

Elaborato per il corso di Basi di dati
A.A 2022/2023
Progetto di una base di dati campaign di *Dungeons & Dragons*

Michele Ciavatti
michele.ciavatti2@studio.unibo.it
Mat. 0001088478

Sommario

Sommario	1
1 Analisi dei requisiti	2
1.1 Intervista	2
1.2 Estrazione dei concetti principali	3
2 Progettazione concettuale	5
2.1 Schema scheletro	5
2.2 Schema finale	9
3 Progettazione logica	11
3.1 Stima del volume dei dati	11
3.2 Descrizione delle principali operazioni e relativa frequenza	11
3.3 Schemi di navigazione e tabelle degli accessi	12
3.4 Raffinamento dello schema	17
3.5 Analisi delle ridondanze	20
3.6 Traduzione di entità e associazioni in relazioni	22
3.7 Schema relazionale finale	24
3.8 Traduzione delle operazioni in query SQL	26
4 Progettazione dell'applicazione	27
4.1 Descrizione dell'architettura dell'applicazione realizzata	27

1 Analisi dei requisiti

Si vuole realizzare un database a supporto dell'automatizzazione della gestione di una o più campagne di *Dungeons & Dragons*. Pertanto, il database dovrà immagazzinare informazioni relative ai personaggi, mostri e NPCs (*Non-Playable Characters*).

1.1 Intervista

Si vuole tenere traccia dei dati di una campagna di *Dungeons & Dragons*. Per ogni personaggio si memorizzi nome (univoco), razza e classe. Ciascun personaggio appartiene ad una classe (e sottoclasse dal livello 3 in poi) ed una razza, ed ogni classe e razza conferisce abilità e condizioni (ovvero abilità passive) specifiche che influenzano le statistiche del personaggio, le quali a loro volta influiscono sul lancio di dadi nell'esecuzione di azioni. Ogni personaggio può essere in possesso di armi, catalizzatori per gli incantesimi e consumabili (questi ultimi si consumano dopo un certo numero di utilizzi, specifico all'oggetto). Sia armi che catalizzatori possiedono un modificatore da applicare al lancio dei dadi quando vengono usati: tuttavia, il numero e tipo di dadi usati per eseguire un'azione con l'arma dipende dall'arma stessa, mentre numero e tipo di dadi usati per lanciare un incantesimo dipende dall'incantesimo (e non dal catalizzatore, che influisce solo con il modificatore). I personaggi possono acquisire risorse comprandole (si memorizzi l'oro per ogni personaggio), ricevendole in dono da NPCs o trovandole (che tendenzialmente significa razziandole dai cadaveri).

Il vero protagonista di una campagna di *D&D* è il party, ovvero il gruppo di personaggi che giocano nella specifica sessione. La campagna, essendo tendenzialmente lunga, si svolge nell'arco di numerose sessioni, e non necessariamente i personaggi che partecipano ad una sessione sono gli stessi di tutte le altre: i personaggi possono morire, oppure il giocatore che controlla un personaggio potrebbe non essere presente ad una partita e via dicendo. Nel caso di personaggi morti, il giocatore proprietario del *char* morto creerà un nuovo personaggio che si aggiungerà automaticamente al party permettendo al giocatore di partecipare alla sessione, mentre in tutti gli altri casi di personaggi assenti il gruppo sarà composto da meno *char* e questo fatto deve essere registrato nella sessione specifica in cui avviene.

Nonostante il fatto che il numero di personaggi possa variare da sessione a sessione, il numero di personaggi massimi per una sessione rimane il numero di giocatori che hanno deciso di cominciare la campagna. Non è quindi concesso l'ingresso di nuovi giocatori alla campagna. Per ogni personaggio (e NPC e mostro) deve essere concessa la possibilità di salvare un'immagine relativa all'estetica del personaggio.

In ogni sessione, potrà accadere che il party interagisca con uno o più NPCs: il database dovrebbe memorizzare gli individui conosciuti dal party e anche il tipo di rapporto che ha mostrato con il party nella sessione fra ostile, benevolo e indifferente. Durante la stessa sessione, il tipo di rapporto può cambiare: in tal caso, si memorizzi solo l'ultimo tipo.

In ogni sessione, il party parteciperà ad un combattimento: *D&D* non pone limiti al numero di incontri che possono essere fatti in una sessione, ma si tratta solitamente di eventi lunghi ore e quindi è difficile farne più di uno per partita. Analogamente si potrebbe fare una sessione senza combattimento, ma una sessione senza un incontro è considerata noiosa e quindi si evita tale pratica. Per ogni combattimento si vuole memorizzare lo storico dei turni, in cui per ogni turno si descrivono le azioni di giocatori e mostri. Particolare enfasi viene posta sui danni fatti da ciascun partecipante al combattimento in ogni turno, in modo da consentire in futuro il calcolo dei danni medi eseguiti da ogni partecipante.

1.2 Estrazione dei concetti principali

Termine	Breve descrizione	Eventuali sinonimi
Campagna	Avventura di gioco	
Sessione	Occasione di gioco per i personaggi della campagna	Partita
Party	Insieme dei personaggi attivi nella sessione	Gruppo
Personaggio	Avatar del giocatore nella partita	Char, partecipante
NPC	Personaggio non giocante, ovvero non controllato da alcun giocatore	Individuo
Mostro	Antagonista dei giocatori in un combattimento	
Incontro	Scontro del party con forze avverse quali mostri o NPCs	Combattimento
Turno	Insieme delle azioni che un combattente esegue consecutivamente in un combattimento	

Figure 1: Tabella dei concetti principali.

La figura 1 mostra i concetti principali che si evincono dall'intervista (sez. 1.1). Dalla lettura e comprensione dei requisiti, si procede redigendo un testo che riassume in maniera chiara e univoca i concetti:

La campagna è basata su un **party** di dimensione massima predefinita, ed è suddivisa in **sessioni**.

Il party è composto da **personaggi**, e per ogni personaggio si memorizza:

- *nome* (univoco).
- *immagine* (opzionale), un file che contiene l'immagine del personaggio

- *livello*
- *oro*
- **oggetti** in possesso (opzionale).
- **razza**
- **classe**, la quale possiede un insieme di **sottoclassi** che fanno capo alla classe considerata.
- **sottoclasse** opzionalmente, in quanto presente solo se il personaggio è di livello superiore a 3.

Ogni **oggetto** è un **catalizzatore**, **arma** o **consumabile**. Tutti gli oggetti hanno un *nome* e una *descrizione* relativa alle conseguenze del loro uso, e un consumabile contiene il *numero di utilizzi* prima di essere sprecato.

Per ogni razza, classe e sottoclasse si memorizza il *nome* e le **abilità** che garantisce, le quali possono essere *attive* o *passive*. Per la **campagna** si memorizza un *nome* e il *numero di giocatori*. La campagna si svolge in **sessioni**, le quali vengono identificate con un *numero progressivo* rispetto alla campagna. In ogni sessione, si memorizza il *party* che ha partecipato alla sessione e i *personaggi morti* (opzionale).

In ogni sessione, il party ha un **combattimento** contro un **gruppo di mostri**. Il combattimento si svolge in **turni**, e per ogni turno si devono memorizzare i *morti* (opzionale) sia per il party che per il gruppo di mostri, i *danni* inflitti dai personaggi del party, e una *descrizione* del turno di ogni partecipante allo scontro.

In ogni sessione, il party può incontrare dei **NPCs**: si memorizzi l'ultimo *tipo di rapporto* con il party nella sessione. Per ogni NPC, si memorizzi il *nome*, un'*immagine* (opzionale) e una breve *descrizione* sulla sua personalità.

Segue un elenco delle principali azioni richieste:

1. Creare una nuova campagna
2. Creare una nuova sessione
3. Creare un party
4. Registrare il turno di un personaggio o mostro
5. Mostrare la media di danni eseguita per uno specifico personaggio
6. Aggiungere interazione con NPC sconosciuto
7. Aggiungere interazione con NPC conosciuto
8. Mostrare NPCs conosciuti nella campagna con relativo tipo di rapporto nei confronti del party
9. Mostrare la composizione del party sulla base di razze, classi e/o sottoclassi.

