Basi di Dati

Progetto A.A. 2022/2023

PERSONAL TRAINER DIGITALE

0292536

Massimo Buniy

Indice

1. Descrizione del Minimondo

Si vuole realizzare un'applicazione per consentire, ai personal trainer di una palestra, di gestire le schede degli esercizi dei propri clienti.

Ciascun utente della palestra è associato ad un personal trainer di riferimento. Questi potrà redigere una scheda di allenamento personalizzata per ciascun utente, scegliendo gli esercizi e i macchinari da utilizzare in un insieme definito dal proprietario della palestra. Per ciascun esercizio, il personal trainer indicherà anche il numero di serie e ripetizioni dell'esercizio.

Periodicamente, quando viene redatta una nuova scheda di esercizi, la scheda precedente viene archiviata. Questa sarà ancora consultabile dagli utenti, ma essi potranno "interagire" solo con quella corrente.
Gli utenti della palestra possono infatti effettuare il login nell'applicazione e mostrare quali sono gli esercizi che devono svolgere in una sessione di allenamento. Il sistema mostrerà l'esercizio e la serie corrente, permettendo all'utente di contrassegnare un dato esercizio come completato. L'atleta ha la possibilità di saltare un esercizio e passare al successivo.

I personal trainer possono generare un report che mostra, per tutti i clienti a loro assegnati, quanti sono gli allenamenti sostenuti in un intervallo temporale richiesto, qual è la percentuale di completamento delle schede di allenamento in ogni sessione di allenamento e quanto tempo è durato ciascun allenamento.

2. Analisi dei Requisiti

Identificazione dei termini ambigui e correzioni possibili

Linea	Termine	Nuovo termine	Motivo correzione
4, 5, 10, 12, 14	Utente	Cliente	Mantenere un solo modo di chiamare chi si iscrive in palestra, per non genere ambiguità
4	Questi	Il personal trainer	Esplicitare il soggetto che andrà a redigere la scheda di allenamento
2, 9	Scheda di esercizi	Scheda di allenamento	Per evitare disambiguità rispetto al termine precendente
10	"interagire"	avviare una sessione di allenamento	Interagire troppo vago. Vado a specificare l'azione che l'utente può fare con la scheda essendo solo quella di segnare gli esercizi svolti durante una sessione di allenamento.
13	esercizi	Scheda di allenamento	Se li lasciassimo esercizi potremmo pensare che un cliente vede solo e soltanto gli esercizi che deve fare e non anche i macchinari
15	Atleta	Cliente	Mantere un solo nome per l'identificazione di uno stesso tipo di utente, per evitare possibili incomprensioni future
20	Allenamento	Sessione di allenamento	Disambiguazione, si fa riferimento alla singola sessione

Specifica disambiguata

Si vuole realizzare un'applicazione per consentire, ai personal trainer di una palestra, di gestire le schede di allenamento dei propri clienti.
Il personal trainer potrà redigere una scheda di allenamento personalizzata per ciascun cliente. Nella scheda di allenamento di ciascun cliente indicherà gli esercizi da fare e i macchinari da utilizzare, con il relativo numero di serie e di ripetizioni.
Gli esercizi e i macchinari sono scelti da un insieme definito dal proprietario della palestra. Periodicamente, quando viene redatta una nuova scheda di allenamento, la scheda precedente viene archiviata.
Ciascun cliente ha una propria scheda di allenamento redatta dal proprio personal trainer.
Il cliente è in grado di consultare sia la scheda di allenamento corrente che le schede di allenamento precedenti, ma può avviare una sessione di allenamento solo con quella corrente.
I clienti possono effettuare il login nell'applicazione e vedere la scheda di allenamento che devono svolgere in una sessione di allenamento. Il sistema mostrerà l'esercizio e la serie corrente. Il cliente può contrassegnare un dato esercizio come completato o saltarlo e passare al successivo. Ciascun cliente della palestra è associato ad un personal trainer di riferimento.

Il sistema tiene traccia delle sessioni di allenamento sostenute dal cliente per una determinata scheda di allenamento.

I personal trainer possono generare un report che dato un intervallo di tempo restituisca per i clienti a lui assegnati:

- il numero di allenamenti sostenuti
- la percentuale di completamento della scheda di allenamento associata alla sessione di allenamento
- la durata della sessione di allenamento

Glossario dei Termini

Termine	Descrizione	Sinonimi	Collegamenti
Personal trainer	Colui che gestisce gli allenamenti dei clienti a lui associati.		Scheda degli esercizi Cliente Macchinari Report
Cliente	Utilizzatore della palestra		Scheda degli esercizi Personal trainer
Scheda di allenamento	Una lista di esercizio, macchinario, numero di serie e ripetizioni per l'esercizio		Cliente Personal Trainer Macchinario
Macchinario	Macchinario che un cliente può utilizzare poiché messo a disposizione		Proprietario Personal Trainer Cliente Scheda di allenamento
Esercizio	Esercizio che un cliente può svolgere		Proprietario Personal Trainer Cliente Scheda di allenamento

Raggruppamento dei requisiti in insiemi omogenei

Frasi relative a Cliente
Ciascun cliente della palestra è associato a un personal trainer di riferimento. Il cliente è in grado di consultare sia la scheda di allenamento corrente che le schede di allenamento precenti. Ciascun cliente ha una propria scheda di allenamento redatta dal proprio personal trainer. [Ciascun cliente] […] può avviare una sessione di allenamento solo sulla scheda di allenamento corrente. I clienti possono effettuare il login. […] [può] vedere la scheda di allenamento che deve svolgere in una sessione di allenamento. I clienti possono vedere […] la scheda di allemanto che deve svolgere in una sessione di allenamento. Il cliente può contrassegnare un dato esercizio come completato o saltarlo e passare al successivo.

