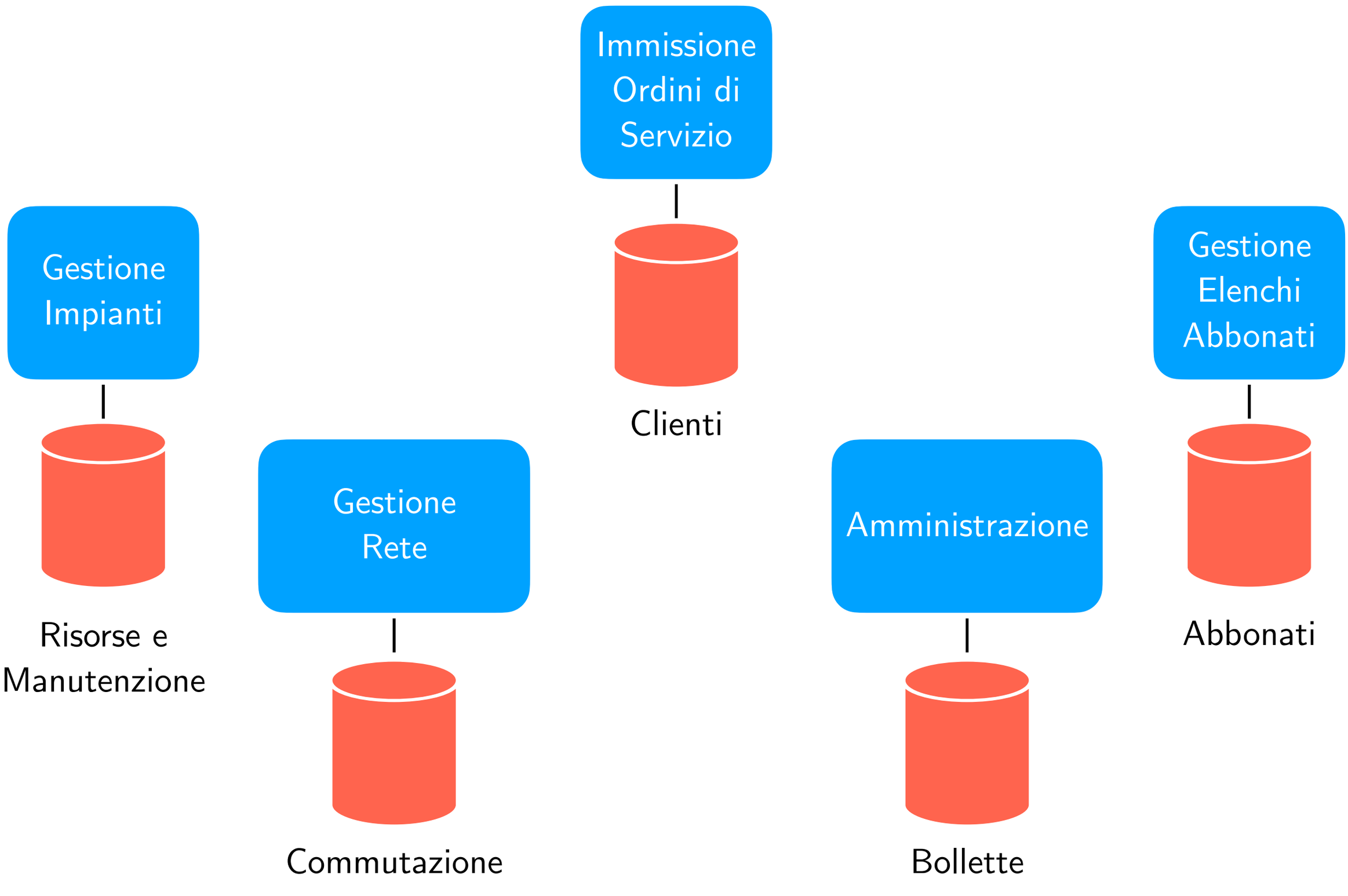
OPPURE

- Una transazione è una **successione di comandi/ operazioni in esecuzione** che forma **un'unità logica** di elaborazione sulla base di dati

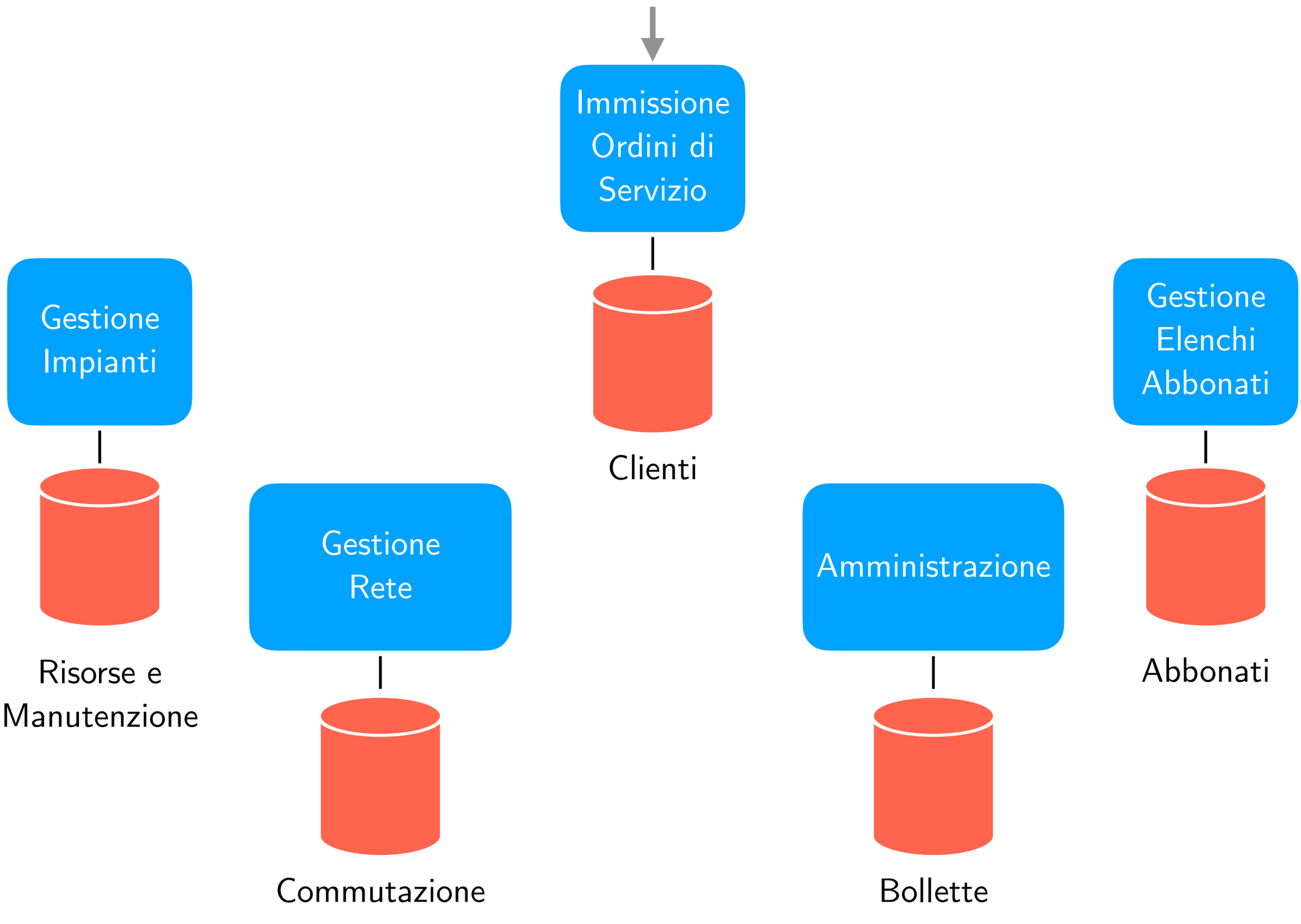
Transazione

- Una **transazione** comprende **una o più operazioni di accesso** alla base di dati:
  - **inserimenti**, **cancellazioni**, **modifiche** o **interrogazioni**
  - quindi una **sequenza** di **letture** (***read***) e **scritture** (***write***)
- Un sistema che mette a disposizione un meccanismo per la definizione e l'esecuzione di transazioni è detto **sistema transazionale**
- Un sistema transazionale è in grado di definire ed eseguire transazioni per conto di applicazioni concorrenti