10. Aggiungere un oggetto nuovo ad un personaggio
11. Mostrare gli oggetti in possesso di un personaggio
12. Verificare se un certo oggetto è in possesso del party

2 Progettazione concettuale

2.1 Schema scheletro

Le entità personaggio e NPC sono specializzazione di una più generica entità. Per semplicità, denominiamo tale sovraclassa con il nome di **Personaggio**, la quale si specializzerà in **Protagonista** per i personaggi controllati da giocatori e **NPC** per gli altri. Per gli NPC sarà sufficiente memorizzare a parte una descrizione della personalità, mentre per i protagonisti entreranno in gioco altre entità che in questo primo schema ignoriamo. I protagonisti si radunano nell'entità **Party** ed attraverso essa partecipano ad una sessione, mentre gli NPC possono partecipare direttamente alle sessioni. La figura 2 mostra anche la suddivisione della campagna in sessioni. La logica dietro agli NPC è che un NPC, se partecipa ad una sessione, interagisce necessariamente con il Party (non si memorizzano NPC che non interagiscono con il Party), e quindi salviamo nell'associazione Comparsa il dato relativo al tipo di rapporto avuto con il Party che il testo richiede.

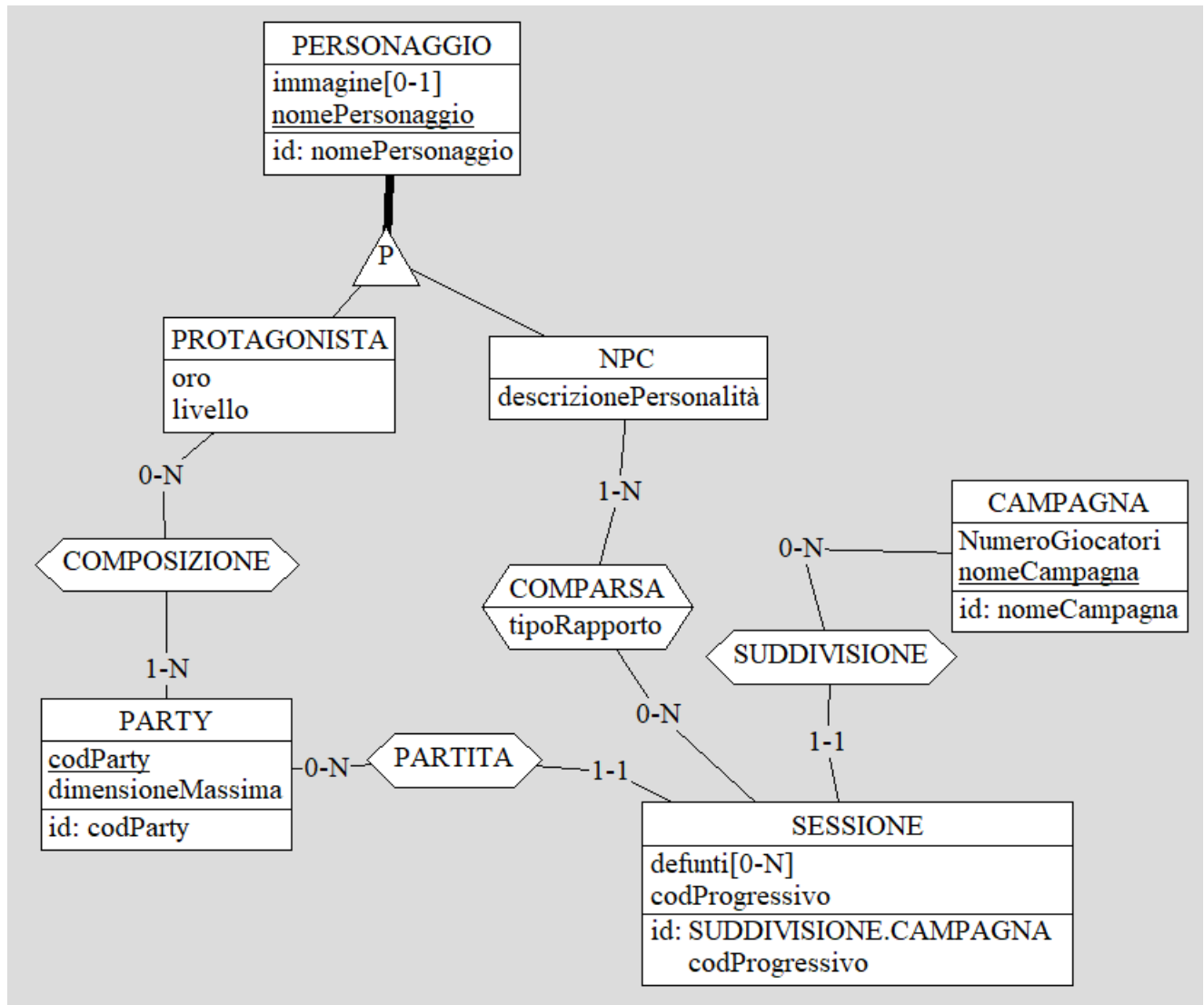


Figure 2: Schema preliminare delle entità dei personaggi legati alle sessioni

Classi, sottoclassi e razze possono essere considerate specializzazioni di una generica sovraentità **Qualità**: ogni qualità offre delle **Abilità**, che si suddividono naturalmente in **Attive** e **Passive**. Ogni **Sottoclasse** fa riferimento ad una classe come da testo. Per collegare tutte le informazioni esposte al protagonista, si è deciso di utilizzare due associazioni binarie: una per la sottoclasse e una per la razza. Notare che la sottoclasse fa sempre capo una classe, quindi specificare una sottoclasse significa anche specificare una classe (ergo non è nec-

essaria un'ulteriore associazione fra protagonista e classe). La figura 3 esplica la logica appena esposta. Limite di questo schema è il fatto che non viene espresso il vincolo per cui un personaggio può avere una sottoclasse solo dal livello 3 in poi: il funzionamento del database prevederà che alla creazione del protagonista il giocatore decida subito la sottoclasse anche se il protagonista ha un livello inferiore al 3, e poi tale sottoclasse sarà effettivamente considerata basandosi sull'attributo *livello* del protagonista in esame.

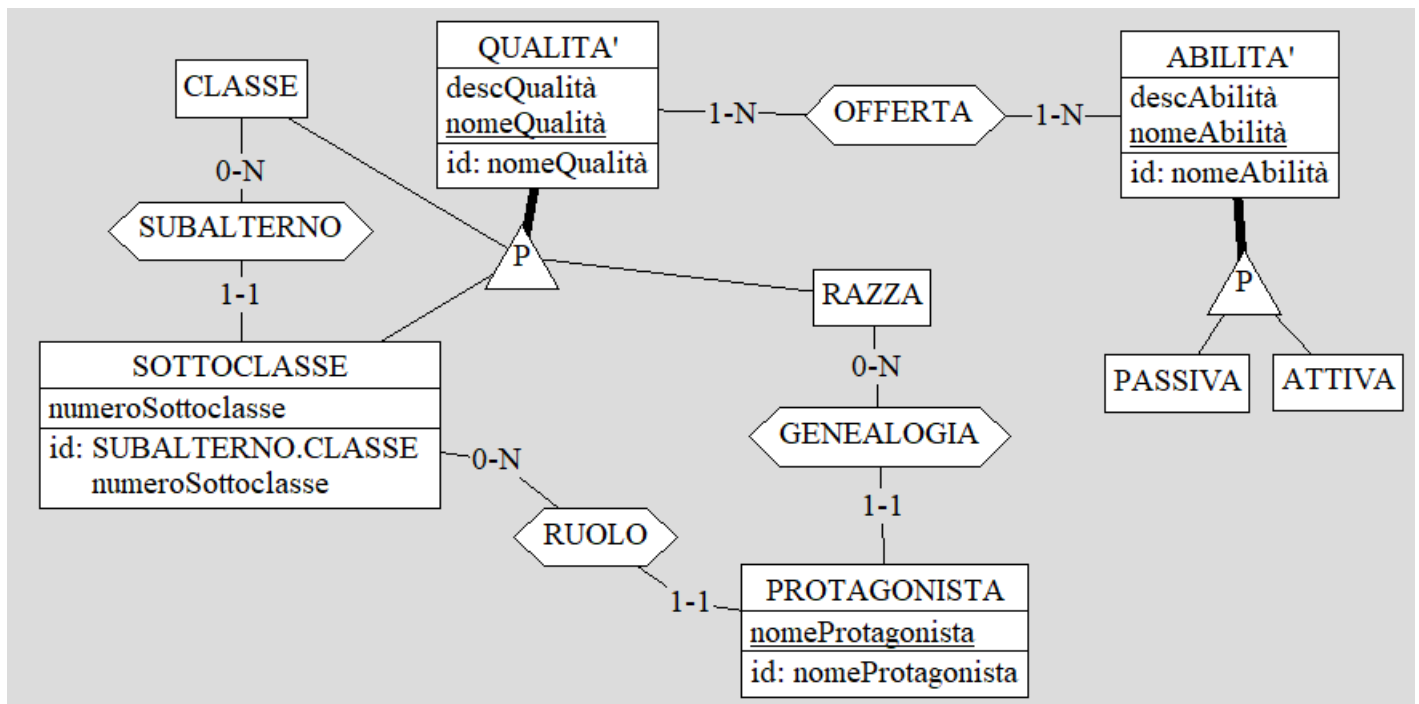


Figure 3: Schema preliminare delle caratteristiche relative ad un protagonista

Il possesso di **oggetti** è schematizzato nella figura 4. Unica nota su tale schema è la scelta di creare una gerarchia parziale: l'intervista afferma che gli oggetti possono essere solo dei 3 tipi riportati, ma per conoscenza più approfondita del dominio sappiamo che possono esistere oggetti che non appartengono a nessuna delle sottocategorie specificate (es. oggetti chiave per progredire nella campagna). Ciò spiega la scelta di una gerarchia parziale.