Frasi relative a Personal Trainer
Il personal trainer potrà redigere una scheda di allenamento personalizzata per ciascun cliente. [Il personal trainer] nella scheda di allenamento di ciascun cliente indicherà gli esercizi da fare e i macchinari da utilizzare, con il relativo numero di serie e di ripetizioni. Il personal trainer può redigere un report.

Frasi relative a Scheda di allenamento
[…] [è] personalizzata per ciascun cliente. […] [contiente] gli esercizi da fare e i macchinari da usare, con il relativo numero di serie e di ripetizioni.

Frasi relative a Esercizio
Gli esercizi […] sono scelti da un insieme definito dal proprietario della palestra. Nella scheda di allenamento di ciascun cliente […] [ci sono] gli esercizi da fare […],con il relativo numero di serie e di ripetizioni.

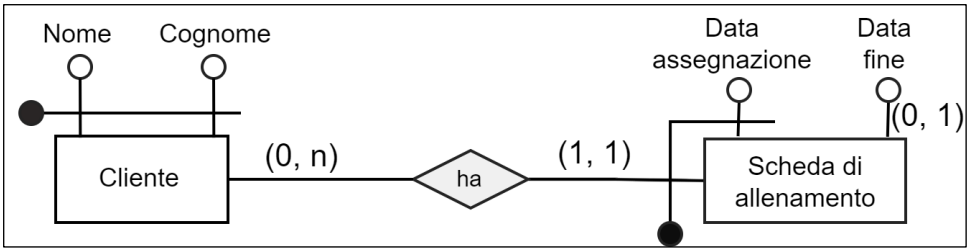
Frasi relative a Macchinario
[…] i macchinari sono scelti da un insieme definito dal proprietario della palestra. Nella scheda di allenamento di ciascun cliente […] [ci sono] […] i macchinari da utilizzare, con il relativo numero di serie e di ripetizioni.

3. Progettazione concettuale

Costruzione dello schema E-R

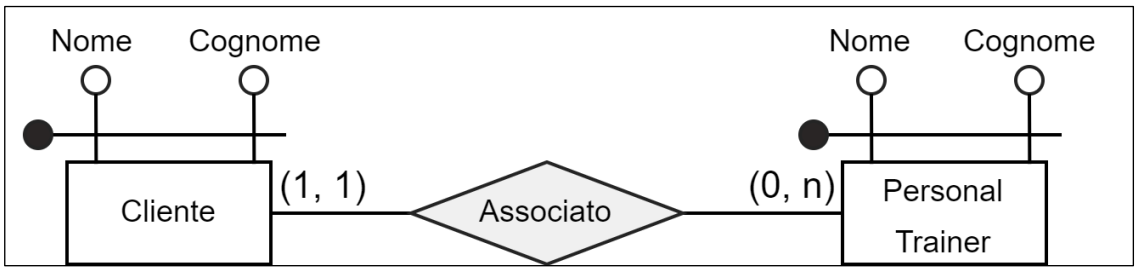
Per la costruzione dello schema E-R son partito individuando le principali entità dello schema che sono: Personal Trainer, Cliente, Proprietario, Scheda di allenamento, Esercizio e Macchinario.
Dopodiché analizzano le frasi della specifica ho creato degli schemi parziali, definendo anche le relazioni tra le entità trovate.

- Per prima sono andato a realizzare la frase "ciascun cliente ha una propria scheda di allenamento":



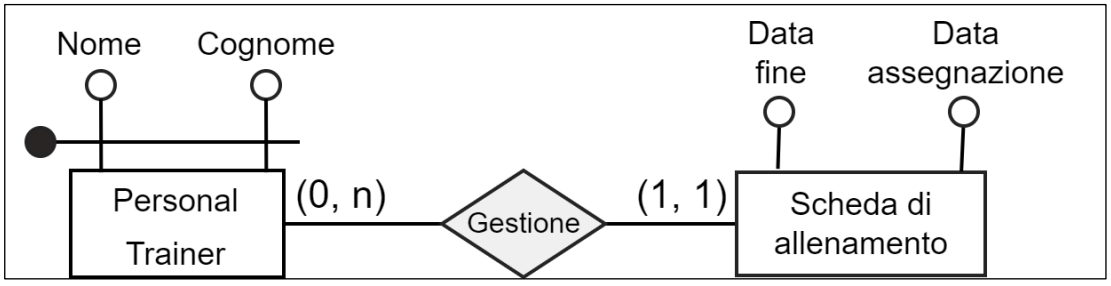
"Cliente" partecipa alla relazione con cardianalità (0, n) poiché un cliente appena iscritto non possiede ancora una scheda di allenamento.
"Scheda di allenamento" partecipa alla relazione con cardinalità (1, 1) poiché non può esserci una scheda di allenamento senza che ci sia un cliente e ogni scheda è unica per ogni cliente. Per rafforzare inoltre questo aspetto ho deciso di rendere "Scheda di allenamento" un'entità debole vincolandola alla relazione.

- Con il prossimo schema parziale sono andato a legare "Cliente" con "Scheda di allenamento" poiché nella specifica si fa chiaramente rifarimento a questo legame:



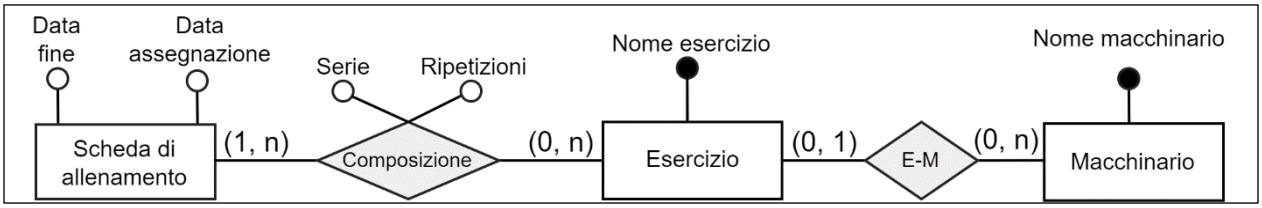
La relazione "Associato" serve a descrivere che a ogni "Cliente", con cardinalità (1, 1) visto che a ogniuno è associato un solo "Personal Trainer" di riferimento, il quale partecipa con cardinalità (0, n) poiché può momentaneamente non seguire nessuno, come seguire diverse persone.

