RELAZIONE BASI DI DATI

Relazione e progetto svolti dalla studentessa:

Gemini Rebecca

Matricola numero: …..

3° anno di Informatica Applicata

Corso:

Basi di Dati

ANALISI DEI REQUISITI

Un programmatore di videogiochi vuole tenere traccia di tutti i dati del gioco creato.

Il videogioco è composto da una serie di **campioni,** i personaggi utilizzabili nel videogioco, che in particolare presentano un ID, dal quale vengono identificati, il nome, la difficoltà, la classe (se attacca corpo a corpo o da lontano), il tipo (se infligge danni magici o fisici) e un ruolo di cui fa parte.

Ogni **ruolo** ha un ID, per identificarlo e un nome. Un ruolo indica dove il campione dovrà giocare in partita, ci sono in totale cinque ruoli (Corsia superiore, corsia centrale, giungla, supporto e ADC)

Per ogni singolo campione troviamo delle **statistiche** , che si riconoscono tramite un ID. Queste statistiche sono caratterizzate dalla salute del campione, l'armatura che ha, il danno fisico, il danno magico e la velocità di attacco.

Per ogni campione poi, c'è un **elenco di abilità**, composto da un ID che identifica l'abilità, il numero dell'abilità (ogni campione ha tre abilità diverse), una descrizione e una statistica che spiega cosa conferisce ad ogni campione.

Dopo di che, sono state create le **rune**, che conferiscono delle statistiche aggiuntive. Le rune sono composte da un ID per identificarle, un nome, una descrizione e una serie di statistiche per far capire a cosa serve ogni runa. Esistono due rune per ogni ruolo.

Sono stati anche creati degli **equipaggiamenti,** che vanno a seconda del ruolo di un campione. Questi equipaggiamenti sono determinati da un ID, un nome e una descrizione. Ci sono quattro equipaggiamenti per ogni ruolo.

Ogni campione, può avere almeno un **aspetto**, identificato con un ID, un nome e il suo livello di rarità. Gli aspetti non aggiungono nessuna statistica al campione, semplicemente gli fanno cambiare aspetto.

Gli aspetti si trovano all'interno di un **negozio.** Il negozio presenta un ID Articolo, con cui si contraddistingue l'articolo da poter comprare e il prezzo (in oro).

Questo gioco, è composto anche da dei giocatori che ne usufruiscono.

Ogni **giocatore** ha un ID univoco, un nome e l'oro totale che possiede.

Per ogni giocatore sono registrate anche le **partite giocate** , dove ogni giocatore ha almeno una partita giocata.