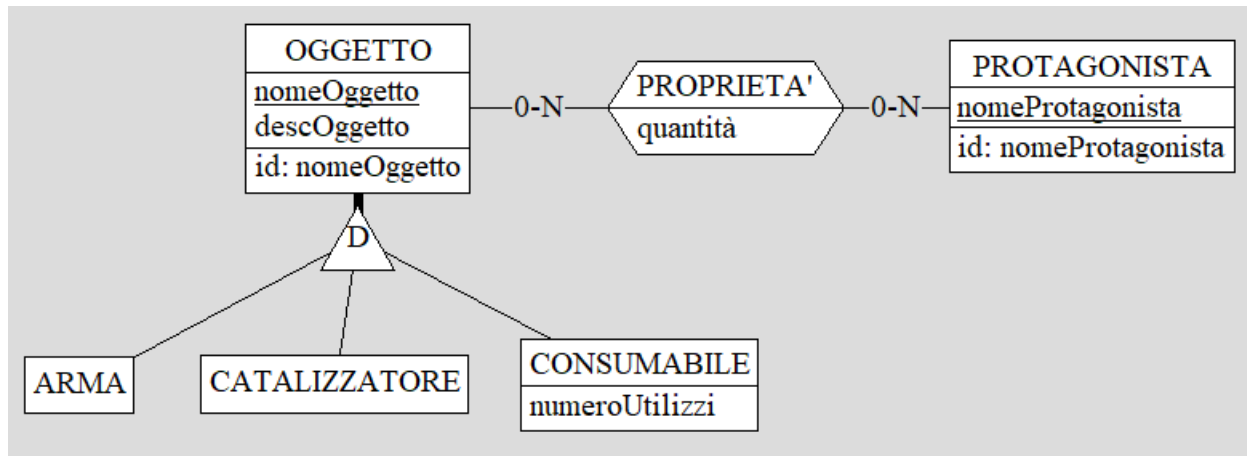


Figure 4: Schema preliminare degli oggetti

In ogni sessione si svolge un solo combattimento: per questo motivo, abbiamo deciso di non creare un'entità dedicata al combattimento ma di creare direttamente l'entità **Turno** che si collega ad una sola sessione. Ad ogni turno partecipa un **mostro** o un **protagonista**: abbiamo deciso che la sovraentità **Personaggio** racchiuda anche i mostri, di modo da associarla ai turni. Con questa schematizzazione, è teoricamente possibile che un NPC prenda parte ad un combattimento, il che è un fatto possibile anche nel dominio. La figura 5 mostra lo schema E/R. Lo schema non esprime il vincolo che un protagonista può partecipare ad un turno solo se parte del party che sta giocando la specifica sessione.

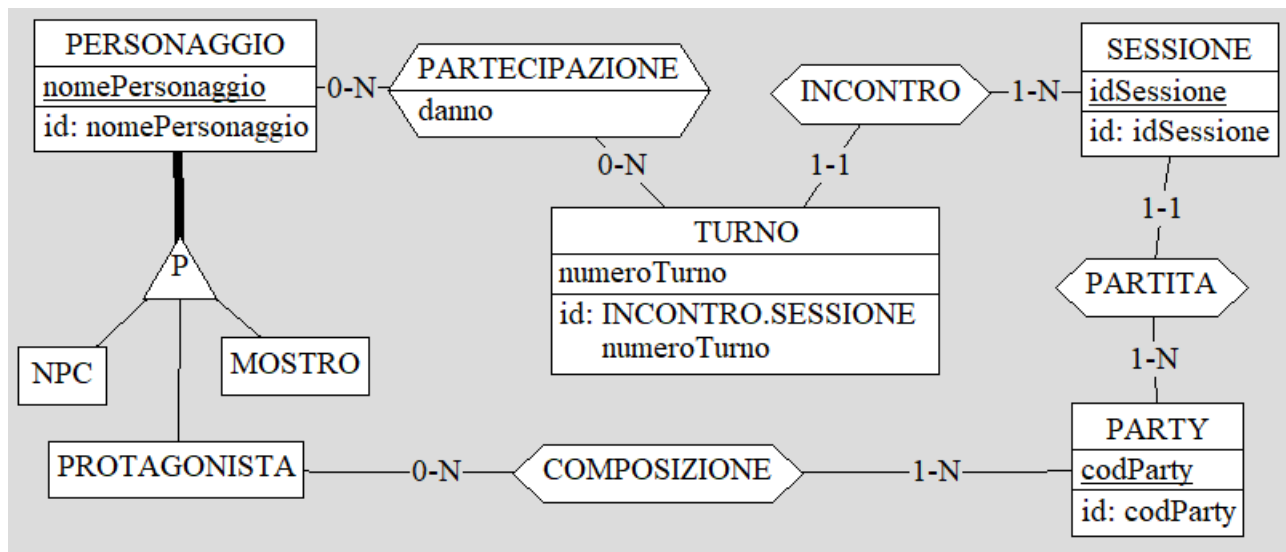
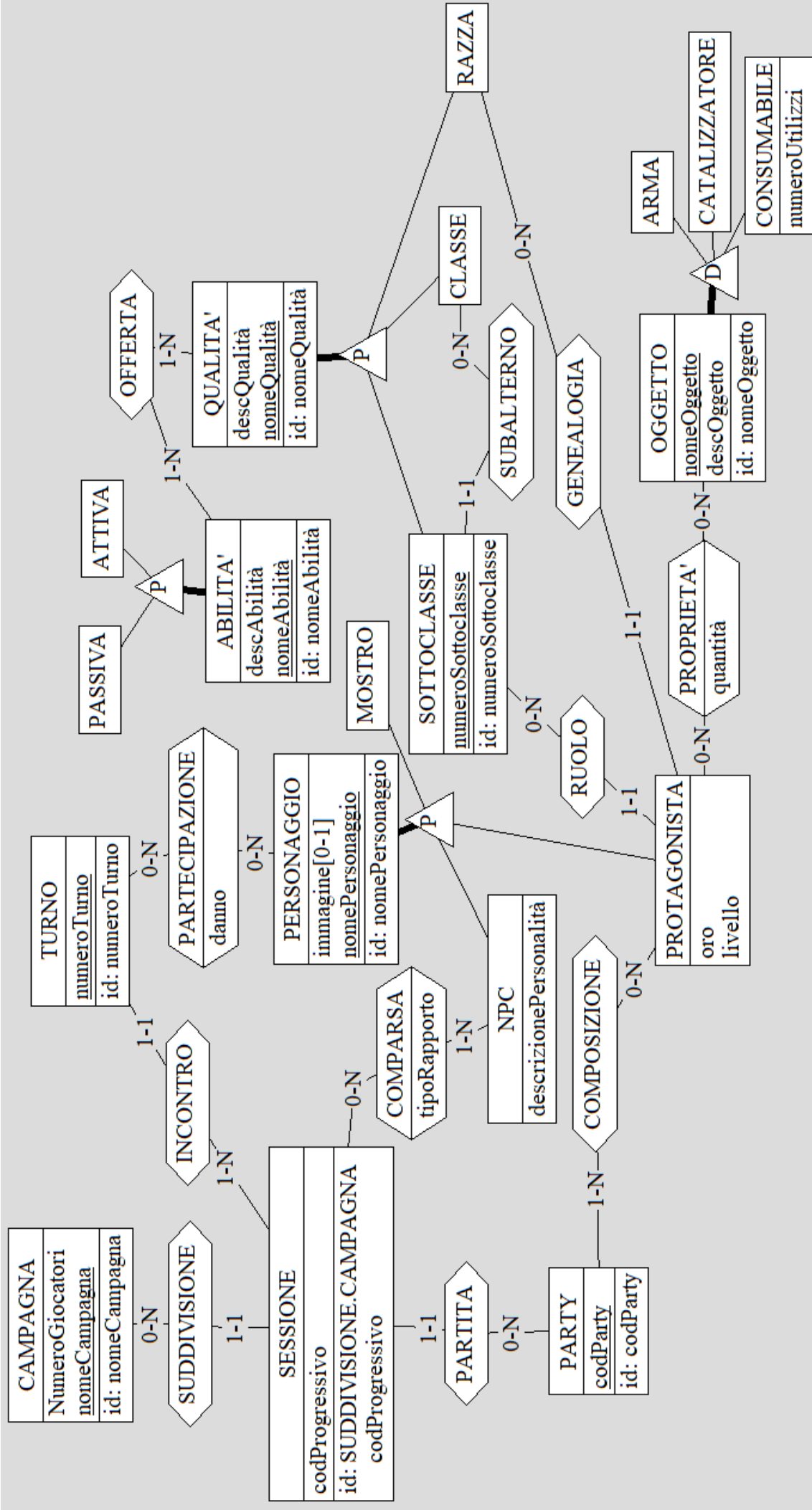


Figure 5: Schema preliminare relativo ai combattimenti

2.2 Schema finale

Figura alla pagina successiva.