- Ho voluto poi rappresentare in qualche modo il fatto che il personal trainer gestisca le varie schede, e che quindi sia direttamente legato a loro in qualche modo:



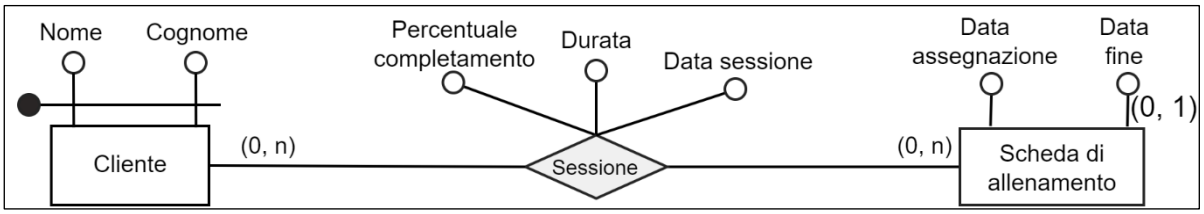
"Schede di allenamento" partecipa con cardinalità (1, 1) visto che ogniuna è redatta da un "Personal Trainer" per uno specifico "Cliente". "Personal Trainer" può gestire da 0 a n schede se al momento non ha ancora clienti assegnati o se ne ha assegnato più di uno.

- Dopodiché son passato a esplicare meglio cosa sia una "Scheda di allenamento" visto che un



"Personal Trainer" può scegliere gli esercizi solo da un determinato insieme dato:
"Scheda di allenamento", che partecipa con cardinalità (1, n), visto che per definizione deve essere composta da almeno un esercizio. "Esercizio" partecipa con cardinalità (0, n) nella relazione "Composizione" poiché possono esserci esercizi che non compaiono in nessun scheda o esercizi che compaiono in più schede.
Ho deciso di trattare "Esercizio" e "Macchiario" come due entità separate. Infatti il nome di un esercizio può essere associato a un macchinario mentre un altro potrebbe non esserlo e non aveva senso usare un macchinario senza aver un esercizio da fare su questo. Per cui "Eserczio" partecipa alla relazione "E-M" con cardinalità (0, 1), poiché non ci sono esercizi che richiedo più di un macchinario per essere svolto, e "Macchinario" con cardinalità (0, n) poiché potrebbero esserci macchinari non assegnati a nessun esercizio e macchinari assegnati a più di un esercizio.

- Ho notato poi che "Personal Trainer" ha gli stessi attributi di "Cliente". Di conseguenza queste due entità possono essere raggruppate con una generalizzazione.
- Per rappresentare una sessione di allenamento sono passato attraverso una relazione tra "Cliente" e "Scheda di allenamento". Infatti una "Sessione" è sempre associata a una scheda di allenamento e ogni scheda di allenamento è unica rispetto a un cliente. Inolte non deve

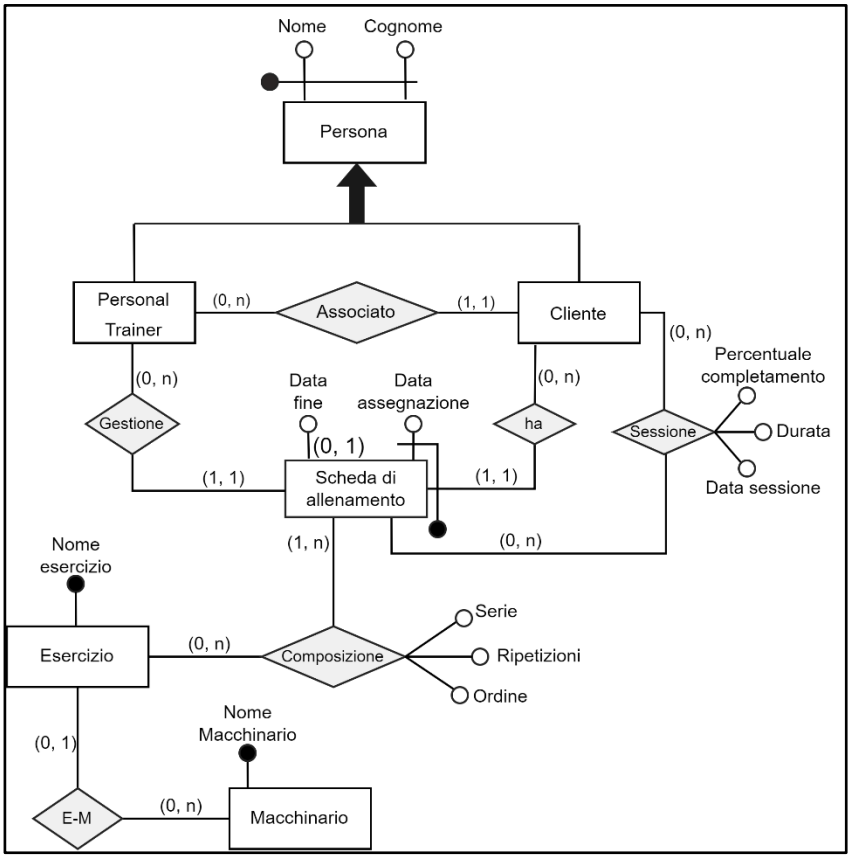


esistere una sessione di allenamento se non si ha la scheda a cui è riferita.
"Cliente" partecipa alla relazione con cardinalità (0, n) visto che potrebbe non avar ancora mai svolto una sessione. "Scheda di allenamento" invece ha cardinalità (0, n) visto che una scheda appena assegnata non è detto che sia già stata svolta, oppure data una scheda un determinato "Cliente" ci abbia svolto su diverse sessioni di allenamento.