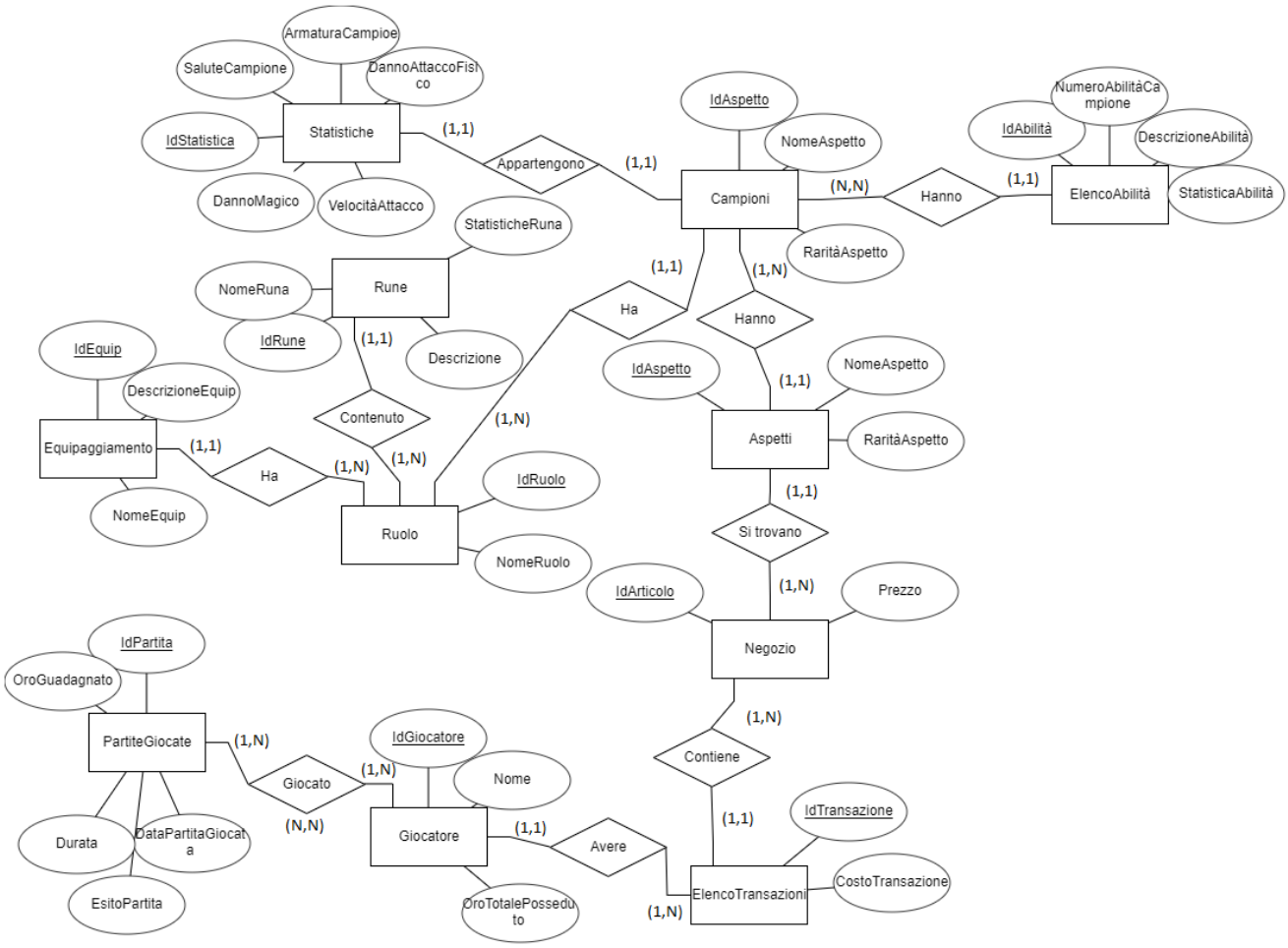
Le **partite giocate** hanno un ID identificativo, la data, l'esito (se ha vinto o perso), la durata in minuti e l'oro guadagnato. Vanno a gestire la parte che riguarda il mantenimento dei dati delle partite per ogni giocatore.

L'**elenco transazioni** è un elenco di transazioni effettuate dai giocatori. Ogni transazione è identificata da un ID e un costo in oro.

MODELLO E-R (modello entità relazionale)

E' un modello teorico per rappresentare concettualmente e graficamente dei dati con un alto livello di astrazione.

Dopo l'analisi dei requisiti, ne consegue il modello E-R:



Schema E-R creato con il sito “ERDPlus” : https://erdplus.com/

MODELLO LOGICO

Un modello logico stabilisce la struttura dei dati e le relazioni tra essi.

Incorpora tutti gli elementi di informazione che sono vitali per una gestione quotidiana di un DB.

In questo caso, tra Giocatore e PartiteGiocate, si crea una cardinalità (N,N), quindi verrà implementata la tabella PartiteGiocate_Giocatore, che mostrerà che i giocatori possono eseguire diverse partite.

Dopo lo schema E-R, ne consegue lo schema logico:

Aspetti(IdAspetto,NomeAspetto,RaritàAspetto,IdCampione)

Campioni(IdCampione,NomeCampione,DifficoltàCampione,ClasseCampione,TipoCampione,IdRuolo)

ElencoAbilità(IdAbilità,NumeroAbilitàCampione,DescrizioneAbilità,StatisticaAbilità,IdCampione)

ElencoTransazioni(IdTransazione,CostoTransazione,IdGiocatore,IdArticolo)

Equipaggiamento(IdEquip,NomeEquipaggiamento,DescrizioneEquip,IdRuolo)

Giocatore(IdGiocatore,Nome,OroTotalePosseduto)

Negozio(IdArticolo,Prezzo,IdAspetto)

PartiteGiocate(IdPartita,DataPartitaGiocata,EsitoPartita,Durata,OroGuadagnato,IdGiocatore)

Rune(IdRune,NomeRuna,Descrizione,StatisticheRuna,IdRuolo)

Ruolo(IdRuolo,NomeRuolo)

Statistiche(IdStatistica,SaluteCampione,ArmaturaCampione,DannoAttaccoFisico,DannoMagico,VelocitàAttacco,IdCampione)

PartiteGiocate_Giocatore(IdPartita,IdGiocatore)

SCELTE PROGETTUALI

I **campioni** presentano più **abilità**, mentre ogni abilità è associata unicamente ad un solo campione.