3 Progettazione logica

3.1 Stima del volume dei dati

Concetto	Costrutto	Volume	Concetto	Costrutto	Volume
Campagna	E	5	Sottoclasse	E	126
Suddivisione	A	100	Subalterno	A	126
Sessione	E	100	Classe	E	13
Partita	A	100	Qualità	E	181
Party	E	15	Offerta	A	3620
Composizione	A	90	Abilità	E	1250
Protagonista	E	50	Comparsa	A	300
Proprietà	A	500	NPC	E	75
Oggetto	E	482	Incontro	A	500
Genealogia	A	50	Turno	E	500
Razza	E	42	Partecipazione	A	5000
Ruolo	A	50	Personaggio	E	475
			Mostro	E	350

3.2 Descrizione delle principali operazioni e relativa frequenza

Le operazioni sono quelle riportate in fase di analisi.

Codice	Operazione	Frequenza
1	Creare una nuova campagna	1/mese
2	Creare una nuova sessione	1/giorno
3	Creare un nuovo party	2/settimana
4	Registrare il turno di un personaggio o nemico	60/giorno
5	Mostrare la media di danni eseguita per uno specifico personaggio	5/giorno
6	Aggiungere interazione con NPC sconosciuto	1/giorno
7	Aggiungere interazione con NPC conosciuto	2/giorno
8	Mostrare NPCs conosciuti nella campagna con relativi dati	5/giorno
9	Mostrare la composizione del party per razze, classi e/o sottoclassi	25/giorno
10	Aggiungere un oggetto nuovo ad un personaggio	10/giorno
11	Mostrare gli oggetti di un personaggio	5/giorno
12	Verificare se un certo oggetto è in possesso del party	1/giorno

Figure 6: Tabella delle operazioni principali

3.3 Schemi di navigazione e tabelle degli accessi

Di seguito riportiamo le tabelle degli accessi delle operazioni riportate nella tabella 6, e anche relativi schemi di navigazione qualora non banali.

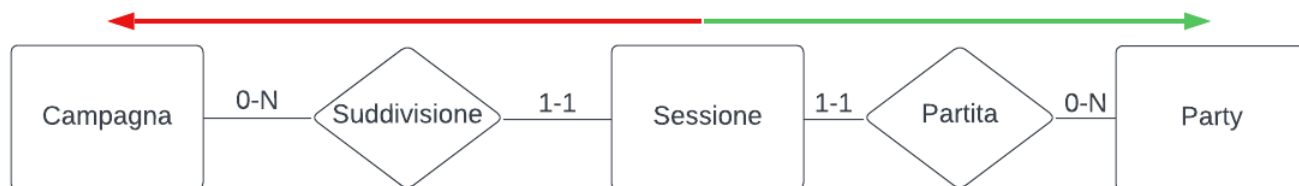
OP 1 - creare una nuova campagna

Concetto	Costrutto	Accessi	Tipo
Campagna	E	1	S
Totale: 1S → 2 al mese			

OP 2 - creare una nuova sessione

La creazione di una sessione implica subito un legame con un party: tale party può essere già presente oppure può dover essere creato. In questa operazione, consideriamo il caso in cui il party è già presente in quanto nell'OP 3 si considera il caso di creazione del party.

Concetto	Costrutto	Accessi	Tipo
Sessione	E	1	S
Suddivisione	A	1	S
Partita	A	1	S
Campagna	E	1	L
Party	E	1	L
Totale: 3S + 2L → 8 al giorno			



OP 3 - creare un nuovo party

Per la creazione di un nuovo party, è necessario creare almeno 1 associazione di *Composizione*. Abbiamo ipotizzato che mediamente un party è composto da 6 individui, e infatti nella tabella di stima del volume dei dati abbiamo riportato 90 istanze di *Composizione*, che è esattamente il numero di party che ci aspettiamo moltiplicato per 6.

Concetto	Costrutto	Accessi	Tipo
Party	E	1	S
Composizione	A	6	S
Protagonista	E	6	L
Totale: 7S + 6L → 40 alla settimana			

OP 4 - registrare il turno di un personaggio o nemico

Concetto	Costrutto	Accessi	Tipo
Partecipazione	A	1	S
Turno	E	1	S
Personaggio	E	1	L
Totale: 1L + 2S → 300 al giorno			

OP 5 - mostrare la media di danni eseguita per uno specifico personaggio

Supponendo che un personaggio abbia partecipato a 6 sessioni mediamente e quindi a 30 turni. Ovviamente, queste supposizioni sono estremamente approssimative perchè è chiaro che il personaggio aumenterà il numero di sessioni a cui partecipa nel tempo (salvo morte o fine della campagna), e quindi il costo di questa operazione cresce man mano che il personaggio partecipa a nuove sessioni.

Concetto	Costrutto	Accessi	Tipo
Personaggio	E	1	L
Partecipazione	A	30	L
Totale: 31L → 155 al giorno			

OP 6 - aggiungere interazione con NPC sconosciuto

Concetto	Costrutto	Accessi	Tipo
Comparsa	A	1	S
NPC	E	1	L
Sessione	E	1	L
Totale: 1S + 2L → 4 al giorno			

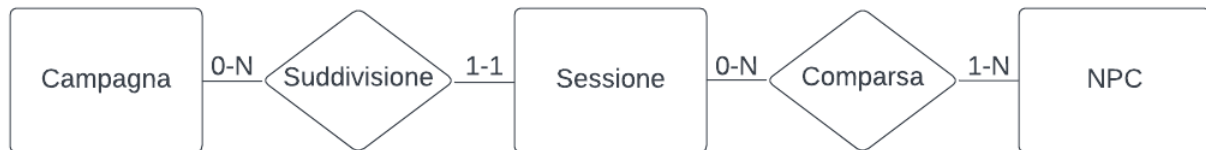
OP 7 - aggiungere interazione con NPC sconosciuto

Concetto	Costrutto	Accessi	Tipo
Comparsa	A	1	S
Totale: 1S → 4 al giorno			

OP 8 - mostrare NPC's conosciuti nella campagna con relativi dati

Supponendo che ogni campagna abbia 20 sessioni, e in ogni sessione ci siano incontri con 3 NPC's mediamente. Tuttavia, l'incontro con un NPC non implica incontrare un NPC nuovo, e solitamente si incontrano NPC già conosciuti, per cui ci dovrà essere un controllo successivo per considerare solo gli NPC (e non gli incontri con questi) con l'ultimo tipo di rapporto che hanno avuto con il party.