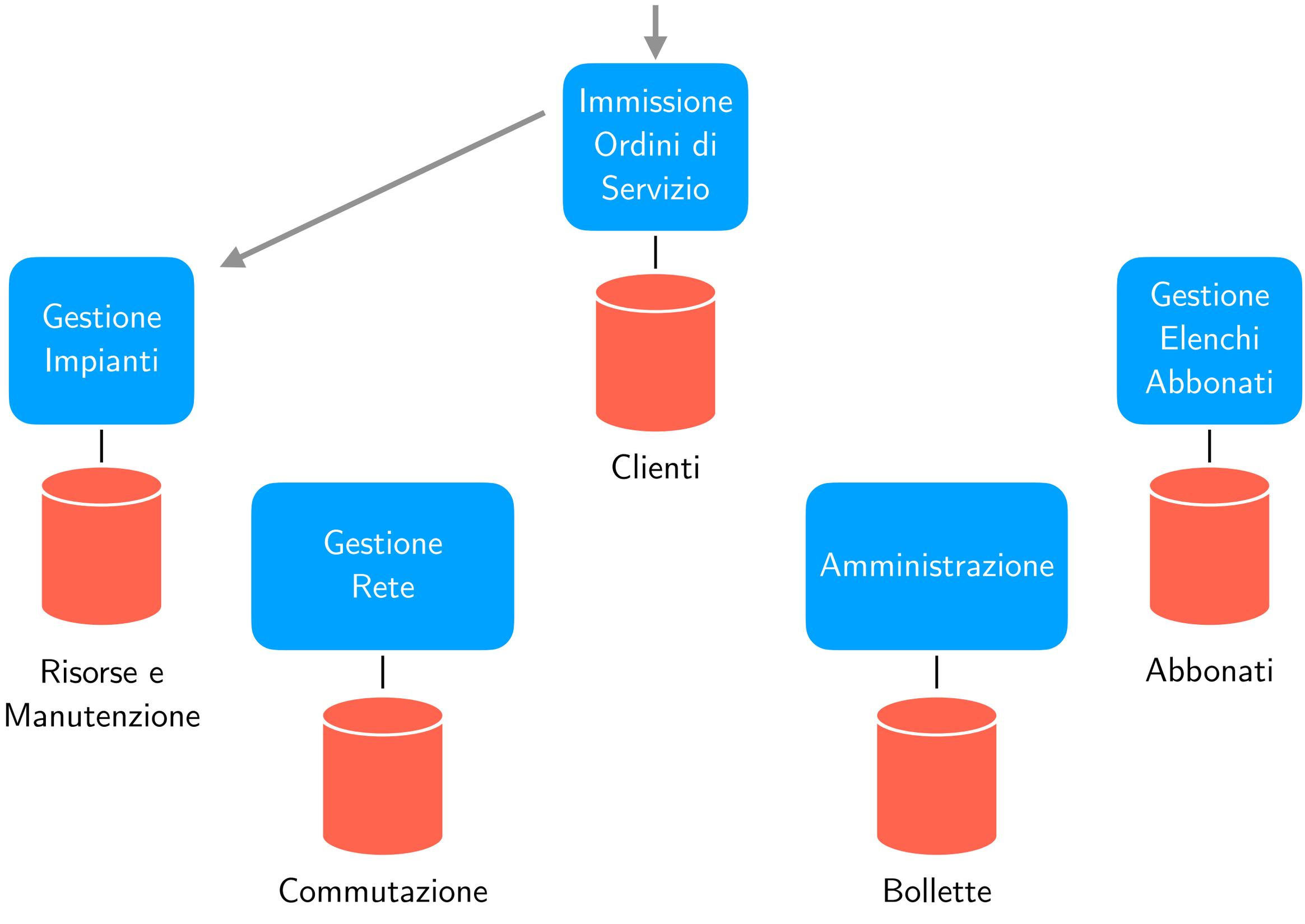
Esempio


Immissione Ordini di Servizio
Clienti
Gestione Impianti
Risorse e Manutenzione
Gestione Rete
Commutazione
Amministrazione
Bollette
Gestione Elenchi Abbonati
Abbonati

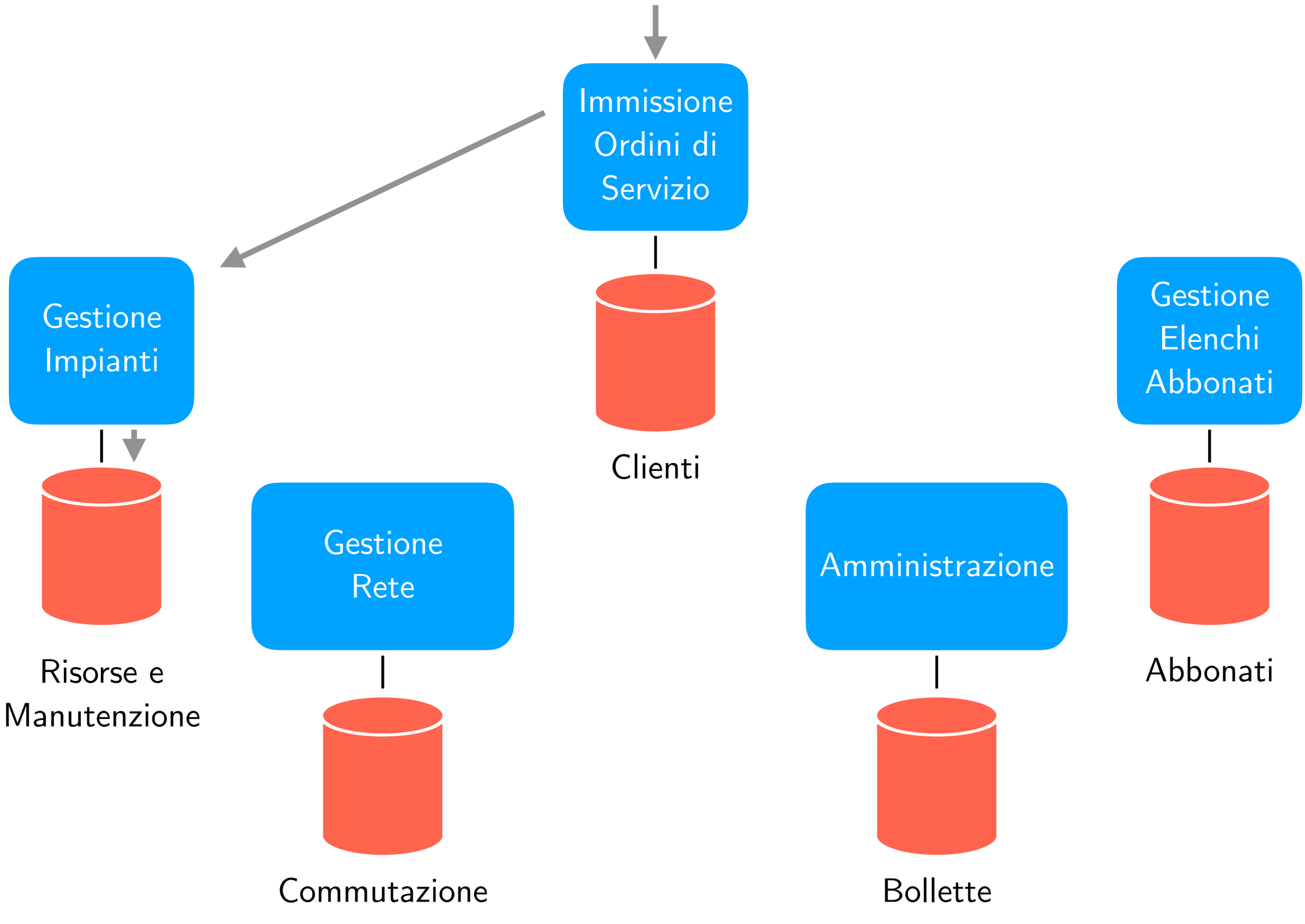
Esempio


Immissione Ordini di Servizio
Clienti
Gestione Impianti
Risorse e Manutenzione
Gestione Rete
Commutazione
Amministrazione
Bollette
Gestione Elenchi Abbonati
Abbonati