Integrazione finale

Non vi sono conflitti sui nomi utilizzati e nemmeno conflitti strutturali.

Lo schema E-R finale, unendo i vari sottoschemi, è il seguente:



Regole aziendali

Il cliente non può avere più di una scheda attiva contemporaneamente.
Il cliente può iniziare una sessione di allenamento soltanto sulla scheda di allenamento corrente.
Il personal trainer può assegnare schede solo a clienti a lui associati.

Dizionario dei dati

Entità	Descrizione	Attributi	Identificatori
Persona	Individuo che frequenta la palestra	Nome, Cognome	Nome, Cognome
Cliente	Iscritto alla palestra		
Personal Trainer	Colui che è incaricato di generare e gestire le schede di allenamento dei clienti		
Scheda di allenamento	Gli esercizi che un cliente deve andare a svolgere quando va ad allenarsi in una sessione di allenamento	Data assegnazione, Data fine	Data assegnazione, Cliente
Esercizio	Gli esercizi messi a disposizione dal proprietario della palestra	Nome esercizio	Nome esercizio
Macchinario	Macchinari messi a disposizione dal proprietario della palestra.	Nome macchinario	Nome macchinario

4. Progettazione logica

Volume dei dati

Ho ipotizzato che il sistema mantenga i dati delle schede di allenamento per 3 anni.
In questo lasso di tempo avremo un'aspettativa di circa 1500 clienti e 50 personal trainer per seguire questi clienti.
Suppongo che venga assegnata una scheda di allenamento al mese per ogni cliente e che questi abbia una media di 3 allenamenti settimanali.
Per la relazioni "Composizione" suppongo una media di 5 esercizi per scheda di allenamento.

Concetto nello schema	Tipo[1]	Volume atteso
Cliente	E	1.500
Personal Trainer	E	50
Persona	E	1.550
Scheda di allenamento	E	54.000
Esercizio	E	140
Macchinario	E	30
Associato	R	1.500
Gestione	R	54.000
Ha	R	54.000
Sessione	R	729.000
Composizione	R	270.000
E-M	R	90

Tavola delle operazioni

Cod.	Descrizione	Frequenza attesa
CL1	Iniziare Sessione	4.500/settimana - 3/settimana a cliente
CL2	Visualizzare esercizi da svolgere di un determinato cliente per la scheda a lui assegnata	15.000/settimana - 10/settimana a cliente
CL3	Visualizzare tutte le schede di allenamento associato a un cliente	4.500/mese - 3/settimana a cliente
PT1	Creare scheda di allenamento per un cliente	1.500/mese - 30/mese per Personal Trainer
PT2	Vedi lista esercizi assegnabili	1.500/mese - 30/mese per Personal Trainer
PT3	Vedi clienti affiliati	1.500/mese - 30/mese per Personal Trainer
PT4	Esegui Report	50/settimana - 1/settimana per Personal Trainer
AM1	Inserisci esercizio	5/mese
AM2	Inserisci macchinario	5/mese
AM3	Rimuovi esercizio	5/mese
AM4	Rimuovi macchinario	5/mese

[1] Indicare con E le entità, con R le relazioni

Costo delle operazioni

- CL1 - inizia sessione

L/S - Entità/Relazione: numero di dati acceduti	*Costo*
L - Scheda di allenamento: 36	Totale: 38
S - Sessione: 1	Accessi/settimana: 171.000

- CL2 – visualizzare esercizi da svolgere

L/S - Entità/Relazione: numero di dati acceduti	*Costo*
L - Scheda di allenamento: 36	Totale: 36
	Accessi/settimana: 540.000

- CL3 – visualizzare tutte le schede di allenamento

L/S - Entità/Relazione: numero di dati acceduti	*Costo*
L - Scheda di allenamento: 36	Totale: 36
	Accessi/mese: 162.000

- PT1 – creare scheda di allenamento

L/S - Entità/Relazione: numero di dati acceduti	*Costo*
L - Personal Trainer: 1	Totale: 185
L - Associato: 1	Accessi/mese: 277.500
L - Cliente: 1	
L - Scheda di allenamento: 36	
L - Esercizi: 140	
S - Scheda di allenamento: 2	
S - Gestione: 1	

- PT2 - vedi lista esercizi assegnabili

L/S - Entità/Relazione: numero di dati acceduti	*Costo*
L - Esercizi: 140	Totale: 140
	Accessi/mese: 210.000

- PT3 - vedi clienti affiliati

L/S - Entità/Relazione: numero di dati acceduti	*Costo*
L - Cliente: 36	Totale: 36
	Accessi/mese: 54.000

- PT4 – Report

L/S - Entità/Relazione: numero di dati acceduti	*Costo*
L - Cliente: 30	Totale: 15.690

L - Sessione: 14.580	Accessi/settimana: 784.500
L - Scheda di allenamento: 1.080	

- AM1 - Inserisci esercizio

L/S - Entità/Relazione: numero di dati acceduti	*Costo*
S - Esercizio: 1	Totale: 2
	Accessi/mese: 10

- AM2 - Inserisci macchinario

L/S - Entità/Relazione: numero di dati acceduti	*Costo*
S - Macchinario: 1	Totale: 2
	Accessi/mese: 10