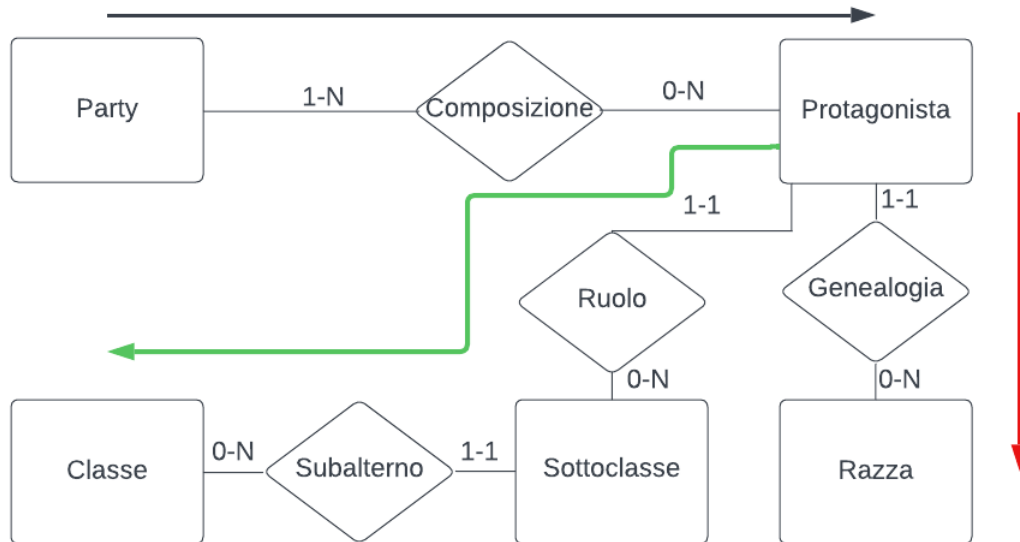
Concetto	Costrutto	Accessi	Tipo
Campagna	E	1	L
Suddivisione	A	20	L
Sessione	E	20	L
Comparsa	A	60	L
NPC	E	60	L
Totale: 161L → 805 al giorno			



OP 9 - mostrare la composizione del party per razze, classi e/o sottoclassi

Un party ha mediamente 6 protagonisti.

Concetto	Costrutto	Accessi	Tipo
Party	E	1	L
Composizione	A	6	L
Protagonista	E	6	L
Ruolo	A	6	L
Sottoclasse	E	6	L
Subalterno	A	6	L
Classe	E	6	L
Genealogia	A	6	L
Razza	E	6	L
Totale: 49L → 1225 al giorno			



OP 10 - aggiungere un oggetto nuovo ad un personaggio

Concetto	Costrutto	Accessi	Tipo
Protagonista	E	1	L
Proprietà	A	1	S
Oggetto	E	1	S
Totale: 1L + 2S → 50 al giorno			

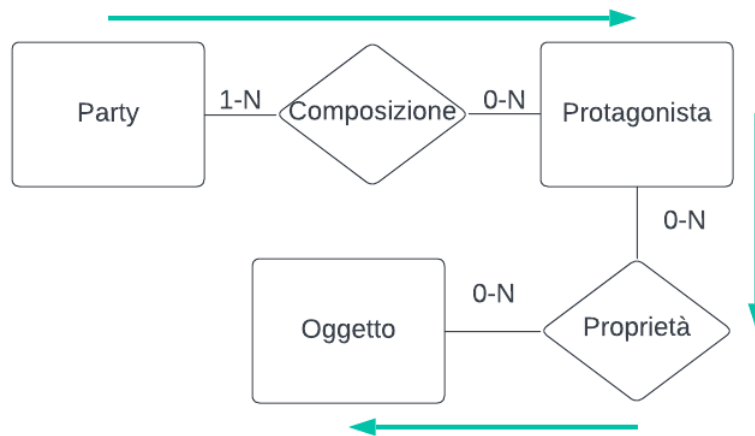
OP 11 - mostrare gli oggetti di un personaggio

Un protagonista ha mediamente 10 oggetti.

Concetto	Costrutto	Accessi	Tipo
Protagonista	E	1	L
Proprietà	A	10	L
Oggetto	E	10	L
Totale: 31L → 155 al giorno			

OP 12 - verificare se un certo oggetto è in possesso del party

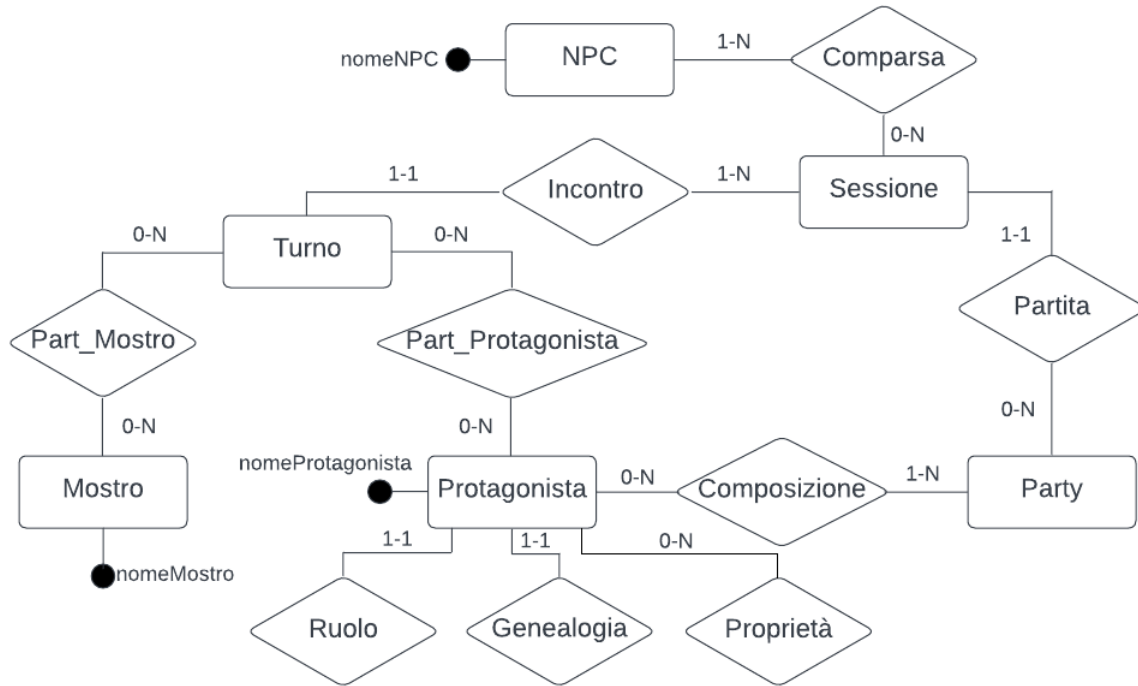
Concetto	Costrutto	Accessi	Tipo
Party	E	1	L
Composizione	A	6	L
Protagonista	E	6	L
Proprietà	A	60	L
Oggetto	E	60	L
Totale: 133L → 133 al giorno			



3.4 Raffinamento dello schema

Eliminazione delle gerarchie Lo schema concettuale contiene 4 gerarchie, che vediamo singolarmente.

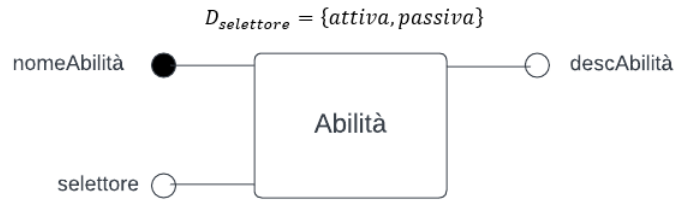
- **Personaggio.** Si è optato per un collasso verso il basso in quanto le operazioni previste tendono a riguardare le entità figlie. Abbiamo operato una modifica: solo *mostri* e *protagonisti* possono partecipare ad un *turno*. La conseguenza è che se un *NPC* dovesse partecipare ad un *turno*, allora dovrebbe essere salvato nuovamente come *mostro*. Questa complicazione è accettabile in quanto l'eventualità che un *NPC* partecipi ad un combattimento è piuttosto rara nel dominio (e infatti non è stata nominata nell'intervista). Nell'immagine, per semplicità non abbiamo riportato attributi salvo gli identificatori dell'entità coinvolte nel collasso.



