RELAZIONE DEL PROGETTO DI BASI DI DATI

PATHFINDER DB

Matteo Lotto, matricola 1100274

Giulio Rossetti, Matricola 1122603

INDICE

1. Abstract 2
2. Descrizione dei Requisiti 3
3. Progettazione Concettuale 3
3.1. Schema Concettuale ER 3
3.2 Lista delle Classi 4
3.3. Lista delle Associazioni 6
4. Progettazione Logica 6
4.1. Schema concettuale rifinito 6
4.2 Rifinitura 6
4.3 Schema Relazionale 7
4.4. Vincoli di Integrità 7
5. Implementazione dello Schema Logico 8
6. Trigger, Funzioni, Query e Procedure 10
6.1. Trigger 10
6.2. Funzioni 12
6.3. Query 13
6.4. Procedure 15

1. ABSTRACT

Pathfinder è un gioco di ruolo che prende ispirazione dal d20 System di Dungeons & Dragons per creare simili meccaniche, ma ugualmente complesse, che portano i personaggi da avere molti dati complessi e dispendiosi in tempo di ricerca di informazioni tra i vari manuali.

La comunità di Pathfinder Italiana richiede un sistema di ricerca dei personaggi giocanti che unisca le informazioni dei manuali esistenti da rendere disponibile in maniera rapida un sistema di memorizzazione e condivisione dei propri personaggi con il resto della comunità.

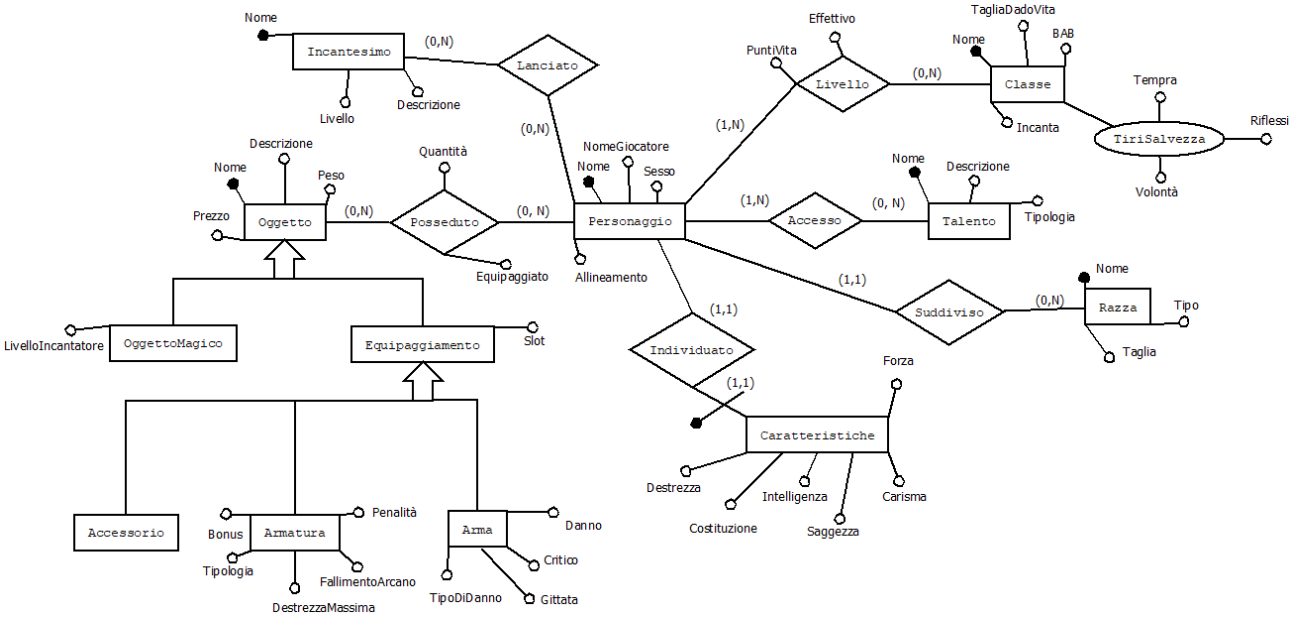
2. DESCRIZIONE DEI REQUISITI

Si vuole realizzare una base di dati per la gestione dei personaggi di Pathfinder per la comunità italiana.

Ogni personaggio deve essere identificato da un Nome, con un Sesso, un Allineamento, insieme al Nome del giocatore che lo ha creato.
Ogni personaggio appartiene ad una data razza, composta da un Nome, un Tipo che ne identifica una categoria di forma, e la Taglia, a indicare la dimensione media della razza.
Un personaggio può ottenere vari talenti durante le partite; tali Talenti sono composti da un Nome, una Descrizione del suo effetto, e una Tipologia, per suddividere i talenti legati al combattimento da quelli generali.
Di importanza è anche il livello di un personaggio: può ottenere uno o più livelli su varie classi, con ogni livello che aggiunge diversi punti vita; ogni classe è identificata da un Nome, da un TagliaDadoVita(un valore che rappresenta il numero di facce del dado da lanciare per calcolare i punti vita da ottenere ogni livello), un BAB(Bonus Attacco Base, un aumento al tiro per colpire base dato dalla classe), qualora sia una classe Incantatore (che può lanciare magie) o meno, e dai tre Tiri Salvezza: Tempra, Volontà e Riflessi (valori usati per valutare qualora la classe sia buona o no a resistere a diversi tipi effetti).Ci devono essere anche le caratteristiche del personaggio: sei interi che rappresentano la Forza, la Destrezza, la Costituzione, l'Intelligenza, la Saggezza e il Carisma.
Qualora un personaggio abbia dei livelli di una classe Incantatore, si devono poter vedere gli incantesimi selezionabili, ciascuno indicato da un Nome, un Livello e una Descrizione che ne rappresenta l'effetto.
Infine, un personaggio può possedere e aver equipaggiato degli oggetti: ciascuno identificato da un Nome, una Descrizione, un Peso, e un Prezzo. Qualora sia un oggetto magico, deve mostrare anche il Livello di Incantatore necessario per essere utilizzato. Se invece fosse un equipaggiamento, come un'arma, un'armatura o accessorio, si deve poter vedere lo Slot in cui vada equipaggiato: qualora sia un Armatura, deve mostrare anche le relative statistiche (Bonus, Penalità, Tipologia, Destrezza Massima e Fallimento Arcano), qualora sia un'Arma, deve mostrare il Danno che può infliggere, capacità di fare Critico, il Tipo di Danno che causa e la possibile Gittata, nel caso sia un'arma per l'attacco a distanza.