- AM3 - Rimuovi esercizio

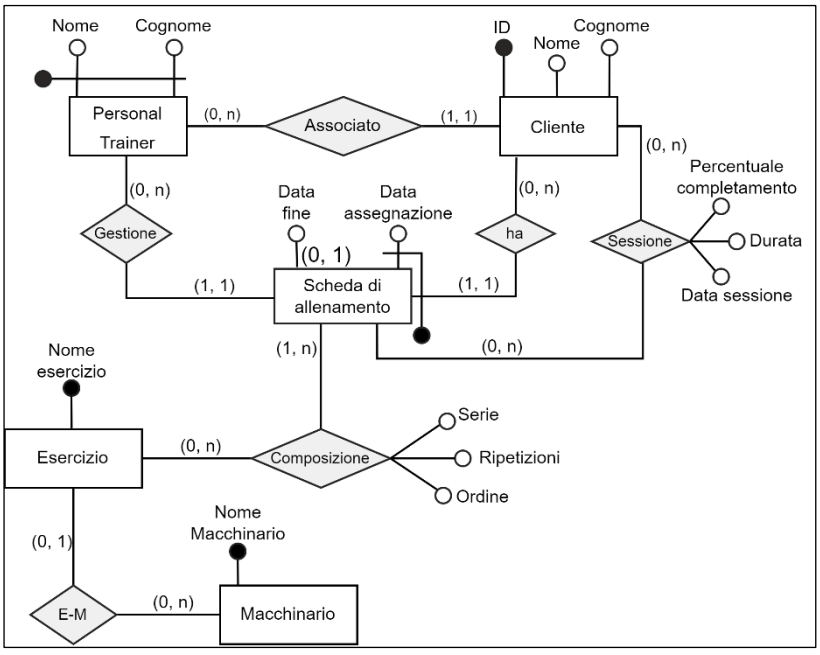
L/S - Entità/Relazione: numero di dati acceduti	*Costo*
S - Esercizio: 1	Totale: 2
	Accessi/mese: 10

- AM4 - Rimuovi macchinario

L/S - Entità/Relazione: numero di dati acceduti	*Costo*
S - Esercizio: 1	Totale: 2
	Accessi/mese: 10

Ristrutturazione dello schema E-R

Come prima cosa vado ad eliminare l'unica generalizzazione collassandola verso il basso:



Andando ad analizzare i costi delle singole operazioni andiamo a verificare come poterne abbattere alcuni di questi per migliorare le prestazioni complessive del sistema.

Partendo alla prima:

- CL1 avrebbe un costo di 36 che però è riducibile ancora di più introducendo un indice sulla data di fine della scheda, così da trovare subito l'ultima scheda a lui assegnata sulla quale si possa iniziare una sessione di allenamento. Costo totale = 3, numero di accessi settimanali = 13.500.
- CL2 con lo stesso indice di CL1 passa a un costo di accesso totale pari a 1 e un numero di accessi settimanali pari a 15.000.
- PT1 sfruttando l'indice sulle schede arriviamo ad avere un costo totale di 150 e un numero di accessi mensili di 225.000.

Trasformazione di attributi e identificatori

Decido di identificare sia personal trainer che cliente tramite un id per semplificare il numero di dati utilizzato per identificarli.
Analizzando il diagramma E-R noto una ridondanza data da Personal Trainer, Scheda di allenamento, Cliente. La relazione Gestione sarebbe ricavabile navigando la relazione Associato e successivamente la relazione Ha, tuttavia questo comporterebbe un aumento dei costi di accesso di alcune operazioni, per cui si è scelto di lasciare questa.

Traduzione di entità e associazioni

- Schema:

PersonalTrainer(Nome, Cognome)
Cliente(Nome, Cognome, NomePT, CognomePT)
SchedaDiAllenamento(DataFine*, DataAssegnazione, NomeC, CognomeC, NomePT, CognomePT)
Sessione(PercentualeCompletamento, Durata, DataSessione, DataAssegnazione, Nome, Cognome)
Composizione(Serie, Ripetizioni, OrdineNellaScheda, NomeEsercizio, DataAssegnazione, Nome, Cognome)
Esercizio(NomeEsercizio, NomeMacchinario)
Macchinario(NomeMacchinario)

- Vincoli:

Cliente(NomePT, CognomePT) ⊆ *PersonalTrainer*(Nome, Cognome)
SchedaDiAllenamento(NomeC, CognomeC) ⊆ *Cliente*(Nome, Cognome)
SchedaDiAllenamento(NomePT, CognomePT) ⊆ *PersonalTrainer*(Nome, Cognome)
Sessione(DataAssegnazione) ⊆ *SchedaDiAllenamento*(DataAssegnazione)
Sessione(Nome, Cognome) ⊆ *Cliente*(Nome, Cognome)
Composizione(DataAssegnazione, Nome, Cognome) ⊆ *SchedaDiAllenamento*(DataAssegnazione, NomeC, CognomeC)
Composizione(NomeEsercizio) ⊆ *Esercizio*(NomeEsercizio)
Esercizio(NomeMacchinario) ⊆ *Macchinario*(NomeMacchinario)

Normalizzazione del modello relazionale

Non è normalizzato poiché mantengo una ridondanza nella scheda di allenamento. Infatti potrei ottenere la scheda di allenamento a cui è associato un cliente passando tramite la relazione tra personal trainer e cliente.