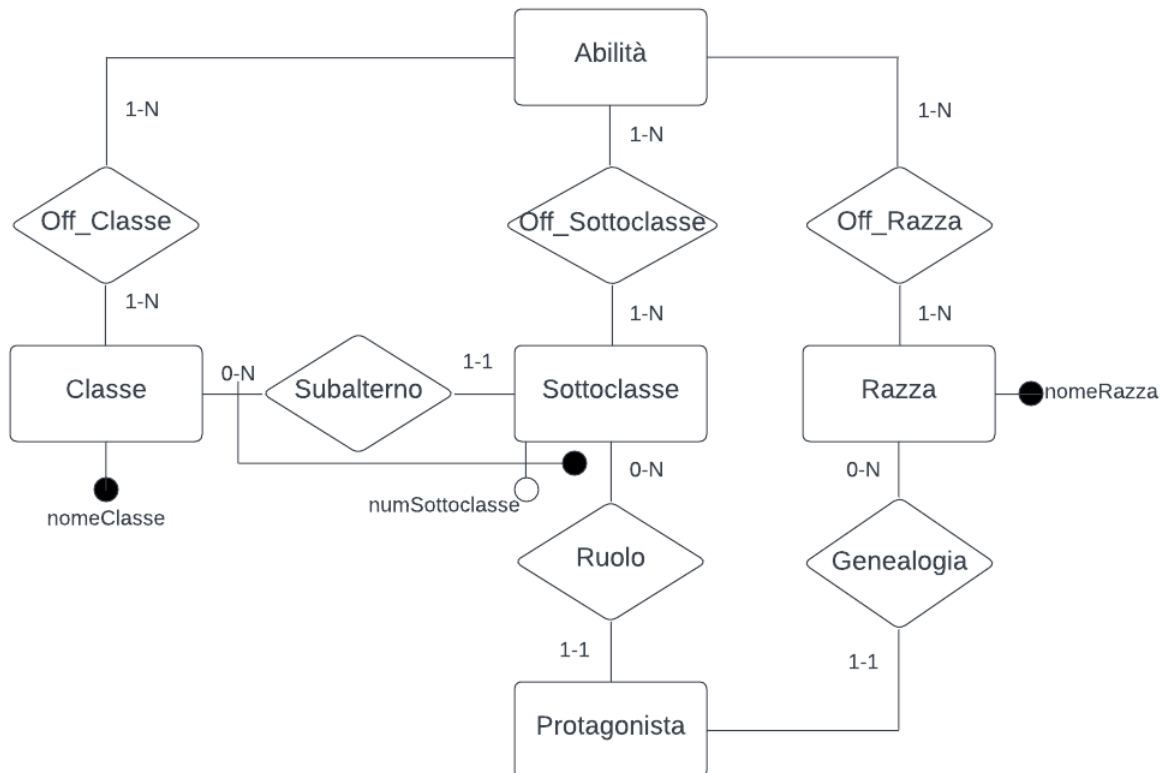
- **Oggetto.** Dato che la copertura non è completa, si esclude la possibilità di fare un collasso verso il basso. Scegliamo l'opzione di fare un collasso verso l'alto, optando per un selettore il cui dominio è $\{arma, catalizzatore, consumabile, altro\}$.



- **Abilità.** Dato che a livello di schema non è contemplata una reale differenza fra abilità attive e passive, optiamo per un collasso verso l'alto con un selettore che permetta di distinguere fra le due categorie.



- **Qualità.** Si è optato per un collasso verso il basso per 2 motivi: in primo luogo, è presente un'associazione fra solo due delle tre entità figlie che dovrebbe essere tradotta con un'associazione ad anello nel caso optassimo per un collasso verso l'alto (perdendo l'informazione riguardo a quali entità figlie sono in associazione), ed in secondo luogo per mantenere chiari i vincoli riguardo alla costruzione del personaggio (che deve avere una *sottoclasse*, una *classe* ed una *razza*).



Scelta delle chiavi primarie

Le chiavi primarie sono già state identificate univocamente nello schema concettuale, eccezion fatta per la chiave di *Sottoclasse*: tale entità ha due identificatori nello schema concettuale riportato alla sezione precedente, ma dopo il raffinamento operato in questa sezione rimane solo una chiave possibile (ovvero il numero progressivo della sottoclasse e classe di riferimento come foreign key).

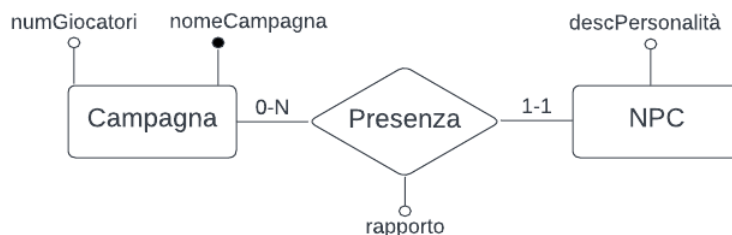
Eliminazione degli identificatori esterni

3.5 Analisi delle ridondanze

In questo paragrafo, valutiamo la possibilità di inserire delle ridondanze al fine di migliorare le prestazioni del sistema.

OP 8 - mostrare NPCs conosciuti nella campagna con relativi dati.

Si valuta la possibilità di aggiungere una associazione fra *Campagna* e *NPC* che permette di risalire più velocemente agli NPC incontrati in una campagna.



Concetto	Costrutto	Accessi	Tipo
Campagna	E	1	L
Presenza	A	15	L
NPC	E	15	L
Totale: 31L → 155 al giorno			

Ricordando che l'operazione senza ridondanza costa 805 al giorno, è chiaro che la ridondanza migliora le prestazioni. Il vantaggio deriva soprattutto dal fatto che lo schema memorizza solo una volta ogni NPC nell'associazione *Presenza* con *Campagna*, cosa che invece non accadeva passando per le *Sessioni* perché un NPC poteva comparire in più *Sessioni*, e quindi questo vincolo non veniva espresso. Tuttavia, dobbiamo anche considerare che questa ridondanza complica le **OP 6 - aggiungere interazione con NPC sconosciuto** e **OP 7 - aggiungere interazione con NPC conosciuto** poiché l'inserimento richiede

la scrittura di un'ulteriore associazione. Allora, valutiamo di quanto si complicano tali operazioni.

OP 6			
Concetto	Costrutto	Accessi	Tipo
Comparsa	A	1	S
NPC	E	1	L
Sessione	E	1	L
Presenza	A	1	S
Totale: 2L + 2S → 6 al giorno			

OP 7			
Concetto	Costrutto	Accessi	Tipo
Comparsa	A	1	S
Presenza	A	1	S
Totale: 2S → 8 al giorno			

Ricordando che l'OP 6 senza ridondanza costa 4 al giorno e l'OP 7 senza ridondanza costa anch'essa 4 al giorno, significa che la ridondanza introduce complessivamente per queste due operazioni un costo di 6 in più al giorno. Al computo finale, risulta che la ridondanza migliora l'efficienza del sistema riducendo il costo delle operazioni di 644 al giorno. Decidiamo quindi di introdurre la ridondanza esposta.

OP 12 - verificare se un oggetto è in possesso del party

Come nel caso precedente, possiamo introdurre un'associazione fra *Oggetto* e *Party* per poter memorizzare gli oggetti distinti in possesso del party. Per poter calcolare il costo di questa operazione, è necessario sapere quanti oggetti distinti ha il party mediamente. Sappiamo che un party è composto mediamente da 6 protagonisti, e che ogni protagonista ha in media 10 oggetti: ipotizziamo che ogni protagonista contribuisce mediamente con 5 oggetti che sono solo in suo possesso. Tale ridondanza complica l'OP 10 - aggiungere un oggetto nuovo ad un personaggio, quindi analizziamo anche quella.



OP 12			
Concetto	Costrutto	Accessi	Tipo
Party	E	1	L
Possesso	A	30	L
Oggetto	E	30	L
Totale: 61L → 61 al giorno			

OP 10			
Concetto	Costrutto	Accessi	Tipo
Oggetto	E	1	S
Proprietà	A	1	S
Protagonista	E	1	L
Possesso	A	1	S
Party	E	1	L
Totale: 2L + 3S → 80 al giorno			

Al computo finale, si ha un costo di 141 al giorno con la ridondanza contro 183 al giorno senza ridondanza: tuttavia, notiamo che il vantaggio che deriva dalla soluzione con ridondanza è causato dalla nostra ipotesi che un personaggio possieda mediamente 5 oggetti distinti non appartenenti ad altri personaggi, e tale informazione non deriva da alcuna stima dei dati. Per cui, data la debolezza delle premesse e lo scarso vantaggio che introdurre ridondanza comporta, decidiamo di non introdurre ridondanza in questo caso.