3. PROGETTAZIONE CONCETTUALE

3.1. Schema Concettuale ER



3.2 Lista delle Classi

Quasi tutti gli attributi delle classi sono NOT NULL poiché sono indispensabili per la caratterizzazione dell'entità rappresentata, fanno eccezione gli attributi di Descrizione di Oggetto, di Talento e di Incantesimo. Per definizione di Destrezza Massima, può essere con un valore NULL.

Ogni campo numerico è considerato senza segno, poiché la parte negativa è inutilizzata nella base di dati.

Classe: modella gli elementi distintivi di una classe

Attributi:

- Nome: String - Nome della classe
- TagliaDadoVita: ENUM (12,10,8,6) - Taglia del dado vita
- BAB: ENUM ('Full', 'ThreeQuarters', 'Half') - Bonus attacco base in funzione del livello
- Incanta: ENUM ('Si", 'Nò) – Determina se la classe ha incantesimi o meno
- Tempra: ENUM ('Good', 'Bad') - Qualità del tiro tempra
- Riflessi: ENUM ('Good', 'Bad') - Qualità del tiro riflessi
- Volontà: ENUM ('Good', 'Bad') - Qualità del tiro volontà

Incantesimo: modella le caratteristiche degli incantesimi

Attributi:

- Nome: String - Nome dell'incantesimo
- Livello: Int - Livello incantatore minimo
- Descrizione: String - Descrizione dell'effetto dell'incantesimo

Oggetto: modella gli oggetti dell'inventario

Attributi:

- Nome: String - Nome dell'oggetto
- Descrizione: String - Descrizione dell'oggetto
- Peso: INT - Peso dell'oggetto espresso in libbre (standard del regolamento)
- Prezzo: INT - Prezzo dell'oggetto espresso in monete di rame (valuta minima)

Sono considerate le seguenti sottoclassi di Oggetto.

1. Oggetto Magico: modella le caratteristiche aggiuntive degli oggetti magici
   Attributi:
   - LivelloIncantatore: INT - Livello incantatore richiesto per l'utilizzo dell'oggetto magico.

2. Equipaggiamento: modella gli oggetti indossabili
   Attributi:
   - Slot: ENUM ('Testà, 'Fronte', 'Occhi', 'Spallè, 'Collò, 'Toracè, 'Corpo', 'Armatura', 'Cintura', 'Polsi', 'Anello', 'Manò) - Parte del corpo su cui l'oggetto va indossato.

   Sono considerate le seguenti sottoclassi di Equipaggiamento.

1. Arma: modella le caratteristiche dell'arma
   Attributi:
   - Danno: String - Descrizione del danno dell'arma
   - Critico: String - Descrizione del critico dell'arma
   - Gittata: INT - Lunghezza, in piedi, della possibile gittata dell'arma
   - TipoDiDanno: String - Descrizione del tipo di danno possibile
2. Armatura: modella le caratteristiche dell'armatura
   Attributi:
   - Bonus: INT - Quantità di Classe Armatura aggiuntiva
   - DestrezzaMassima: INT - Destrezza massima applicabile finchè l'armatura è indossata
   - FallimentoArcano: INT - Possibilità di fallire gli incantesimi
   - Penalita: INT - Penalità applicabile alle prove come conseguenza dell'armatura
   - Tipologia: ENUM ('Leggera', 'Media', 'Pesante', 'Scudo') - Descrizione della tipologia di armatura
3. Accessorio: non ha caratteristiche aggiuntive

Personaggio: modella gli elementi indispensabili per la caratterizzazione del personaggio

Attributi:

- Nome: String - Nome del personaggio
- NomeGiocatore: String - Nome del giocatore
- Sesso: ENUM ('M', 'F') - Sesso del personaggio
- Allineamento: ENUM ('LB', 'NB', 'CB', 'LN', 'NN', 'CN', 'LM', 'NM', 'CM') - Descrive l'allineamento del personaggio secondo gli assi Legale/Caotico e Buono/Malvagio
- Forza: INT - Caratteristica di forza del personaggio
- Destrezza: INT - Caratteristica di destrezza del personaggio
- Costituzione: INT - Caratteristica di costituzione del personaggio
- Intelligenza: INT - Caratteristica d'intelligenza del personaggio
- Saggezza: INT - Caratteristica di saggezza del personaggio
- Carisma: INT - Caratteristica di carisma del personaggio