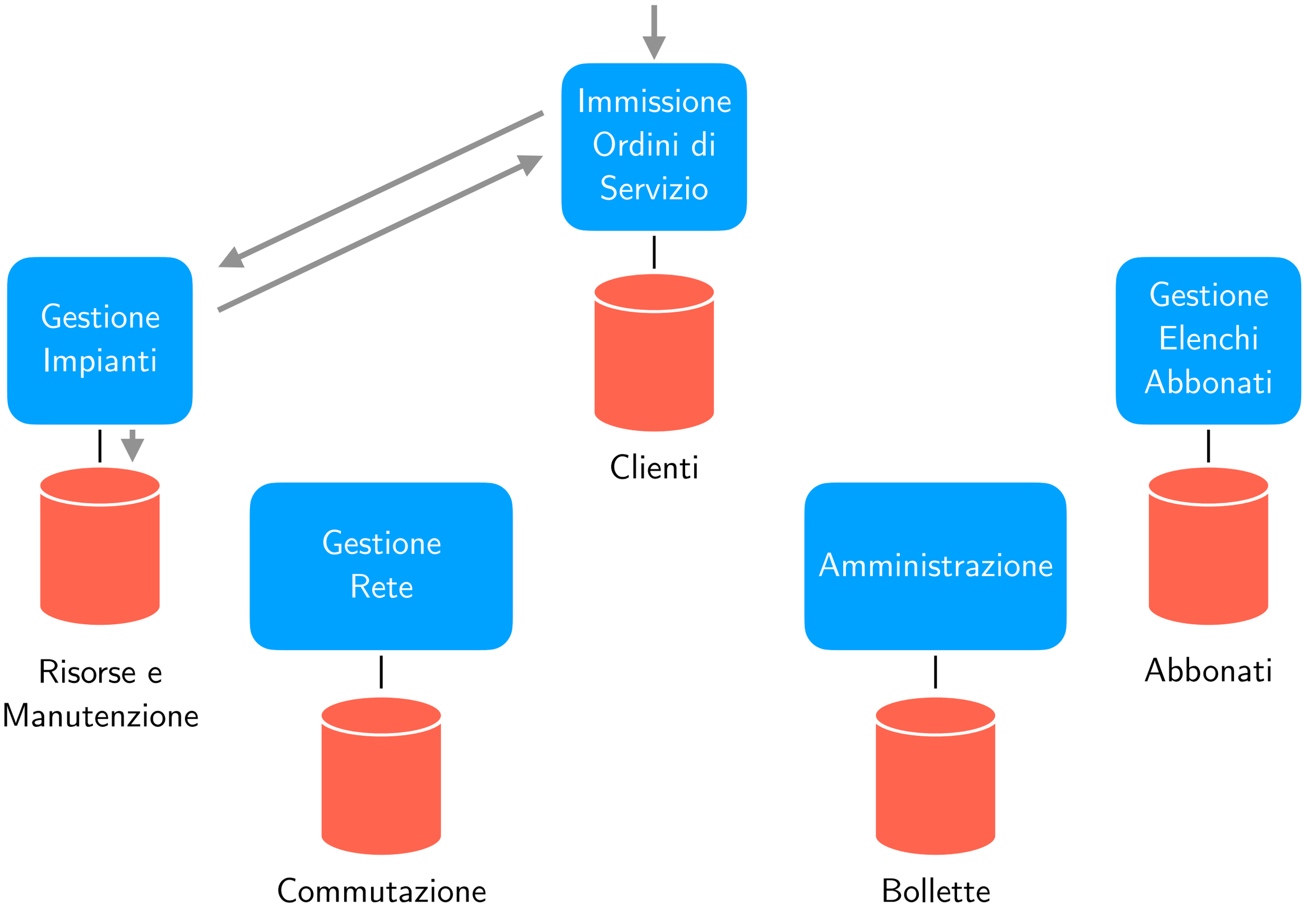
Esempio


Immissione Ordini di Servizio
Clienti
Gestione Impianti
Risorse e Manutenzione
Gestione Rete
Commutazione
Amministrazione
Bollette
Gestione Elenchi Abbonati
Abbonati

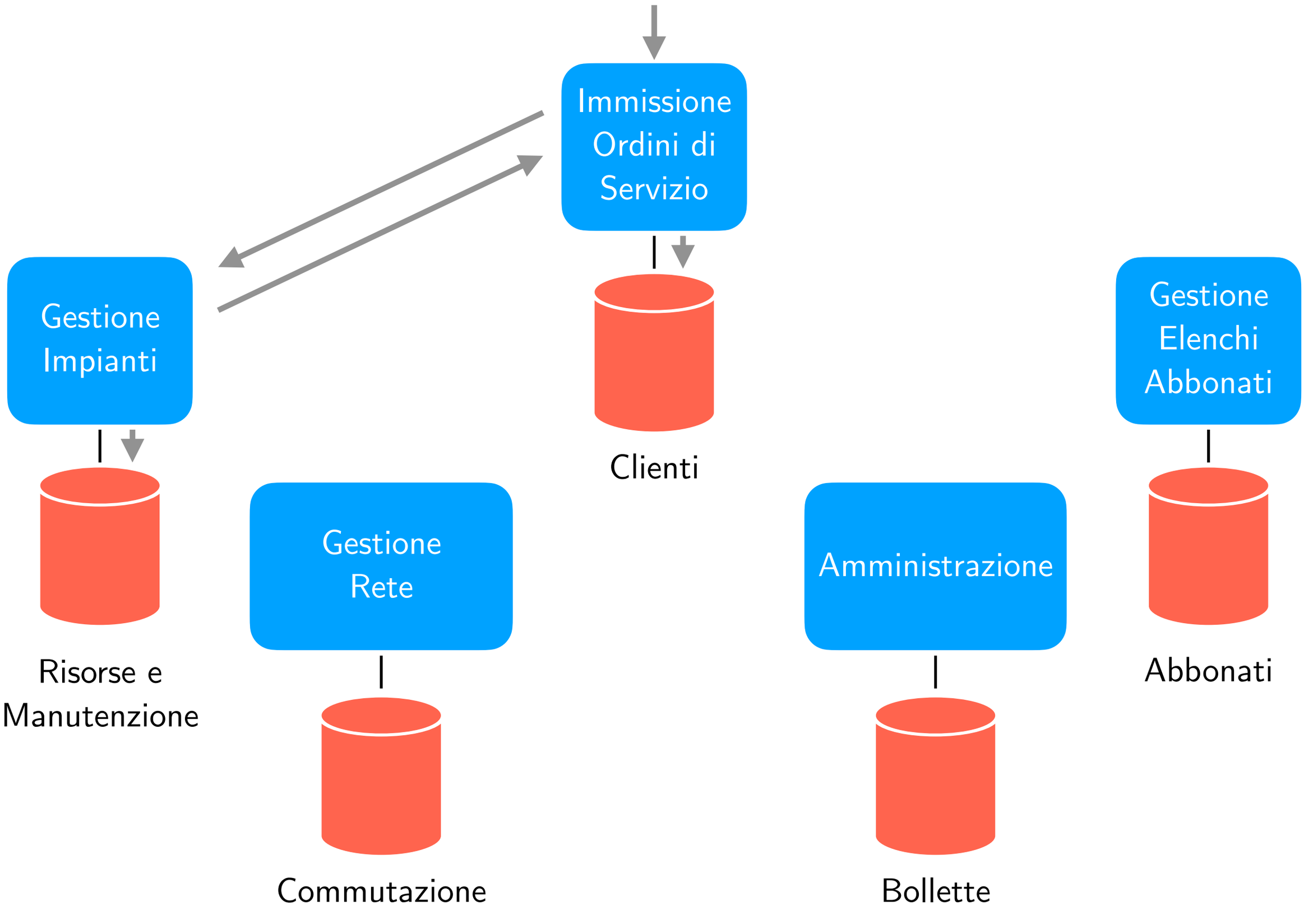
Esempio


Immissione Ordini di Servizio
Clienti
Gestione Impianti
Risorse e Manutenzione
Gestione Rete
Commutazione
Amministrazione
Bollette
Gestione Elenchi Abbonati
Abbonati