3.6 Traduzione di entità e associazioni in relazioni

CAMPAGNE(nomeCampagna, numGiocatori)

SESSIONI(codProgressivo, nomeCampagna: CAMPAGNE, codParty: PARTIES)

TURNI(numTurno, codProgressivo: SESSIONI, nomeCampagna: CAMPAGNE)

PARTIES(codParty)

CLASSI(nomeClasse, descrizioneClasse)

ABILITA'(nomeAbilità, descrizioneAbilità, tipoAbilità)

OFFERTE_CLASSI(nomeAbilità: ABILITA', nomeClasse: CLASSI)

SOTTOCLASSI(numSottoclasse, nomeClasse: CLASSI, descrizioneSottoclasse)

OFFERTE_SOTTOCLASSI(nomeAbilità: ABILITA', numSottoclasse: SOTTOCLASSI, nomeClasse: CLASSI)

RAZZE(nomeRazza, descrizioneRazza)

OFFERTE_RAZZE(nomeAbilità: ABILITA', nomeRazza: RAZZE)

PROTAGONISTI(nomeProtagonista, *immagineProtagonista, oro, livello, num-
Sottoclasse: SOTTOCLASSI, nomeClasse: CLASSI, nomeRazza: RAZZE)

COMPOSIZIONI(codParty: PARTIES, nomeProtagonista: PROTAGONISTI)

NPCs(nomeNPC, *immagineNPC, descrizionePersonalità, nomeCampagna: CAM-
PAGNE, tipoInterazione)

COMPARE(nomeCampagna: CAMPAGNE, codProgressivo: SESSIONI, nomeNPC:
NPCs, tipoRapporto)

PARTECIPAZIONI_PROTAGONISTI(nomeProtagonista: PROTAGONISTI, numTurno:
TURNI, codProgressivo: SESSIONI, nomeCampagna: CAMPAGNE, danno-
Protagonista)

MOSTRI(nomeMostro, *immagineMostro)

PARTECIPAZIONI_MOSTRI(nomeMostro, numTurno: TURNI, codProgressivo:
SESSIONI, nomeCampagna: CAMPAGNE, dannoMostro)

OGGETTI(nomeOggetto, tipoOggetto, *numUtilizzi, descrizioneOggetto)

PROPRIETA'(nomeOggetto: OGGETTI, nomeProtagonista: PROTAGONISTI,
quantità)

Lo schema risulta in 3NF.

3.7 Schema relazionale finale

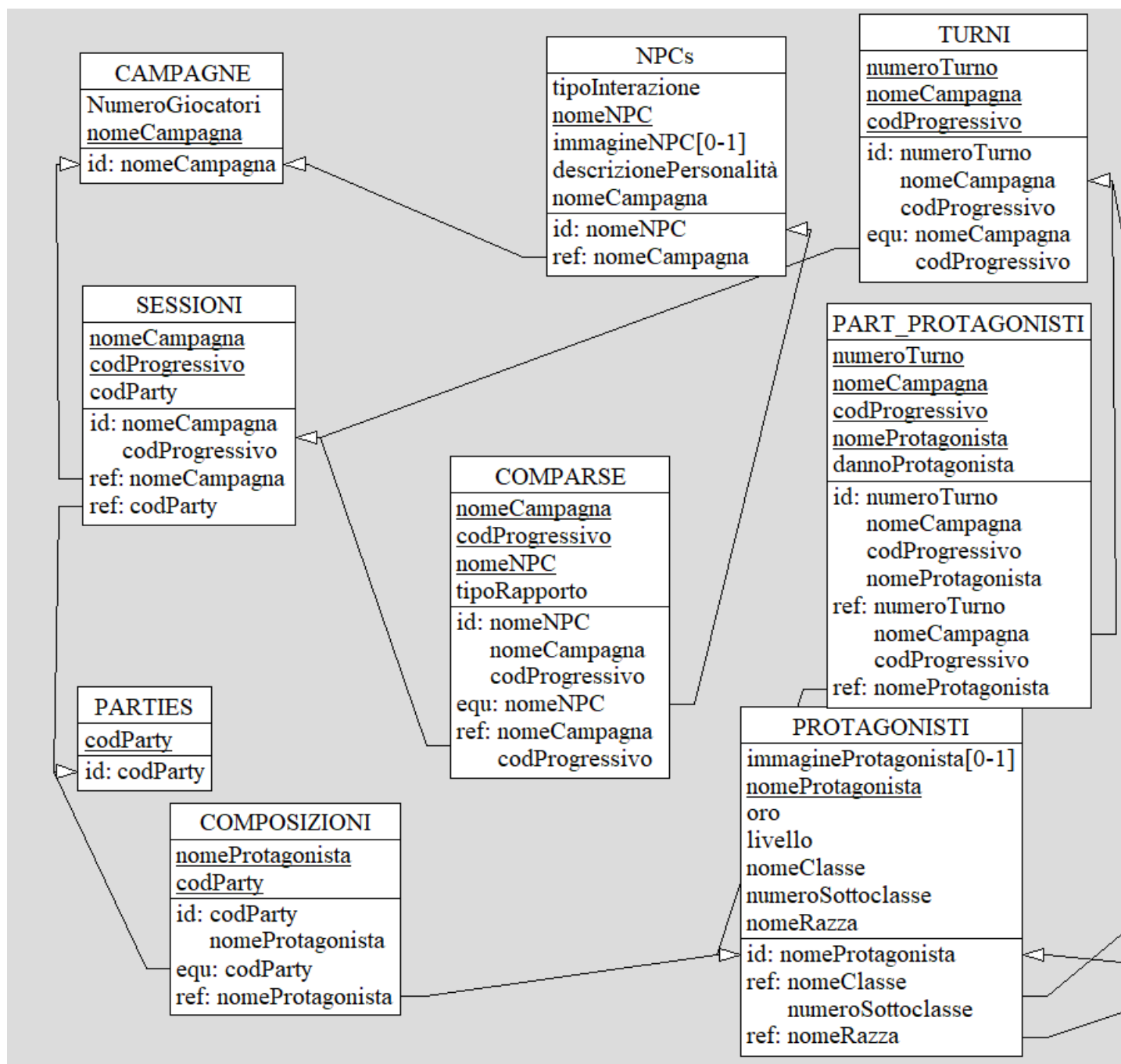


Figure 7: Parte sinistra dello schema relazionale

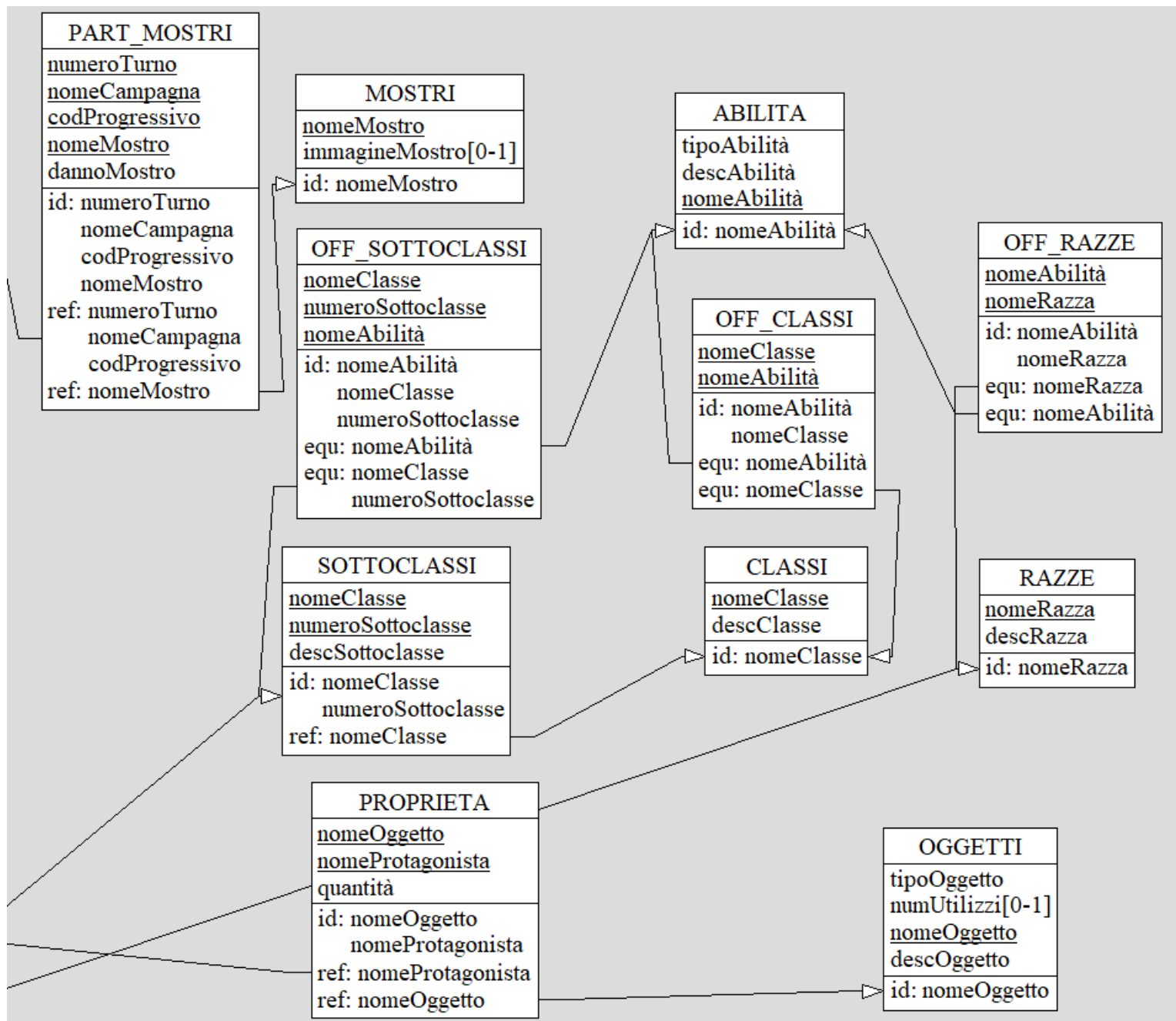


Figure 8: Parte destra dello schema relazionale

3.8 Traduzione delle operazioni in query SQL

OP 1 - creare una nuova campagna

```
INSERT INTO campagne (nomeCampagna, numGiocatori)
VALUES (?, ?)
```