Razza: modella le caratteristiche distintive tra le razze

Attributi:

- Nome: String – Nome della razza
- Taglia: ENUM ('Colossal', 'Gargantuan', 'Huge, 'Largè, 'Medium', 'Small', 'Tiny', 'Diminutivè, 'Finè) – Taglia della razza
- Tipo: String – Tipologia della razza

Talento: modella i talenti e le capacità speciali ottenibili dal personaggio

Attributi:

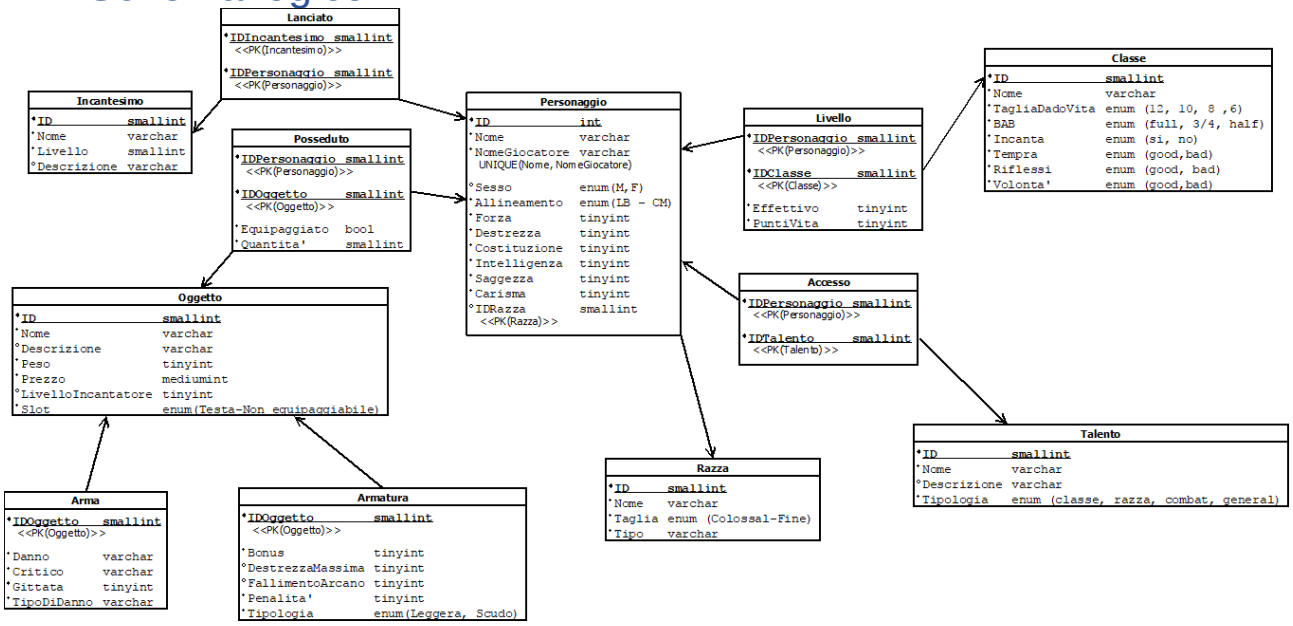
- Nome: String – Nome del talento o della capacità speciale
- Descrizione: String – Descrizione degli effetti del talento o della capacità speciale
- Tipologia: ENUM ('DiClassè, 'DiRazzà, 'Combat', 'General') – Tipologia del talento

3.3. Lista delle Associazioni

- Personaggio – Oggetto: Posseduto
  - Ogni personaggio possiede nessuno o almeno un' oggetto
  - Un oggetto può essere posseduto da uno o più personaggi
- Personaggio – Caratteristiche: Individuato
  - Ogni personaggio è individuato da una serie di caratteristiche
  - Le caratteristiche sono appartenenti a un solo personaggio
- Personaggio – Incantesimo: Lanciato
  - Ogni personaggio, la cui classe lo permette, può lanciare degli incantesimi
  - Un incantesimo può essere conosciuto e lanciato da uno o più personaggi
- Personaggio – Classe: Livello
  - Ogni personaggio ha almeno un livello in una classe
  - La classe è il livello del personaggio
- Personaggio – Talento: Accesso
  - Ogni personaggio ha accesso ad almeno un talento
  - I talenti possono appartenere a dei personaggi
- Personaggio – Razza: Suddiviso
  - Ogni personaggio appartiene ad una razza
  - Una razza può contenere uno o piu' personaggi
- Oggetto-Armatura: TipoArmatura
  - Ogni oggetto può essere un solo tipo di armatura
  - Ogni armatura è necessariamente un oggetto.
- Oggetto-Arma: TipoArma
  - Un oggetto può essere un'arma
  - Un'arma è un solo oggetto

4. PROGETTAZIONE LOGICA

4.1. Schema logico



4.2 Rifinitura

Abbiamo deciso di utilizzare codici numerici per identificare le varie entità, poiché sebbene i Nomi sono unici e obbligatori (valgono come chiavi), possono risultare in stringhe abbastanza lunghe (per esempio "Pergamena di Evoca Protoplasma Nero", un tipo di oggetto magico).