5. Progettazione fisica

Utenti e privilegi

Il sistema prevede tre ruoli distinti:

- Cliente:
  - Responsabile delle operazioni CL1, CL2, CL3
- Personal Trainer:
  - Responsabile delle operazioni PT1, PT2, PT3, PT4
- Amministratore:
  - Responsabile delle operazioni AM1, AM2, AM3, AM4

Strutture di memorizzazione

Tabella Personal Trainer		
Colonna	**Tipo di dato**	**Attributi[2]**
Id	INT	PK, UN, AI
Nome	VARCHAR(20)	NN
Cognome	VARCHAR(20)	NN

Tabella Cliente		
Colonna	**Tipo di dato**	**Attributi[2]**
Id	INT	PK, UN, AI
Nome	VARCHAR(20)	NN
Cognome	VARCHAR(20)	NN
PersonalTrainer_ID	INT	NN

Tabella Scheda Di Allenamento		
Colonna	**Tipo di dato**	**Attributi[2]**
DataAssegnazione	DATE	PK
DataFine	DATE	
PersonalTrainer_ID	INT	
Cliente_ID	INT	PK

Tabella Macchinario		
Colonna	**Tipo di dato**	**Attributi[2]**
NomeMacchinario	VARCHAR(45)	PK

Tabella Storico_Macchinario		
Colonna	**Tipo di dato**	**Attributi[2]**
NomeMacchinario	VARCHAR(45)	PK

[2] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella Esercizio		
Colonna	**Tipo di dato**	**Attributi[2]**
NomeEsercizio	VARCHAR(45)	PK
Macchinario_NomeMacchinario	VARCHAR(45)	

Tabella Storico_Esercizio		
Colonna	**Tipo di dato**	**Attributi[2]**
NomeEsercizio	VARCHAR(45)	PK
Macchinario_NomeMacchinario	VARCHAR(45)	

Tabella Composizione		
Colonna	**Tipo di dato**	**Attributi[2]**
Serie	INT	NN, UN
Ripetizioni	INT	NN, UN
OrdineNellaScheda	INT	NN, UN
NomeEsercizio	VARCHAR(45)	PK
DataAssegnazione	DATE	PK
Cliente_ID	INT	PK

Tabella Sessione		
Colonna	**Tipo di dato**	**Attributi[2]**
PercentualeCompletamento	FLOAT	NN, UN
Durata	TIME	NN
Data Sessione	DATE	PK
Data Assegnazione	DATE	NN
Cliente_ID	INT	PK

Indici

Tabella Personal Trainer	
Indice personaltrainer_nome_cognome	**Tipo[3]: UQ**
Colonna 1	Nome
Colonna 2	Cognome

Tabella Cliente	
Indice cliente_nome_cognome	**Tipo[3]: UQ**
Colonna 1	Nome
Colonna 2	Cognome

Tabella Cliente	
Indice fk_cliente_personaltrainer1_idx	**Tipo[3]: IDX**
Colonna 1	PersonalTrainer_ID

[3] IDX = index, UQ = unique, FT = full text, PR = primary.

Tabella SchedaDiAllenamento	
Indice fk_schedadiallenamento_personaltrainer1_idx	**Tipo[3]: IDX**
Colonna 1	PersonalTrainer_ID

Tabella SchedaDiAllenamento	
Indice fk_schedadiallenamento_cliente1_idx	**Tipo[3]: IDX**
Colonna 1	Cliente_ID

Tabella SchedaDiAllenamento	
Indice byfinescheda	**Tipo[3]: IDX**
Colonna 1	DataFine

Tabella Sessione	
Indice fk_sessione_schedadiallenamento1_idx	**Tipo[3]: IDX**
Colonna 1	DataAssegnazione
Colonna 2	Cliente_ID

Tabella Esercizio	
Indice fk_esercizio_macchinario1_idx	**Tipo[3]: IDX**
Colonna 1	Macchinario_NomeMacchinario

Tabella Storico_Esercizio	
Indice fk_esercizio_macchinario1_idx	**Tipo[3]:**
Colonna 1	Macchinario_NomeMacchinario

Tabella Composizione	
Indice fk_composizione_esercizio1	**Tipo[3]: IDX**
Colonna 1	NomeEsercizio

Tabella Composizione	
Indice fk_composizione_schedadiallenamento1	**Tipo[3]: IDX**
Colonna 1	DataAssegnazione
Colonna 2	Cliente_ID

Trigger

Tabella SchedaDiAllenamento: BEFORE_INSERT

```
CREATE DEFINER = CURRENT_USER TRIGGER `palestraBDprj`.`schedadiallenamento_BEFORE_INSERT`
BEFORE INSERT ON `schedadiallenamento` FOR EACH ROW

BEGIN
	-- Controllo se effettivamente il personal trainer è associato a quel cliente in quel dato momento
	if new.cliente_id not in (select cliente.id from cliente
                                  where cliente.personaltrainer_id = new.personaltrainer_id)
	then
		signal sqlstate '45001' set message_text = 'Non si ha affiliato questo cliente';
	end if;
END
```

Tabella Sessione: BEFORE INSERT

```
CREATE DEFINER = CURRENT_USER TRIGGER `palestraBDprj`.`sessione_BEFORE_INSERT` BEFORE INSERT
ON `sessione` FOR EACH ROW

BEGIN
	declare dataFine date;

	set dataFine := (select datafine
			from schedadiallenamento as sda
			where sda.cliente_id = new.cliente_id and
				sda.dataassegnazione = new.dataassegnazione);

	if dataFine is not null and dataFine != curdate() then
		signal sqlstate '45001' set message_text = 'Scheda di allenamento allenamento non valida';
	end if;

	if new.datasessione < new.dataassegnazione then
		signal sqlstate '45001'
		set message_text = 'Data sessione non valida per la scheda di allenemento voluta';
	end if;
END
```

Tabella Composizione: BEFORE INSERT

```
CREATE DEFINER = CURRENT_USER TRIGGER `palestraBDprj`.`composizione_BEFORE_INSERT` BEFORE
INSERT ON `composizione` FOR EACH ROW

BEGIN
	if (select datafine from schedadiallenamento
	   where new.cliente_id = cliente_id and new.dataassegnazione = dataassegnazione)
	is not null then
		signal sqlstate '45001' set message_text = 'Scheda di allenamento vecchia';
	end if;
END
```