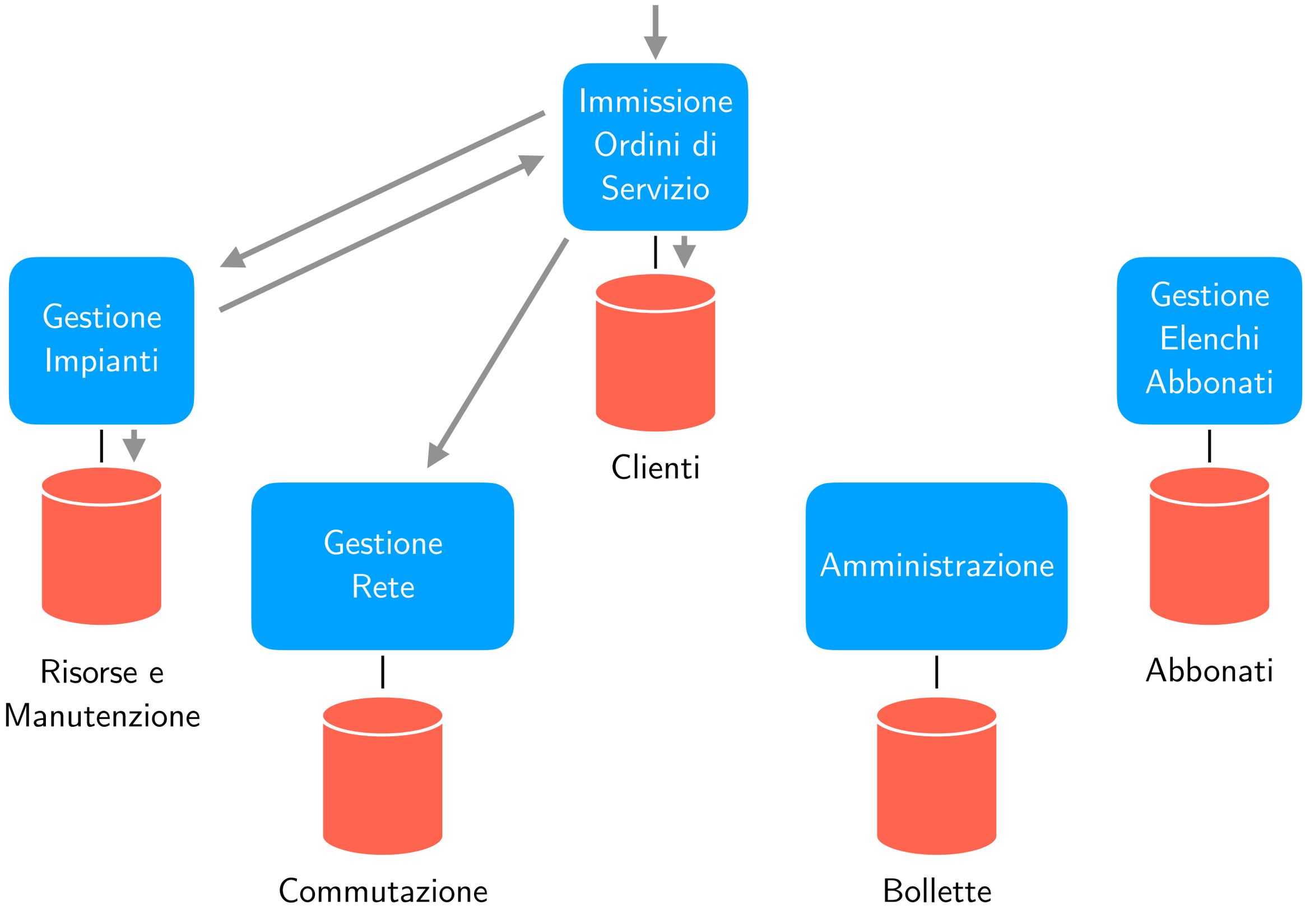
Esempio


Immissione Ordini di Servizio
Gestione Impianti
Clienti
Gestione Elenchi Abbonati
Risorse e Manutenzione
Gestione Rete
Amministrazione
Abbonati
Commutazione
Bollette

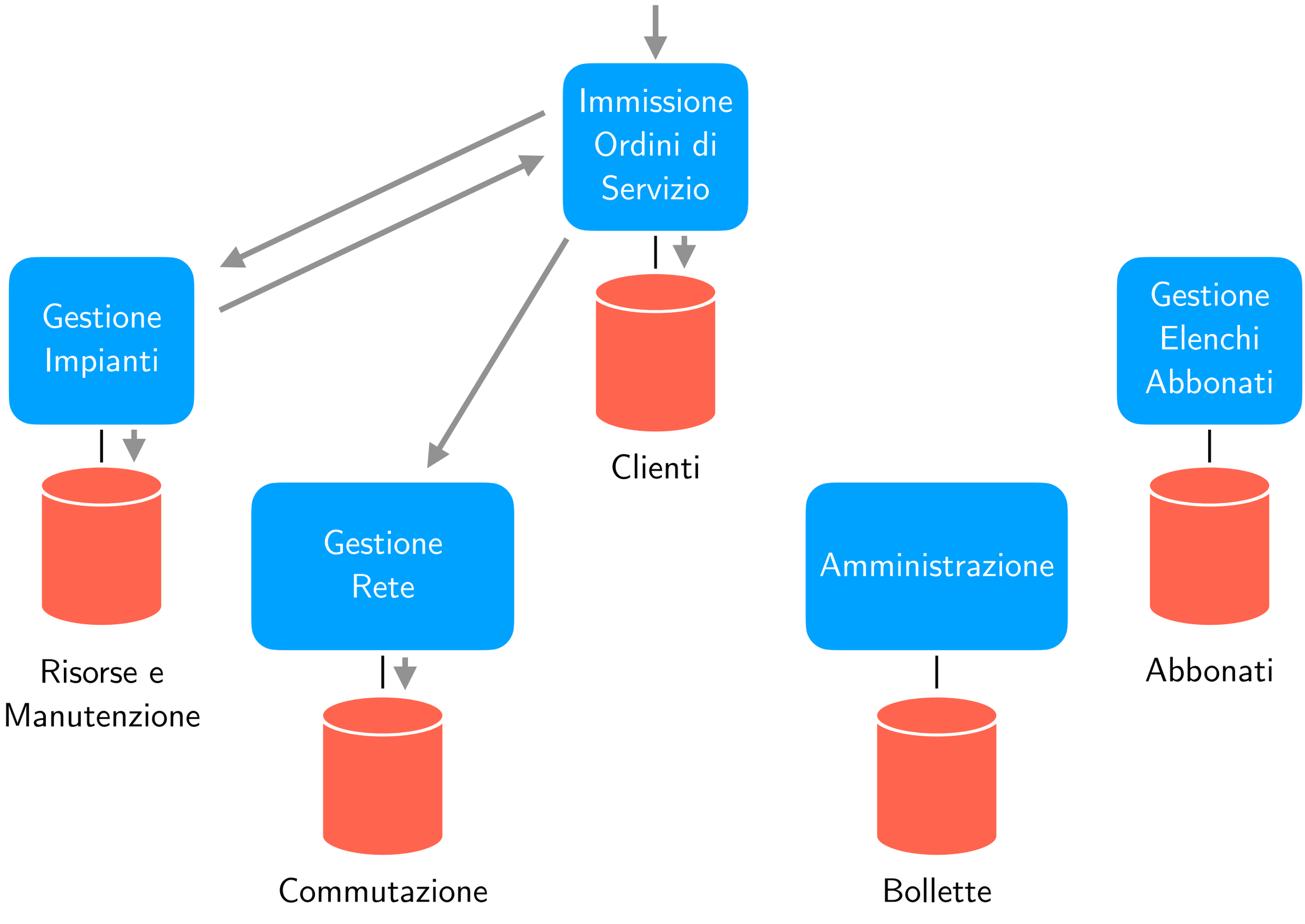
Esempio


Immissione Ordini di Servizio
Gestione Impianti
Clienti
Gestione Elenchi Abbonati
Risorse e Manutenzione
Gestione Rete
Amministrazione
Abbonati
Commutazione
Bollette

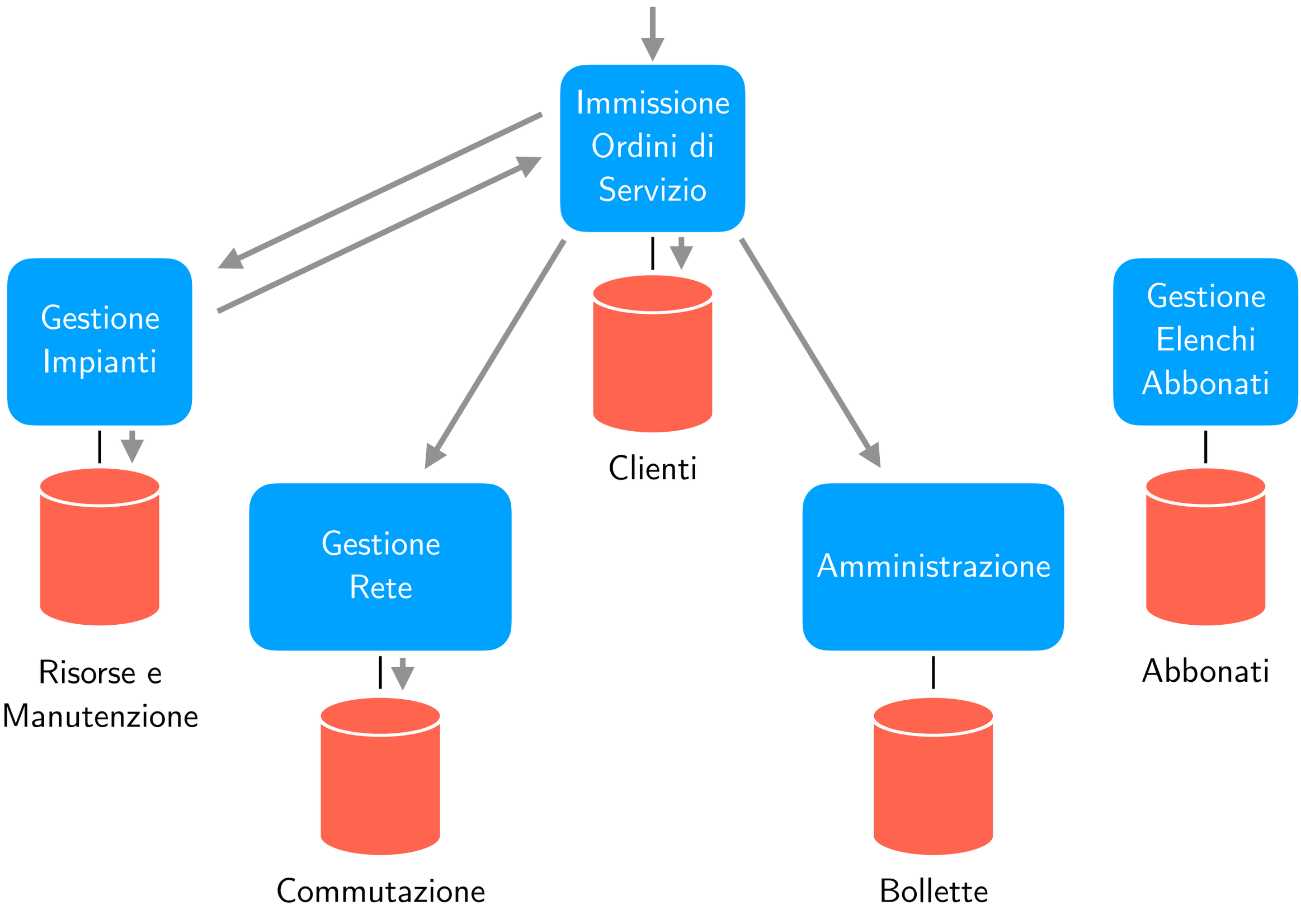
Esempio


Immissione Ordini di Servizio
Gestione Impianti
Clienti
Gestione Elenchi Abbonati
Risorse e Manutenzione
Gestione Rete
Amministrazione
Abbonati
Commutazione
Bollette