Per poter implementare lo schema in codice SQL, è necessario rimuovere le generalizzazioni, sostituendole con relazioni e entità. Per la prima generalizzazione parziale, dove alcuni oggetti possono essere oggetti magici o equipaggiamenti: dato che entrambi sono unici solo per un nuovo attributo, questa generalizzazione è stata rimossa portando i due attributi all'entità Oggetto. Per la seconda generalizzazione, sempre parziale, abbiamo creato entità separate tramite relazioni. Poiché gli equipaggiamenti possono essere delle armi, degli accessori e delle armature. Gli accessori non possiedono attributi esclusivi, quindi viene eliminata la entità, avendo già tutti gli attributi necessari in Equipaggiamento. Rimangono le entità Arma e Armatura, ciascuna con i propri attributi. Dato che esistono gli scudi, oggetti che risultano essere sia Armi che Armature, è possibile avere oggetti che appartengano a entrambe le categorie; per tale ragione Arma e Armatura diventano entità e vengono aggiunte due relazioni che le collegano a oggetto: Un Equipaggiamento può essere un'Arma, un'Armatura, entrambe o nessuna delle due.

4.3 Schema Relazionale

Legenda **Chiave Primaria**, Chiave Esterna

Classe (**ID**, Nome, TagliaDadoVita, BAB, Incanta, Tempra, Riflessi, Volontà)

Incantesimo (**ID**, Nome, Livello, Descrizione)

Oggetto (**ID**, Nome, Descrizione, Peso, Prezzo, Slot, LivelloIncantatore)

Armatura(**IDOggetto,** Bonus, Penalità, Tipologia, FallimentoArcano, DestrezzaMassima)

Arma(**IDOggetto**, Danno, Critico, Gittata, TipoDiDanno)

Personaggio (**ID**, Nome, NomeGiocatore, Sesso, Allineamento, Forza, Destrezza, Costituzione, Intelligenza, Saggezza, Carisma, IDRazza)

Razza (**ID**, Nome, Taglia, Tipo)

Talento (**ID**, Nome, Descrizione, Tipologia)

Accesso (**IDPersonaggio**, **IDTalento**)

Lanciato (**IDPersonaggio**, **IDIncantesimo**)

Livello (**IDPersonaggio**, **Effettivo**, IDClasse, PuntiVita)

Posseduto (**IDPersonaggio**, **IDOggetto**, Quantità, Equipaggiato)

4.4. Vincoli di Integrità

I vincoli di integrità referenziale sono:

-Fra IDOggetto in Armatura e la chiave di Oggetto

-Fra IDOggetto in Arma e la chiave di Oggetti

-Fra IDRazza e la chiave di Razza

-Fra IDPersonaggio in Accesso e la chiave di Personaggio; fra IDTalento e la chiave di Talento

-Fra IDPersonaggio in Lanciato e la chiave di Personaggio; fra IDIncantesimo e la chiave di Incantesimo

-Fra IDPersonaggio di Livello e la chiave di Personaggio; fra IDClasse e la chiave di Classe

-Fra IDPersonaggio di Posseduto e la chiave di Personaggio; fra IDOggetto e la chiave di Oggetto