OP 2 - creare una nuova sessione

```
INSERT INTO sessioni (codProgressivo, nomeCampagna, codParty)
VALUES(?, ?, ?)
```

OP 3 - creare un nuovo party

```
INSERT INTO parties(codParty)
VALUES(?)
```

OP 4 - registrare il turno di un personaggio o nemico

```
INSERT INTO turni(numeroTurno, nomeCampagna, codProgressivo)
VALUES(?, ?, ?)
```

Si richiede poi di inserire la partecipazione del personaggio o nemico: nella seguente query, registriamo il turno per il personaggio, ma la registrazione del mostro è analoga.

```
INSERT INTO partecipazioni_protagonisti(nomeProtagonista, numTurno, codProgressivo, nomeCampagna, dannoProtagonista)
VALUES(?, ?, ?, ?, ?)
```

OP 5 - mostrare la media di danni eseguita per uno specifico personaggio

```
SELECT AVG(dannoProtagonista) as 'Media danni per turno'
FROM partecipazioni_protagonisti
WHERE nomeProtagonista = ?
```

OP 6 - aggiungere interazione con NPC sconosciuto

Il primo passaggio consiste nell'inserire l'NPC dentro il database.

```
INSERT INTO npcs(nomeNPC, descrizionePersonalità, nomeCampagna, tipoInterazione)
VALUES(?, ?, ?, ?)
```

Secondariamente, registriamo l'interazione con il party

```
INSERT INTO comparse(nomeCampagna, codProgressivo, nomeNPC, tipoRapporto)
VALUES(?, ?, ?, ?)
```

OP 7 - aggiungere interazione con NPC conosciuto

Aggiorniamo l'interazione nella tabella delle comparse.

```
UPDATE comparse
SET tipoRapporto = ?
WHERE nomeNPC = ? AND nomeCampagna = ? AND codProgressivo = ?
```

Aggiorniamo anche l'interazione nella tabella dell'NPC.

```
UPDATE npcs
SET tipoInterazione = ?
WHERE nomeNPC = ?
```

OP 8 - mostrare NPCs conosciuti nella campagna con relativi dati

```
SELECT *
FROM npcs
```

WHERE nomeCampagna = ?

OP 9 - mostrare la composizione del party per razze, classi e/o sottoclassi

```
SELECT P.nomeProtagonista, P.nomeRazza, P.nomeClasse, P.numeroSottoclasse
FROM composizioni C JOIN protagonisti P ON (C.nomeProtagonista = P.nomeProtagonista)
WHERE codParty = ?
```

OP 10 - aggiungere un oggetto nuovo ad un personaggio

```
INSERT INTO proprietà(nomeOggetto, nomeProtagonista, quantità)
VALUES(?, ?, ?)
```

OP 11 - mostrare gli oggetti di un personaggio

```
SELECT P.quantità, O.*
FROM proprietà P JOIN oggetti O ON (P.nomeOggetto = O.nomeOggetto)
WHERE P.nomeProtagonista = ?
```

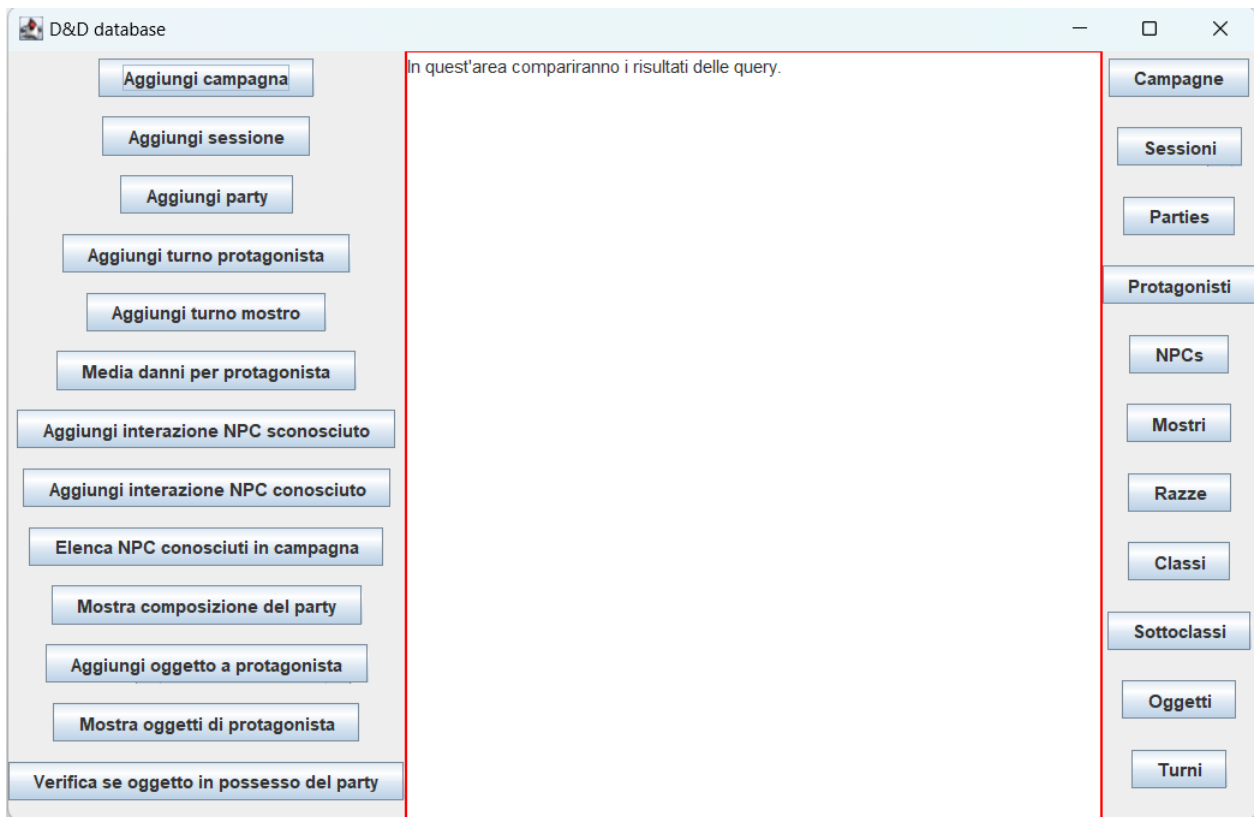
OP 12 - verificare se un certo oggetto è in possesso del party

```
SELECT EXISTS (SELECT PR.nomeOggetto
FROM composizioni C, protagonisti P, proprietà PR
WHERE C.nomeProtagonista = P.nomeProtagonista AND P.nomeProtagonista
= PR.nomeProtagonista AND C.codParty = ? AND PR.nomeOggetto = ? )
```

4 Progettazione dell'applicazione


4.1 Descrizione dell'architettura dell'applicazione realizzata

L'applicazione per interfacciarsi al database è stata realizzata in Java mediante JDBC. Il database risiede in locale e il DBMS usato è MySQL. L'applicazione è una semplice Java Swing application.



Sulla sinistra, vi sono dei tasti per le operazioni specificate nella relazione, mentre sulla destra vi sono dei tasti che permettono di vedere tutte le righe delle tabelle più importanti (es. premere sul tasto "Campagna" mostrerà tutte le campagne presenti nel database). Per tutte le query che richiedono l'inserimento di dati da parte dell'utente, è stato implementato un form sotto forma di finestra pop-up che chiede i dati all'utente. Nella figura seguente, un'esempio per l'operazione di inserimento turno di un protagonista.

Inserire numTurno, nomeCampagna, codProgressivo, nomeProtagonista e dannoPro... X

 numTurno:

nomeCampagna:

codProgressivo:

nomeProtagonista:

dannoProtagonista:

Link al repository: <https://github.com/MicheleCiavatti/DB2023-project.git>.