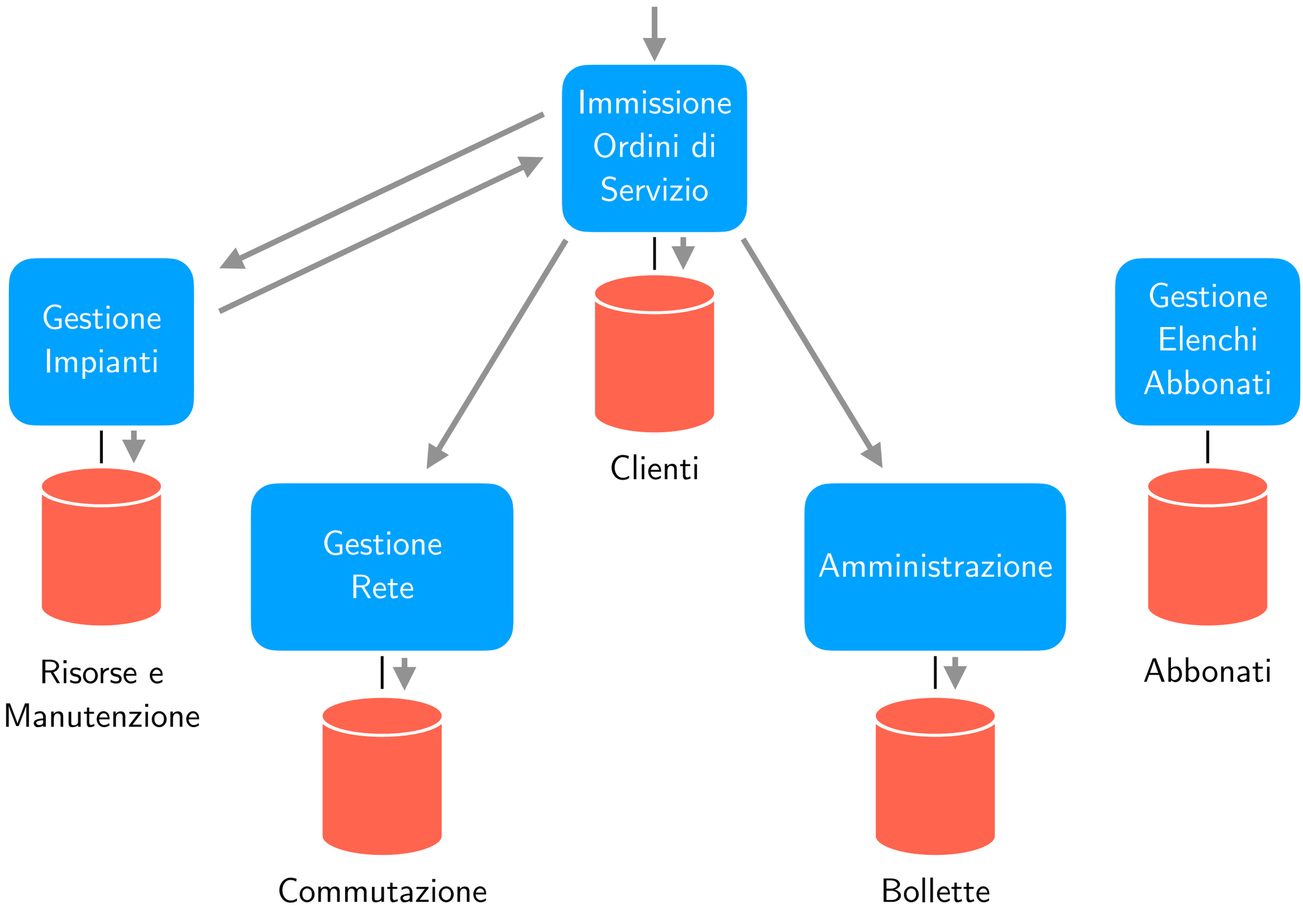
Esempio


Immissione Ordini di Servizio
Gestione Impianti
Clienti
Gestione Elenchi Abbonati
Risorse e Manutenzione
Gestione Rete
Amministrazione
Abbonati
Commutazione
Bollette

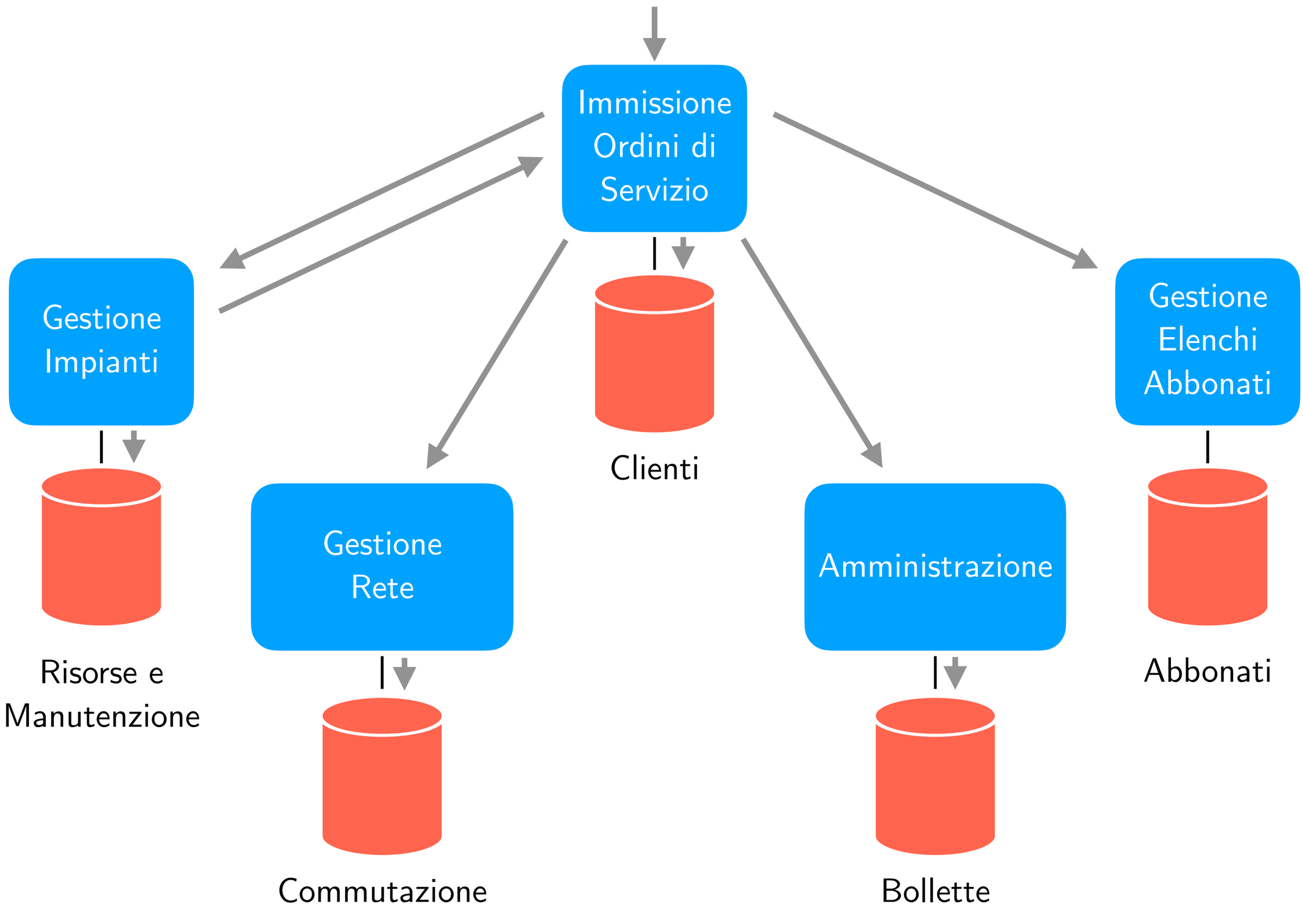
Esempio


Immissione Ordini di Servizio
Gestione Impianti
Risorse e Manutenzione
Clienti
Gestione Rete
Commutazione
Amministrazione
Bollette
Gestione Elenchi Abbonati
Abbonati