5. IMPLEMENTAZIONE DELLO SCHEMA LOGICO

```
1.  /* Pulizia */
2.  DROP TABLE IF EXISTS Log;
3.  DROP TABLE IF EXISTS Accesso;
4.  DROP TABLE IF EXISTS Arma;
5.  DROP TABLE IF EXISTS Armatura;
6.  DROP TABLE IF EXISTS Lanciato;
7.  DROP TABLE IF EXISTS Livello;
8.  DROP TABLE IF EXISTS Posseduto;
9.  DROP TABLE IF EXISTS Personaggio;
10. DROP TABLE IF EXISTS Classe;
11. DROP TABLE IF EXISTS Incantesimo;
12. DROP TABLE IF EXISTS Oggetto;
13. DROP TABLE IF EXISTS Razza;
14. DROP TABLE IF EXISTS Talento;
15. /* Fine Pulizia */
16.
17. /* Tabelle */
18. CREATE TABLE IF NOT EXISTS Log
19. (
20. ID smallint UNSIGNED NOT NULL AUTO_INCREMENT,
21. Descrizione varchar(255) DEFAULT NULL,
22. PRIMARY KEY (ID)
23. )ENGINE=InnoDB;
24.
25. CREATE TABLE IF NOT EXISTS Classe
26. (
27. ID smallint UNSIGNED NOT NULL AUTO_INCREMENT,
28. Nome varchar(255) NOT NULL,
29. TagliaDadoVita ENUM ('12','10','8','6') NOT NULL,
30. BAB ENUM ('Full', 'ThreeQuarters', 'Half') NOT NULL,
31. Incanta ENUM ('Si', 'No') NOT NULL,
32. Tempra ENUM ('Good', 'Bad') NOT NULL,
33. Riflessi ENUM ('Good', 'Bad') NOT NULL,
34. Volontà ENUM ('Good', 'Bad') NOT NULL,
35. UNIQUE (Nome),
36. PRIMARY KEY (ID)
37. )ENGINE=InnoDB;
38.
39. CREATE TABLE IF NOT EXISTS Incantesimo
40. (
41. ID smallint UNSIGNED NOT NULL AUTO_INCREMENT,
42. Nome varchar(255) NOT NULL,
43. Livello tinyint UNSIGNED NOT NULL,
44. Descrizione varchar(1024) DEFAULT NULL,
45. UNIQUE (Nome),
46. PRIMARY KEY (ID)
47. )ENGINE=InnoDB;
48.
49. CREATE TABLE IF NOT EXISTS Oggetto
50. (
51. ID smallint UNSIGNED NOT NULL AUTO_INCREMENT,
52. Nome varchar(255) NOT NULL,
53. Descrizione varchar(1536) DEFAULT NULL,
54. Peso tinyint UNSIGNED NOT NULL,
```

```
Prezzo mediumint UNSIGNED NOT NULL,
LivelloIncantatore tinyint UNSIGNED DEFAULT NULL,
Slot ENUM ('Testa', 'Fronte', 'Occhi', 'Spalle', 'Collo', 'Torace', 'Corpo', 'Armatura', '
Cintura', 'Polsi', 'Anello', 'Mano', 'Non Equipaggiabile') NOT NULL DEFAULT 'Non Equipaggi
abile',
UNIQUE (Nome),
PRIMARY KEY (ID)
)ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Razza
(
ID smallint UNSIGNED NOT NULL AUTO_INCREMENT,
Nome varchar(255) NOT NULL,
Taglia ENUM ('Colossal', 'Gargantuan', 'Huge', 'Large', 'Medium', 'Small', 'Tiny', 'Diminu
tive', 'Fine') NOT NULL DEFAULT 'Medium',
Tipo varchar(255) NOT NULL,
UNIQUE (Nome),
PRIMARY KEY (ID)
)ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Talento
(
ID smallint UNSIGNED NOT NULL AUTO_INCREMENT,
Nome varchar(255) NOT NULL,
Descrizione varchar(1024) DEFAULT NULL,
Tipologia ENUM ('DiClasse', 'DiRazza', 'Combat', 'General'),
UNIQUE (Nome),
PRIMARY KEY (ID)
)ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Personaggio
(
ID smallint UNSIGNED NOT NULL AUTO_INCREMENT,
Nome varchar(255) NOT NULL,
NomeGiocatore varchar(255) NOT NULL,
Sesso ENUM ('M', 'F') NOT NULL,
Allineamento ENUM ('LB', 'NB', 'CB', 'LN', 'NN', 'CN', 'LM', 'NM', 'CM') DEFAULT 'NN',
Forza tinyint UNSIGNED NOT NULL DEFAULT 10,
Destrezza tinyint UNSIGNED NOT NULL DEFAULT 10,
Costituzione tinyint UNSIGNED NOT NULL DEFAULT 10,
Intelligenza tinyint UNSIGNED NOT NULL DEFAULT 10,
Saggezza tinyint UNSIGNED NOT NULL DEFAULT 10,
Carisma tinyint UNSIGNED NOT NULL DEFAULT 10,
IDRazza smallint UNSIGNED NOT NULL,
UNIQUE (Nome, NomeGiocatore),
PRIMARY KEY (ID),
FOREIGN KEY (IDRazza) REFERENCES Razza(ID) ON DELETE CASCADE
)ENGINE=InnoDB;

      CREATE TABLE IF NOT EXISTS Accesso
      (
      IDPersonaggio smallint UNSIGNED NOT NULL,
      IDTalento smallint UNSIGNED NOT NULL,
      PRIMARY KEY (IDPersonaggio, IDTalento),
      FOREIGN KEY (IDPersonaggio) REFERENCES Personaggio(ID) ON DELETE CASCADE,
      FOREIGN KEY (IDTalento) REFERENCES Talento(ID) ON DELETE CASCADE
      )ENGINE=InnoDB;

      CREATE TABLE IF NOT EXISTS Arma
      (
      IDOggetto smallint UNSIGNED NOT NULL,
      Danno varchar(9) NOT NULL,
      Critico varchar(8) NOT NULL,
      Gittata tinyint UNSIGNED NOT NULL DEFAULT 0,
      TipoDiDanno varchar(3) NOT NULL,
      PRIMARY KEY (IDOggetto),
```

```
118.        FOREIGN KEY (IDOggetto) REFERENCES Oggetto(ID) ON UPDATE CASCADE ON DELETE CASCADE
119.        )ENGINE=InnoDB;
120.
121.        CREATE TABLE IF NOT EXISTS Armatura
122.        (
123.        IDOggetto smallint UNSIGNED NOT NULL,
124.        Bonus tinyint UNSIGNED NOT NULL,
125.        DestrezzaMassima tinyint UNSIGNED DEFAULT NULL,
126.        FallimentoArcano tinyint UNSIGNED NOT NULL DEFAULT 0,
127.        Penalita tinyint UNSIGNED NOT NULL DEFAULT 0,
128.        Tipologia ENUM ('Leggera', 'Media', 'Pesante', 'Scudo') NOT NULL DEFAULT 'Leggera'
    ,
129.        PRIMARY KEY (IDOggetto),
130.        FOREIGN KEY (IDOggetto) REFERENCES Oggetto(ID) ON UPDATE CASCADE ON DELETE CASCADE
131.        )ENGINE=InnoDB;
132.
133.        CREATE TABLE IF NOT EXISTS Lanciato
134.        (
135.        IDIncantesimo smallint UNSIGNED NOT NULL,
136.        IDPersonaggio smallint UNSIGNED NOT NULL,
137.        PRIMARY KEY (IDPersonaggio, IDIncantesimo),
138.        FOREIGN KEY (IDIncantesimo) REFERENCES Incantesimo(ID) ON DELETE CASCADE,
139.        FOREIGN KEY (IDPersonaggio) REFERENCES Personaggio(ID) ON DELETE CASCADE
140.        )ENGINE=InnoDB;
141.
142.        CREATE TABLE IF NOT EXISTS Livello
143.        (
144.        IDClasse smallint UNSIGNED NOT NULL,
145.        IDPersonaggio smallint UNSIGNED NOT NULL,
146.        Effettivo tinyint UNSIGNED DEFAULT 0,
147.        PuntiVita tinyint UNSIGNED NOT NULL DEFAULT 1,
148.        PRIMARY KEY (IDPersonaggio, Effettivo),
149.        FOREIGN KEY (IDClasse) REFERENCES Classe(ID) ON DELETE CASCADE,
150.        FOREIGN KEY (IDPersonaggio) REFERENCES Personaggio(ID) ON DELETE CASCADE
151.        )ENGINE=InnoDB;
152.
153.        CREATE TABLE IF NOT EXISTS Posseduto
154.        (
155.        IDOggetto smallint UNSIGNED NOT NULL,
156.        IDPersonaggio smallint UNSIGNED NOT NULL,
157.        Equipaggiato boolean NOT NULL DEFAULT FALSE,
158.        Quantita smallint NOT NULL DEFAULT 0,
159.        PRIMARY KEY (IDPersonaggio, IDOggetto),
160.        FOREIGN KEY (IDOggetto) REFERENCES Oggetto(ID) ON DELETE CASCADE,
161.        FOREIGN KEY (IDPersonaggio) REFERENCES Personaggio(ID) ON DELETE CASCADE
162.        )ENGINE=InnoDB;
163.        /* Fine Tabelle */
```