Tabella Esercizio: BEFORE INSERT

```
CREATE DEFINER = CURRENT_USER TRIGGER `palestraBDprj`.`esercizio_BEFORE_INSERT` BEFORE INSERT
ON `esercizio` FOR EACH ROW

BEGIN
	if(new.nomeesercizio not in (select nomeesercizio from storico_esercizio)) then
		insert into storico_esercizio
		value (new.nomeesercizio, new.macchinario_nomemacchinario);
	end if;
END
```

Tabella Macchinario: BEFORE INSERT

```
CREATE DEFINER = CURRENT_USER TRIGGER `palestraBDprj`.`macchinario_BEFORE_INSERT` BEFORE
INSERT ON `macchinario` FOR EACH ROW

BEGIN
	if(new.nomemacchinario not in (select nomemacchinario from storico_macchinario)) then
		insert into storico_macchinario value (new.nomemacchinario);
	end if;
END
```

Eventi

Parlando con il cliente è stato deciso di mantenere all'interno del DB le schede fino a tre anni dalla loro assegnazione. Per andarle a riumovere è stato creato un evento settinamale che andrà periodicamente a compiere questa azione.

Evento per la cancellazione schede di allenamento ogni 3 anni:

```
set global event_scheduler = on;

create event if not exists cleanup
	on schedule
		every 7 day
			on completion preserve
		comment 'Rimuove schede di allenamento più vecchie di 3 anni'
	do
		delete from schedadiallenamento where datediff(curdate(), dataassegnazione) >
1092;
```

Viste

Nessuna vista utilizzata.

Stored Procedures e transazioni

Procedura per poter inserire una scheda di allenamento in schedadiallenamento. Si è utilizzata una transizione poiché sulla tabella è presente un trigger before insert che viene scatenato.

```
CREATE PROCEDURE `creaNuovaSchedaDiAllenamento` (in personalTrainerID int, in clienteID int)
BEGIN
	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;

	set transaction isolation level repeatable read;
	set transaction read write;

	start transaction;
		-- Aggiorno l'ultima scheda di allenamento del cliente
		update schedadiallenamento
		set datafine = curdate() where cliente_id = clienteID and datafine is null;

/* Trigger before insert per controllare se il personal trainer effettivamente possa assegnargli la scheda di
allenamento */

		 -- Inserisco il riferimento alla nuova scheda di allenamento
		insert into schedadiallenamento (dataassegnazione, personaltrainer_id, cliente_id)
		value (curdate(), personalTrainerID, clienteID);
		commit;
END
```

Procedura per inserire un nuovo esercizio tra quelli assegnabili dai Personal Trainer. Si usa una transazione poiché viene scatenato un trigger.

```
CREATE PROCEDURE `inserisciEsercizio` (in nomeEsercizio varchar(45), in nomeMacchinario varchar(45))
BEGIN
	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;

	set transaction isolation level serializable;
	set transaction read write;

	start transaction;
		/* Trigger per inserire l'esercizio anche nella tabella storico_esercizio */

		-- Inserisco l'esercizio nella tabella degli esercizi assegnabili
		insert into esercizio value (nomeEsercizio, nomeMacchinario);
	commit;
END
```

Procedura per assegnare a una scheda il suo esercizio. Inserisce in composizione. Transazione poiché viene scatenato un trigger.

```
CREATE PROCEDURE `inserisciEsercizioInScheda` (in personalTrainerID int, in clienteID int, in dataassegnazione date, in nomeEsercizio varchar(45), in serie int, in ripetizioni int, in ordineNellaScheda int)

BEGIN
	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;

	set transaction isolation level read committed;
	set transaction read write;

	start transaction;
		-- Controllo che effivamente il personal trainer abbia assegnato il cliente
		if personalTrainerID not in (select personaltrainer_id from cliente where id = clienteID) then
			signal sqlstate '45001' set message_text = 'Non si ha affiliato questo cliente';
		end if;

		/* Trigger per controllare se la scheda di allenamento sia valida */

		 -- Inserisco l'esercizio da assegnare alla scheda
		insert into composizione
		value (serie, ripetizioni, ordineNellaScheda, nomeEsercizio, dataassegnazione, clienteID);
	commit;
END
```

Inserisce un nuovo macchinario nella palestra.

```
CREATE PROCEDURE `inserisciMacchinario` (in nomeMacchinario varchar(45))
BEGIN
	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;

	set transaction read write;

	start transaction;
		/* Trigger per inserire il macchinario anche nella tabella storico_macchinario */

		-- Inserisco il macchinario nella tabella dei macchinari utilizzabili
		insert into macchinario value (nomeMacchinario);
	commit;
END
```

Inserisce nella tabella Sessione un nuova sessione svolta da un utente. La transazione è necessaria poiché questa operazione scatena un trigger.

```
CREATE PROCEDURE `inserisciSessione` (in clienteID int, in dataAssegnazione date, in
percentualeCompletamento float, in durata time, in dataEsecuzione datetime)
BEGIN
	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;

	set transaction isolation level serializable;
	set transaction read write;

	start transaction;
		/* Trigger per controllare che il cliente possa inserire una sessione di allenamento per la
scheda di allenamento specificata */

		-- Inserisco la sessione di allenamento nella tabella
		insert into sessione
		value (percentualeCompletamento, durata, dataEsecuzione, dataAssegnazione, clienteID);
	commit;
END
```