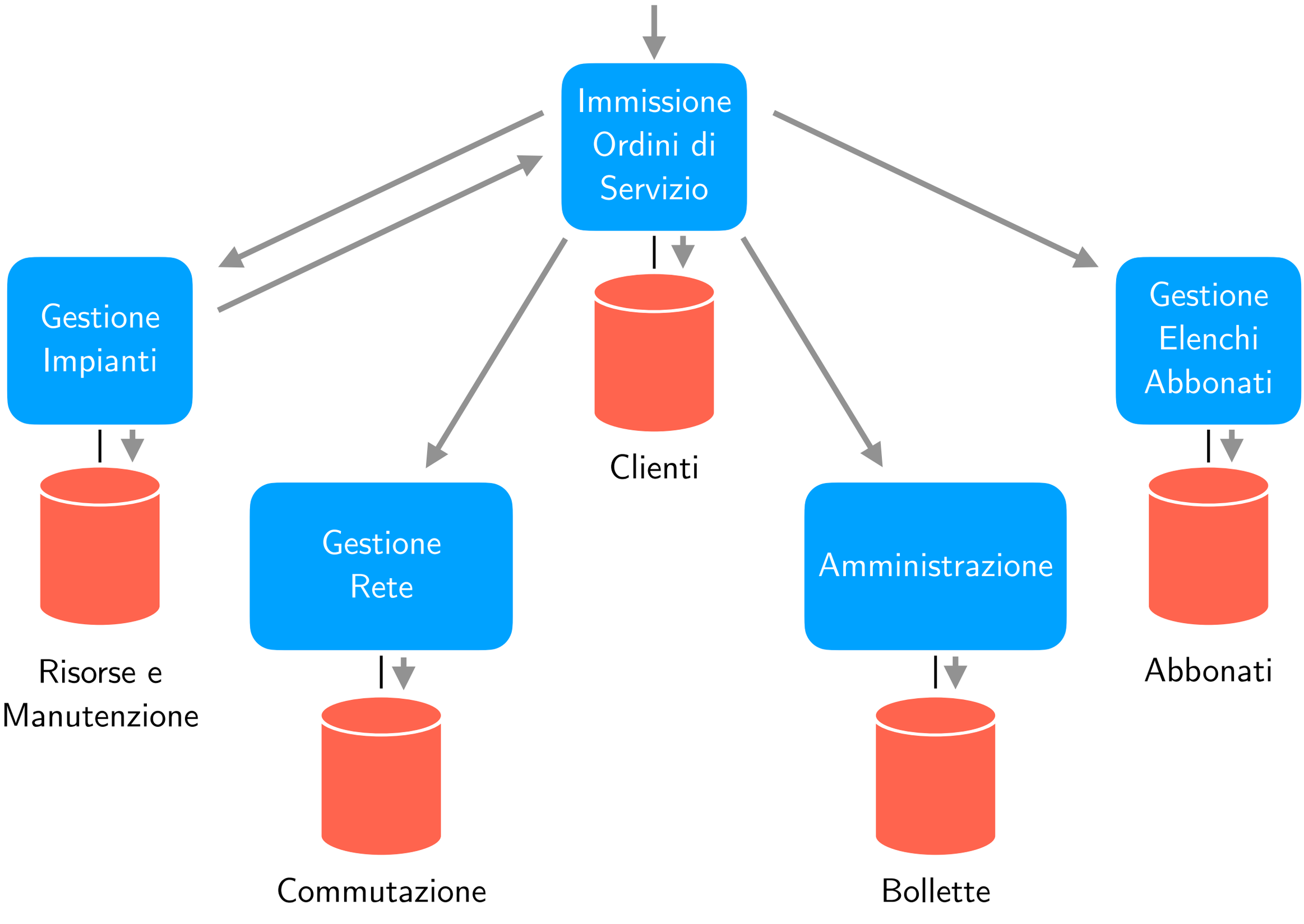
Esempio


Immissione Ordini di Servizio
Gestione Impianti
Risorse e Manutenzione
Clienti
Gestione Rete
Commutazione
Amministrazione
Bollette
Gestione Elenchi Abbonati
Abbonati

Esempio


Immissione Ordini di Servizio
Gestione Impianti
Gestione Elenchi Abbonati
Clienti
Risorse e Manutenzione
Gestione Rete
Amministrazione
Abbonati
Commutazione
Bollette

Esempio

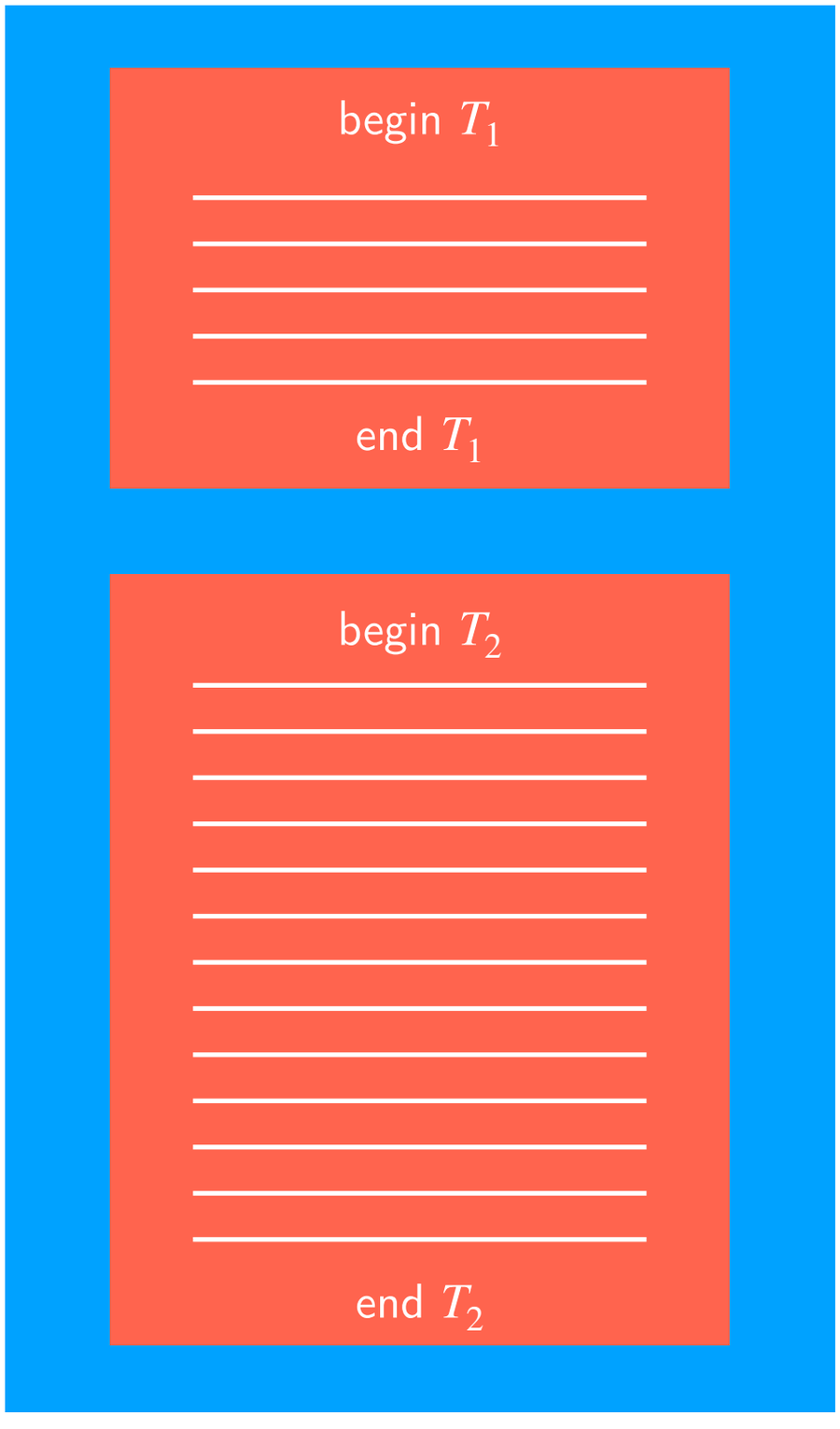

Immissione Ordini di Servizio
Gestione Impianti
Gestione Elenchi Abbonati
Clienti
Risorse e Manutenzione
Gestione Rete
Amministrazione
Abbonati
Commutazione
Bollette