6. TRIGGER, FUNZIONI, QUERY E PROCEDURE

6.1. Trigger

6.1.1) LevelUp e LevelCheck controllano la coerenza del livello con la classe relativa. LevelUp aiuta anche nella sequenziazione dei livelli consentendo l'inserimento di livelli senza specificare quale sia effettivamente, ma aggiungendoli in coda.

```
1.  DELIMITER $
2.  -- LevelUp
3.  DROP TRIGGER IF EXISTS LevelUp$
4.  CREATE TRIGGER LevelUp
5.  BEFORE INSERT ON Livello
6.  FOR EACH ROW
```

```
BEGIN
    -- Set del livello effettivo
    SET @temp := (SELECT MAX(Effettivo) FROM Livello WHERE IDPersonaggio = NEW.IDPersonaggio) + 1;
    IF (@temp <> NEW.Effettivo AND (SELECT COUNT(*) FROM Livello WHERE IDPersonaggio = NEW.IDPersonaggio AND Effettivo = NEW.Effettivo) <> 0)
    THEN
        SET NEW.Effettivo := @temp;
        INSERT INTO Log(Descrizione) VALUES (CONCAT ('È stato sequenziato il livello effettivo: ',
            NEW.Effettivo));
    END IF;
    -- Controllo dei punti vita
    SET @temp := CAST((SELECT TagliaDadoVita FROM Classe WHERE ID = NEW.IDClasse) AS UNSIGNED INTEGER);
    IF (NEW.PuntiVita = 0 OR
        NEW.PuntiVita > @temp)
    THEN
        INSERT INTO Log(Descrizione) VALUES (CONCAT ('I Punti vita non sono validi (',
            NEW.PuntiVita,
            '), sono stati settati a 1 ')
        );
        SET NEW.PuntiVita := 1;
    END IF;
    --
END$

-- LevelCheck
DROP TRIGGER IF EXISTS LevelCheck$
CREATE TRIGGER LevelCheck
BEFORE UPDATE ON Livello
FOR EACH ROW
BEGIN
    -- Controllo dei punti vita
    SET @temp := CAST((SELECT TagliaDadoVita FROM Classe WHERE ID = NEW.IDClasse) AS UNSIGNED INTEGER);
    IF (NEW.PuntiVita = 0 OR
        NEW.PuntiVita > @temp)
    THEN
        INSERT INTO Log(Descrizione) VALUES (CONCAT ('I Punti vita non sono validi (',
            NEW.PuntiVita,
            '), sono stati settati a 1 ')
        );
        SET NEW.PuntiVita := 1;
    END IF;
    --
END$
DELIMITER ;
```

6.1.2) ControlloArmatura fa il controllo sul tipo di armatura e sullo slot corrispondente dell'oggetto riferito

```
DELIMITER $
-- ControlloArmatura
DROP TRIGGER IF EXISTS ControlloArmatura$
CREATE TRIGGER ControlloArmatura
BEFORE INSERT ON Armatura
FOR EACH ROW
BEGIN
    SET @flag := 0;
    IF NEW.Tipologia = 'Scudo' AND (SELECT Slot FROM Oggetto WHERE ID = NEW.IDOggetto) <> 'Mano'
    THEN
        INSERT INTO Log(Descrizione)
        VALUES ('Slot inadatto per uno scudo'),
        SET @flag := 1;
```

```
    ELSEIF NEW.Tipologia <> 'Scudo' AND (SELECT Slot FROM Oggetto WHERE ID = NEW.IDOggetto
) <> 'Armatura'
    THEN
        INSERT INTO Log(Descrizione)
        VALUES ('Slot inadatto per una armatura');
        SET @flag := 2;
    END IF;
    IF @flag <> 0
    THEN
        SET NEW.IDOggetto := NULL;
    END IF;
END$

DELIMITER ;
```