Procedura per poter effettuare il login.

```
CREATE PROCEDURE `login` (in var_nome varchar(20), in var_cognome varchar(20), in var_password
varchar(32), out var_ruolo int, out var_id int)
BEGIN
	declare var_user_role ENUM('amministratore', 'cliente', 'personalTrainer');

	select ruolo from utenti
	where nome = var_nome and cognome = var_cognome and password = md5(var_password)
	into var_user_role;

	-- Faccio combaciare con l'enum del client
	if var_user_role = 'amministratore' then
		set var_ruolo = 1;
		set var_id = 0;
	elseif var_user_role = 'cliente' then
		set var_ruolo = 2;
		set var_id = (select id from cliente where nome = var_nome and cognome = var_cognome);
	elseif var_user_role = 'personalTrainer' then
		set var_ruolo = 3;
		set var_id = (select id from personaltrainer
				where nome = var_nome and cognome = var_cognome);
	else
		set var_ruolo = 4;
		set var_id = -1;
	end if;
END
```

Ho scelto di rendere questa procedura "serializable" poiché mentre sto modificando questa tabella vorrei che nessun personal trainer possa anche solo leggerla così da evitare l'assegnazione di esercizi sbagliati.

```
CREATE PROCEDURE `rimuoviEsercizio` (in var_nomeEsercizio varchar(45))
BEGIN
	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;

	set transaction isolation level serializable;
	set transaction read write;

	start transaction;
		delete from esercizio where (nomeesercizio = var_nomeEsercizio);
	commit;
END
```

Procedura per rimuovere un macchinario. Segnata come transazione visto che è una foreign key della tabella esercizio ed è stata implementata come cascade.

```
CREATE PROCEDURE `rimuoviMacchinario` (in var_nomeMacchinario varchar(45))
BEGIN
	declare exit handler for sqlexception
	begin
		rollback;
		resignal;
	end;

	set transaction isolation level serializable;
	set transaction read write;

	start transaction;
		delete from macchinario where nomemacchinario = var_nomeMacchinario;
	commit;
END
```

Procedura per vedere i clienti affiliati a un personal trainer

```
CREATE PROCEDURE `vediClientiAffiliati` (in var_personalTrainerID int)
BEGIN
	select id, nome, cognome from cliente where personaltrainer_id = var_personalTrainerID;
END
```

Procedura che restituisce la lista degli esercizi assegnabili da un personal trainer a un cliente.

```
CREATE PROCEDURE `vediListaEserciziDisponibili` ()
BEGIN
	select nomeesercizio from esercizio;
END
```

Procedura per ottenere la lista dei macchinari disponibili.

```
CREATE PROCEDURE `vediListaMacchinariDisponibili` ()
BEGIN
	select * from macchinario;
END
```

Procedura per vedere gli esercizi della scheda di allenamento assegnata (con datafine = null)

```
CREATE PROCEDURE `vediSchedaDiAllenamento` (in clienteID int)
BEGIN
	select dataassegnazione, ordinenellascheda as ordine, nomeesercizio, serie, ripetizioni
	 from composizione
	where cliente_id = clienteID
		and dataassegnazione = (select dataassegnazione from schedadiallenamento
					where cliente_id = clienteID and datafine is null)
	order by ordinenellascheda;
END
```

Procedura per vedere lo storico delle schede assegnate a un cliente. Il cursore serve a distinguere le varie schede e per ogni scheda inviare i relativi esercizi

```
CREATE PROCEDURE `vediTutteLeSchedeDiAllenamento` (in clienteID int)
BEGIN
	declare done int default false;
	declare var_scheda date;

	declare cur cursor for
		select dataassegnazione from schedadiallenamento where cliente_id = clienteID;

	declare continue handler for not found set done = true;

	if (select id from cliente where id = clienteID) is null then
		signal sqlstate '45001' set message_text = 'ID cliente non valido';
	end if;

	open cur;
	read_loop: loop
		fetch cur into var_scheda;

		if done then leave read_loop;
		end if;

		select dataassegnazione, ordinenellascheda, nomeesercizio, serie, ripetizioni
		from composizione where cliente_id = clienteID and dataassegnazione = var_scheda
		order by ordinenellascheda;
	end loop;
	close cur;
END
```

Creo un cursore per poter scorrere indice per indice i clienti affiliati al personal trainer identificato da personaTrainerID. Uso un if per definire il livello di isolamento della transazione diverso a seconda della datainizio. Per ogni cliente invio prima il numero di allenamenti svolti e successivamente tutte le sessioni da lui sostenute nell'intervallo di tempo richiesto.

```
CREATE PROCEDURE `vediReport` (in personalTrainerID int, in datainizio date, in datafine date)
BEGIN
	declare done int default false;
	declare var_clienteID int;

	declare cur cursor for
		select id from cliente where personaltrainer_id = personalTrainerID;

	declare continue handler for not found set done = true;

	-- Nel caso la data inserita per il report fosse avanti rispetto a oggi
	if datafine > curdate() then
		set datafine = curdate();
	end if;

	if datafine = curdate() then set transaction isolation level repeatable read;
	end if;
	if datafine < curdate() then set transaction isolation level read uncommitted;
	end if;
	set transaction read only;

	start transaction;
		open cur;
		read_loop: loop
			fetch cur into var_clienteID;

			if done then leave read_loop;
			end if;

			select nome, cognome, cliente_id, count(*) as numeroallenamenti
			from cliente
			join sessione on (cliente.ID = sessione.cliente_id and cliente.ID = var_clienteID)
			where personaltrainer_id = personalTrainerID
				and (datainizio <= datasessione and datasessione <= datafine)
				and cliente_id = var_clienteID
			group by cliente_id;

			select cliente_id, nome, cognome, dataassegnazione as dataassegnazionescheda,
datasessione, durata, percentualecompletamento
			from cliente
			join sessione on cliente.ID = sessione.cliente_id
			where personaltrainer_id = personalTrainerID
				and (datainizio <= datasessione and datasessione <= datafine)
				and cliente_id = var_clienteID;
		end loop;
		close cur;
	commit;
END;
```