Definizione di Transazione

- Una **transazione** è parte di **programma** caratterizzata da un **inizio**, una **fine** e al cui interno deve essere eseguito **una e una sola** volta uno dei seguenti comandi
  - `commit` per terminare correttamente
  - `rollback/abort` per abortire la transazione
- In SQL:
  - inizio: begin-transaction, start transaction
  - fine: end-transaction, non esplicitata
- Un **sistema transazionale** (*online transaction processing*, **OLTP**) è in grado di definire ed eseguire transazioni per conto di un certo numero di applicazioni concorrenti

Differenza tra Programma e Transazione

Programma applicativo



Transazione T_1

Transazione T_2

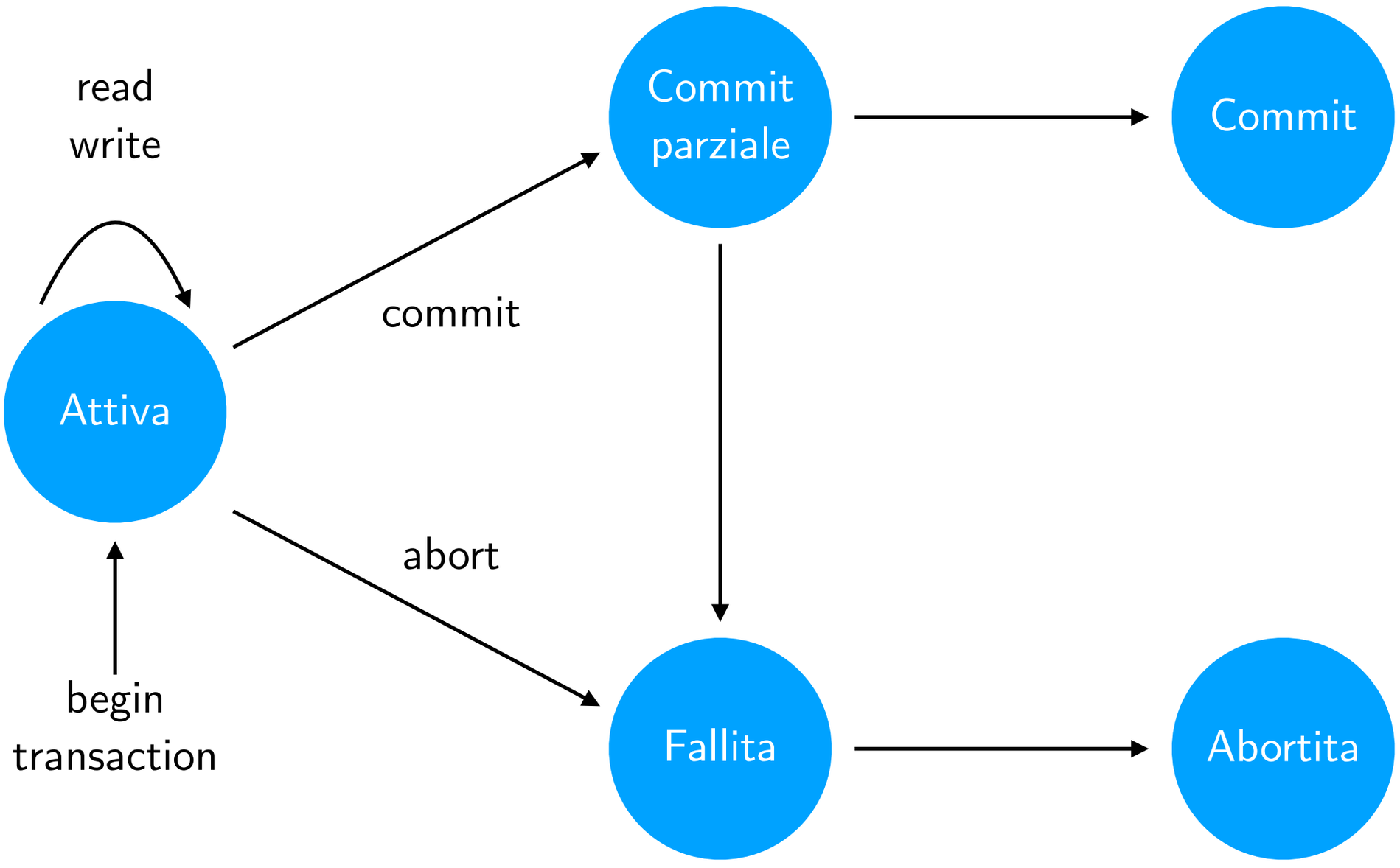
Esempio di Transazione

```
start transaction;
update ContoCorrente
   set Saldo = Saldo + 10
      where NumConto = 12202;
update ContoCorrente
   set Saldo = Saldo - 10
      where NumConto = 42177;
commit work;
```

Esempio di Transazione

```
start transaction;
update ContoCorrente
  set Saldo = Saldo + 10 where NumConto = 12202;
update ContoCorrente
  set Saldo = Saldo - 10 where NumConto = 42177;
select Saldo as A
  from ContoCorrente
  where NumConto = 42177;
if (A >= 0) then commit work
       else rollback work;
```

Stato di una transazione



Stato di una transazione

- Una transazione entro nello stato **attivo** subito dopo essere iniziata, dove rimane per tutta la sua esecuzione
- Una transazione si sposta nello stato **commit parziale** quando termina
- Una transazione si sposta nello stato **commit** se è stata eseguita con successo e tutti i suoi aggiornamenti alla base di dati sono permanenti
- Una transazione si sposta nello stato **fallito** quando non può passare da commit parziale a commit o se se è stata interrotta in seguito a un fallimento mentre era nello stato attivo
- Una transazione si sposta nello stato **abortito** se è stata interrotta e tutti i suoi aggiornamenti alla base di dati sono stati annullati

Proprietà delle Transazioni

- **A**tomicità
- **C**onsistenza
- **I**solamento
- **D**urabilità (o persistenza)

Atomicità

- Una transazione è una **unità atomica di elaborazione**
- Non può lasciare la base di dati in uno **stato intermedio**
  - un guasto o un errore prima del commit debbono causare **l'annullamento** (***UNDO***) delle operazioni svolte
  - un guasto o errore dopo il commit non deve avere conseguenze; se necessario vanno **ripetute** (***REDO***) le operazioni
- Esito della transazione
  - **Commit**
    - caso "normale" e più frequente (99%)
  - **Rollback** (o **abort**)
    - richiesto dall'applicazione
    - richiesto dal sistema
      - violazione dei vincoli, concorrenza, incertezza in caso di fallimento

Consistenza

- La transazione **rispetta i vincoli di integrità**
  - se lo **stato iniziale** è **corretto**
  - allora anche lo **stato finale** è **corretto**
- Conseguenza
  - **All'inizio** ed **alla fine** della transazione il sistema è in uno **stato "consistente"**
  - **Durante** l'esecuzione della transazione stessa il sistema può **temporaneamente** essere in uno stato **inconsistente**

Isolamento

- La transazione **non risente degli effetti** delle altre **transazioni concorrenti**
  - **L'esecuzione concorrente** di una collezione di transazioni deve produrre un risultato che si potrebbe ottenere con una **esecuzione sequenziale**
- Conseguenza:
  - Una transazione non espone i suoi **stati intermedi**
    - Si evita un "**effetto domino**"

Durabilità (o persistenza)