6.2. Funzioni

6.2.1) CalcolaLivelloIncantatore: Ritorna il livello incantatore del personaggio passato per ID

```
DELIMITER $
-- CalcolaLivelloIncantatore
DROP FUNCTION IF EXISTS CalcolaLivelloIncantatore$
CREATE FUNCTION CalcolaLivelloIncantatore (IDPersonaggio smallint UNSIGNED)
RETURNS tinyint
DETERMINISTIC
READ SQL DATA
BEGIN
    RETURN (
        SELECT COUNT(*)
        FROM Livello
        JOIN Classe ON Classe.ID = Livello.IDClasse
        WHERE Classe.Incanta = 'Si'
    );
END$
DELIMITER ;
```

6.2.2) PulisciLivelliSuccessivi: Rimuove gli elementi il cui livello è maggiore di quello passato per parametro con il relativo personaggio passato per ID. Ritorna il numero di livelli tolti.

```
DELIMITER $
DROP FUNCTION IF EXISTS PulisciLivelliSuccessivi$
CREATE FUNCTION PulisciLivelliSuccessivi (ID_Personaggio smallint UNSIGNED,
Livello tinyint UNSIGNED)
RETURNS tinyint
DETERMINISTIC
MODIFIES SQL DATA
BEGIN
    SET @ret := (SELECT COUNT(*) FROM Livello WHERE IDPersonaggio = ID_Personaggio AND Eff
ettivo > Livello);
    IF @ret > 0
    THEN
        DELETE FROM Livello
        WHERE IDPersonaggio = ID_Personaggio
            AND Effettivo > Livello;
    END IF;

    RETURN @ret;
END$
DELIMITER ;
```

6.3. Query

6.3.1) Elencare gli oggetti di tipo arma posseduti da personaggi mostrando i relativi attributi, inclusi Nome e Descrizione

```
1.  CREATE VIEW ElencoArmiUtilizzate
2.  AS
3.      SELECT Nome, Danno, Critico, TipoDiDanno, Gittata, Descrizione
4.      FROM Oggetto i
5.      JOIN Arma w ON i.ID = w.IDOggetto
6.      JOIN Posseduto o ON i.ID = o.IDOggetto
7.      ORDER BY i.Nome ASC;
```

Nome	Danno	Critico	TipoDiDanno	Gittata	Descrizione
Arco Corto Composito	1d6	×3	P	70	Sono necessarie almeno due mani per usare un arco,...
Arco Lungo	1d8	×3	P	100	*NULL*
Arco Lungo Composito	1d8	×3	P	110	Sono necessarie almeno due mani per usare un arco,...
Ascia da Guerra Nanica	1d10	×3	T	0	Un'ascia da guerra nanica è dotata di una lama spe...
Balestra Pesante	1d10	19–20/×2	P	120	Una balestra pesante si carica girando una piccola...
Balestra Pesante	1d10	19–20/×2	P	120	Una balestra pesante si carica girando una piccola...
Colpo Senz'Armi	1d3	×2	C	0	Un personaggio di taglia Media infligge 1d3 Danni ...
Giavellotto	1d6	×2	P	30	Quest'arma è una lancia leggera e flessibile conce...
Lancia Corta	1d6	×2	P	20	*NULL*
Noccoliera ad Ascia (G)	1d6	×3	T	0	Le noccoliere ad ascia, solitamente usate in coppi...
Pugnale	1d4	19–20/×2	PT	10	Il pugnale è dotato di una lama lunga circa 30 cm....
Scudo Torre	0	0		0	Questo massiccio scudo di legno è alto quasi quant...
Spada Lunga	1d8	19–20/×2	T	0	*NULL*
Stocco	1d6	18–20/×2	P	0	È possibile utilizzare il talento Arma Accurata pe...

6.3.2) Elencare gli oggetti di tipo Armatura posseduti da personaggi, mostrando i relativi attributi, inclusi Nome e Descrizione

```
1.  CREATE VIEW ElencoArmatureUtilizzate
2.  AS
3.      SELECT Nome, Bonus, DestrezzaMassima, FallimentoArcano, Penalita, Tipologia, Descrizio
    ne
4.      FROM Oggetto i
5.      JOIN Armatura a ON i.ID = a.IDOggetto
6.      JOIN Posseduto o ON i.ID = o.IDOggetto
7.      WHERE a.Tipologia <> 'Scudo'
8.      ORDER BY a.Tipologia ASC;
```

Nome	Bonus	DestrezzaMassima	FallimentoArcano	Penalita	Tipologia	Descrizione
Cuoio Borchiato	3	5	15	1	Leggera	*NULL*
Giaco di Maglia	4	4	20	2	Leggera	*NULL*
Cotta di Maglia	6	2	30	5	Media	*NULL*
Armatura Completa	9	1	35	6	Pesante	Questa armatura comprende guanti d'arme, pesanti c...

6.3.3) Elencare gli oggetti di tipo Scudo posseduti da personaggi, mostrando i relativi attributi, inclusi Nome e Descrizione

```sql
CREATE VIEW ElencoScudiUtilizzati
AS
    SELECT Nome, Danno, Bonus, DestrezzaMassima, FallimentoArcano, Penalita, Tipologia, Descrizione
    FROM Oggetto i
    JOIN Armatura a ON i.ID = a.IDOggetto
    JOIN Arma w ON i.ID = w.IDOggetto
    JOIN Posseduto o ON i.ID = o.IDOggetto
    WHERE a.Tipologia = 'Scudo'
    ORDER BY i.Nome ASC;
```

Nome	Danno	Bonus	DestrezzaMassima	FallimentoArcano	Penalita	Tipologia	Descrizione
Scudo Torre	0	4	2	50	10	Scudo	Questo massiccio scudo di legno è alto quasi quant...

6.3.4) Elencare la lista di incantesimi usati dai personaggi, mostrandone gli attributi

```sql
CREATE VIEW ElencoIncantesimiUtilizzati
AS
    SELECT Nome, Livello, Descrizione
    FROM Incantesimo s
    JOIN Lanciato t ON s.ID = t.IDIncantesimo
    ORDER BY s.Livello ASC;
```

Nome	Livello	Descrizione
Messaggio	0	Conversazione sussurrata a distanza
Raggio di Gelo	0	Raggio che infligge 1d3 danni da freddo
Cura Ferite Leggere	1	Cura 1d8 danni +1/livello (max +5)
Cura Ferite Leggere	1	Cura 1d8 danni +1/livello (max +5)
Individuazione delle Porte Segrete	1	Rivela porte nascoste entro 18 m
Infliggi Ferite Leggere	1	Il proprio tocco infligge 1d8 danni +1/livello (ma...
Occhio del Bombarolo	1	Aumenta la gittata dell'arma da lancio; attacco +1
Panacea Polivalente	1	Si ottiene un effetto rilassante o divertente
Pugnale di Ghiaccio	1	Un pugnale perfetto di ghiaccio infligge +1 danni ...
Tiro Lungo	1	Conferisce bonus +3 metri all'incremento di gittat...
Vedere Allineamento	1	Scelto un allineamento, alla propria vista, le cre...
Cura Ferite Moderate	2	Cura 2d8 danni +1/livello (max +10)
Gelo Pungente	2	Il bersaglio è afflitto da gelo intenso
Trama Ipnotica	2	Affascina 2d4 + livello DV di creature
Fulmine	3	1d6 danni/livello da elettricità
Infliggi Ferite Moderate	3	Il proprio tocco infligge 2d8 danni +1/livello (ma...

6.3.5) Elencare le classi per ordine di utilizzo dai personaggi, mostrando la media dei livelli aggiunti per personaggio

```
1.  CREATE VIEW StatisticaClasseUsata
2.  AS
3.      SELECT c.Nome NomeClasse, COUNT(DISTINCT L.IDPersonaggio) NumeroUtilizzi, COUNT(*)/COU
    NT(DISTINCT L.IDPersonaggio) MediaLivelli
4.      FROM Livello l, Classe c
5.      WHERE l.IDClasse = c.ID
6.      GROUP BY c.ID
7.      ORDER BY NumeroUtilizzi DESC;
```

NomeClasse	NumeroUtilizzi	MediaLivelli
Ladro	2	4.5000
Ranger	2	3.5000
Fattucchiere	1	6.0000
Guerriero	1	6.0000
Monaco	1	6.0000
Alchimista	1	4.0000

6.3.6) Elencare i Personaggi, in ordine di nome, mostrando la loro ricchezza otale (la somma dei prezzi degli oggetti posseduti)

```
1.  CREATE VIEW RicchezzaPersonaggi
2.  AS
3.      SELECT c.nome, SUM(i.prezzo) AS Totale
4.      FROM Personaggio c
5.      JOIN Posseduto o ON c.ID = o.IDPersonaggio
6.      JOIN Oggetto i ON o.IDOggetto = i.ID
7.      GROUP BY c.ID
8.      ORDER BY c.Nome
```

nome	Totale
Belector	11000
Hegnar	900
Kasidra	17500
Pari O'Dispari	22500
Rogar Il LanciaTorre	6350
Yamira	15200

6.4. Procedure

6.4.1) CompattaLivelli: Compatta i livelli di un personaggio, assegnando l'Effettivo minore dipsonibile mantenendo la sequenza

```
1.  DELIMITER $
2.  -- CompattaLivelli
```

```
3.  DROP PROCEDURE IF EXISTS CompattaLivelli$
4.  CREATE PROCEDURE CompattaLivelli (Personaggio smallint UNSIGNED)
5.  DETERMINISTIC
6.  MODIFIES SQL DATA
7.  BEGIN
8.      SET @top := (SELECT COUNT(*) FROM Livello WHERE IDPersonaggio = Personaggio);
9.      SET @counter := 1;
10.
11.     WHILE @counter <= @top
12.     DO
13.         IF (SELECT COUNT(*) FROM Livello WHERE IDPersonaggio = Personaggio AND Effettivo =
    @counter) <> 1
14.         THEN
15.             UPDATE Livello
16.             SET Effettivo := (Effettivo - 1)
17.             WHERE IDPersonaggio = Personaggio
18.                 AND Effettivo > @counter;
19.         ELSE
20.             SET @counter := @counter + 1;
21.         END IF;
22.     END WHILE;
23. END$
24. DELIMITER ;
```

6.4.2) ElencoEquipaggiato: Mostra gli oggetti equipaggiati da un Personaggio

```
1.  DELIMITER $
2.  -- ElencoEquipaggiato
3.  DROP PROCEDURE IF EXISTS ElencoEquipaggiato$
4.  CREATE PROCEDURE ElencoEquipaggiato (Personaggio smallint UNSIGNED)
5.  DETERMINISTIC
6.  MODIFIES SQL DATA
7.  BEGIN
8.      SELECT Nome, Slot, Descrizione
9.      FROM Oggetto i
10.     JOIN Posseduto o ON i.ID = o.IDOggetto
11.     WHERE o.IDPersonaggio = Personaggio
12.     AND o.Equipaggiato = TRUE
13.     ORDER BY i.Slot ASC;
14. END$
15. DELIMITER ;
```