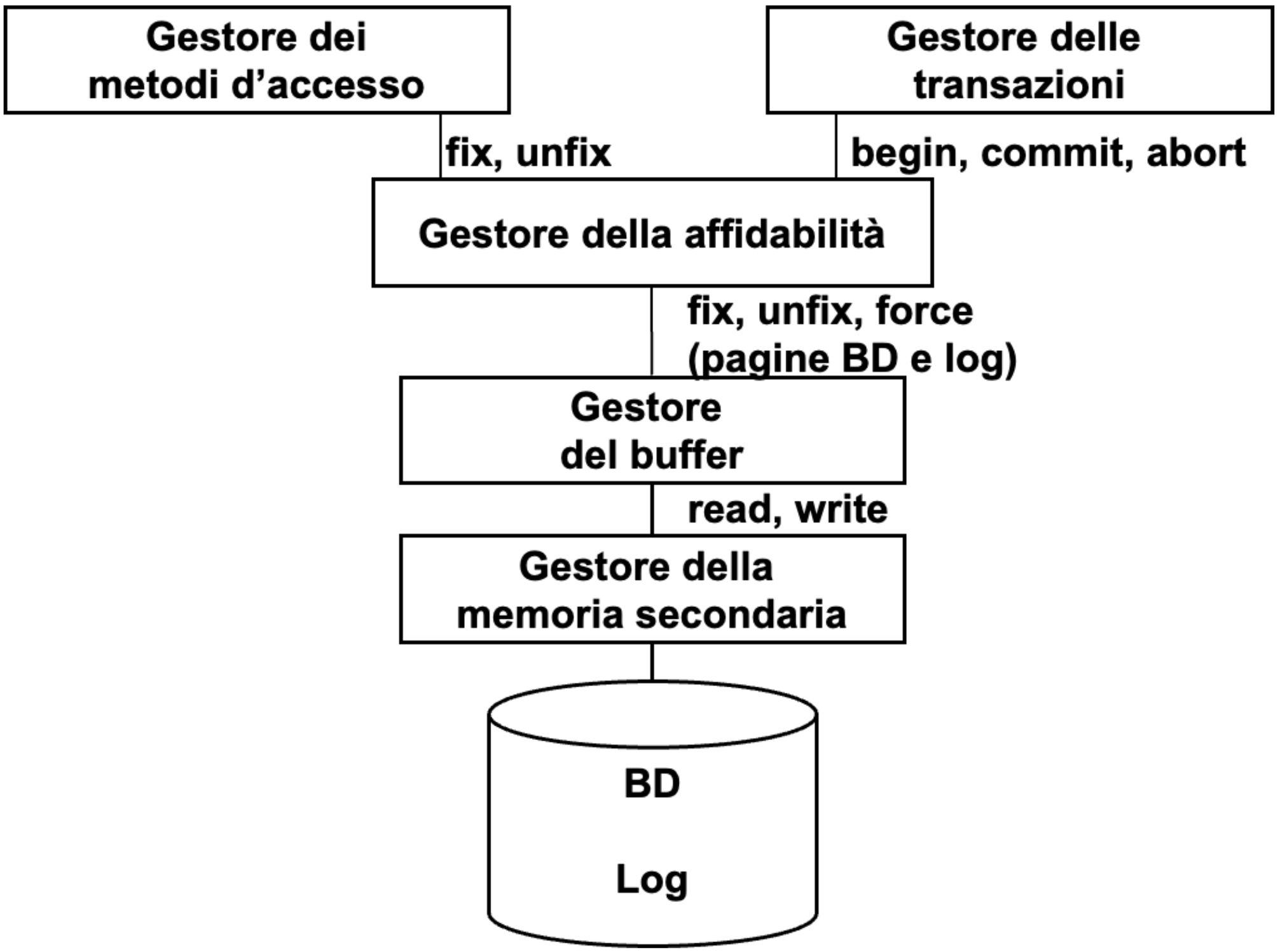
- Gli effetti di una transazione andata in commit non vanno perduti ("durano per sempre"),
- Conseguenza
  - I guasti non hanno effetto sullo stato del database: uno stato consistente sarà sempre recuperato

Gestione della affidabilità

Gestore dell'affidabilità

- Gestisce **l'esecuzione** dei **comandi transazionali**
  - *start transaction* (B, *begin*)
  - *commit work* (C)
  - *rollback work* (A, *abort*)
- e le operazioni di **ripristino** (*recovery*) dopo i guasti :
  - ripresa **a caldo** (*warm restart*)
  - ripresa **a freddo** (*cold restart*)
- Assicura **atomicità** e **durabilità**
- Usa il **log**:
  - Un archivio permanente che registra le operazioni svolte

Architettura del gestore dell'affidabilità



Persistenza delle memorie

- **Memoria centrale**: è persistente, l'informazione viene distrutta da qualunque guasto del sistema
- **Memoria di massa**: è persistente, sopravvive ai guasti di sistema, ma può danneggiarsi l'unità di memorizzazione
- **Memoria stabile**: memoria che non può danneggiarsi (è una astrazione):
  - perseguita attraverso la ridondanza:
    - dischi replicati
    - nastri
    - ...
  - con probabilità di fallimento indipendenti

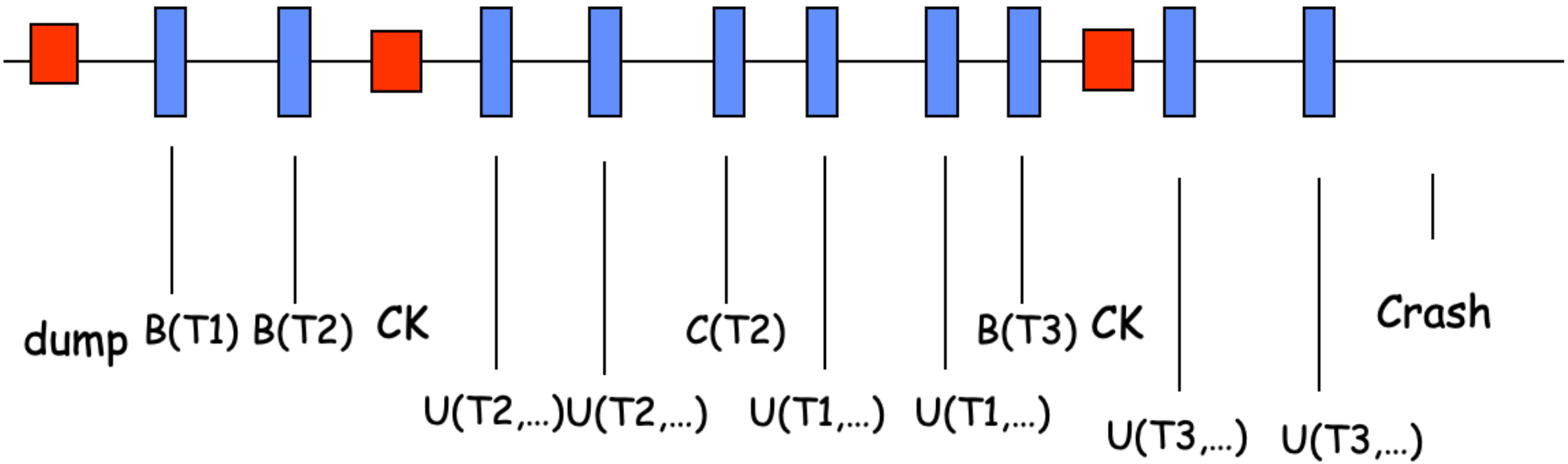
Log

- Il **log** è un **file sequenziale** gestito dal gestore dell'affidabilità, scritto in **memoria stabile**
- "Diario di bordo": **riporta tutte le operazioni in ordine**
- **Record** nel log
  - **operazioni delle transazioni**:
    - `begin`, $B(T)$
    - `insert`, $I(T, O, AS)$
      - nel record O, T memorizza per la prima volta l'***after state*** AS
    - `delete`, $D(T, O, BS)$
      - nel record O, T elimina per sempre il ***before state*** BS
    - `update`, $U(T, O, BS, AS)$
      - nel record O, T elimina il ***before state*** BS e memorizza l'***after state*** AS
    - `commit`, $C(T)$
    - `abort`, $A(T)$
  - **record di sistema**:
    - `dump`
    - `checkpoint`

Esempio di log

• begin transaction		$B(T_1)$
• $w[A]$	$A = 50$	$I(T_1, A,50)$
• $w[A]$	$A = 20$	$U(T_1, A,50,20)$
• $r[A]$	$A = 20$	nessuna modifica
• $r[B]$	$B = 50$	nessuna modifica
• $w[B]$	$B = 80$	$U(T_1, B,50,80)$
• commit		$C(T_1)$

Struttura del Log



Checkpoint e Dump

- Il log serve a "**ricostruire**" le **operazioni**
- **Checkpoint** e **dump** servono ad evitare che la ricostruzione debba partire dall'inizio dei tempi
  - si usano con riferimento a tipi di guasti diversi
- L'**operazione di checkpoint** serve a "fare il punto" della situazione, semplificando le successive operazioni di ripristino
  - Ha lo scopo di **registrare** quali **transazioni** sono **attive** in un **certo istante**, cioè le transazioni "a metà strada"
  - Ha lo scopo duale di **confermare** che le **altre** o **non sono iniziate** o sono **finite**
    - Per tutte le transazioni che hanno **effettuato** il **commit** i **dati** possono essere **trasferiti** in **memoria di massa**

Operazione di checkpoint

- **Varie modalità**, vediamo la più semplice:
  - Si **sospende l'accettazione** delle operazioni di **commit** o **abort** da parte delle transazioni
  - Si **forza** (*force*) la **scrittura** in **memoria di massa** delle pagine in memoria modificate da **transazioni** che hanno già fatto **commit**
  - Si **forza** (*force*) la **scrittura** nel **log** di un record contenente gli identificatori delle **transazioni attive**
  - Si **riprendono ad accettare** tutte le operazioni da parte delle transazioni
- Con questo funzionamento si **garantisce la persistenza** delle transazioni che hanno eseguito il commit

Dump

- **Copia completa** ("di riserva", *backup*) della base di dati
  - Solitamente prodotta mentre il **sistema non è operativo**
  - Salvato in **memoria stabile**, come backup
  - Un **record di dump** nel log indica il momento in cui il log è stato effettuato
    - e dettagli pratici, file, dispositivo, ...