Una **statistica** appartiene unicamente solo ad un campione, e un campione ha solo quelle statistiche.

Alcuni campioni possono avere più di un **aspetto**, ma non tutti. Ogni aspetto è unico, ed appartiene esclusivamente ad un solo campione.

Esiste solo un **negozio**, contenente tutti gli aspetti.

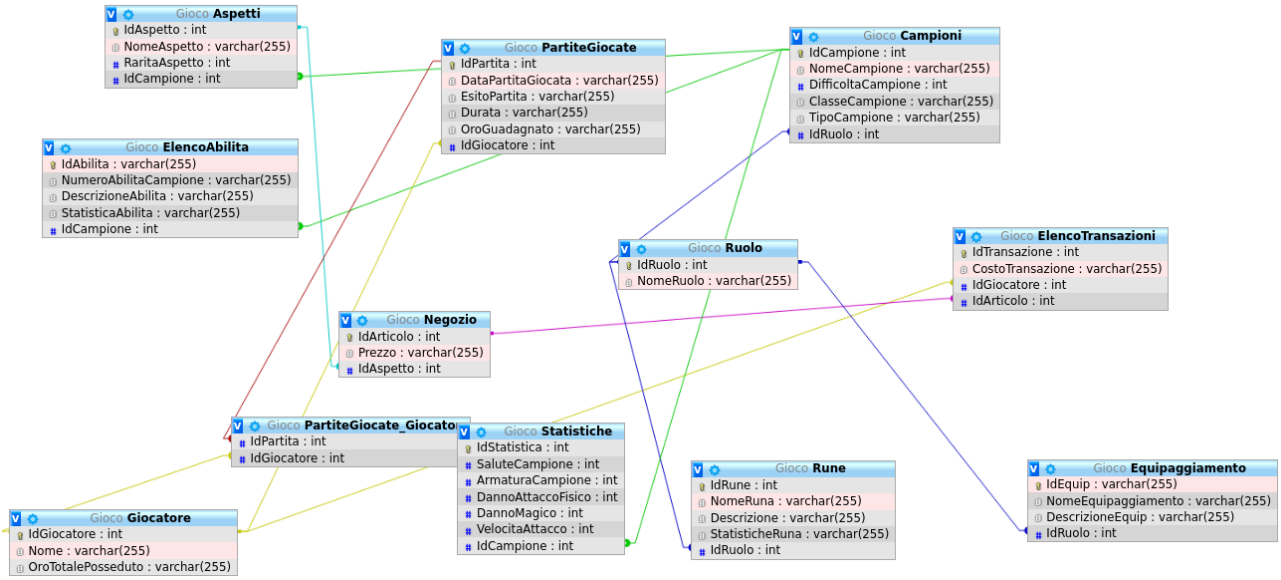
Nel negozio sono riportati anche gli **elenchi delle transazioni** di ogni giocatore. Ma non tutti i giocatori hanno effettuato delle transazioni.

Ogni **giocatore** ha giocato minimo ad una partita.

Ogni campione appartiene ad un singolo **ruolo**, mentre un ruolo può appartenere a più campioni.

Le singole **rune** possono essere utilizzate per un singolo ruolo, mentre per un ruolo ci sono diverse rune da poter utilizzare.

Un **equipaggiamento** può essere utilizzato in un singolo ruolo, mentre un ruolo può avere più di un equipaggiamento.



Per questo schema grafico ho utilizzato “phpmyadmin”

POPOLAZIONE DATABASE

Popolazione del database su mysql:

Tabelle normali: (ad esempio Campioni e Rune)

```
CREATE TABLE `Campioni` (
  `IdCampione` int NOT NULL,
  `NomeCampione` varchar(255) NOT NULL,
  `DifficoltaCampione` int NOT NULL,
  `ClasseCampione` varchar(255) NOT NULL,
  `TipoCampione` varchar(255) NOT NULL,
  `IdRuolo` int NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Rune` (
  `IdRune` int NOT NULL,
  `NomeRuna` varchar(255) NOT NULL,
  `Descrizione` varchar(255) NOT NULL,
  `StatisticheRuna` varchar(255) NOT NULL,
  `IdRuolo` int NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Aggiunta di chiavi primarie:

```
ALTER TABLE `Aspetti`
  ADD PRIMARY KEY (`IdAspetto`),
  ADD KEY `IdCampione` (`IdCampione`);

--
-- Indici per le tabelle `Campioni`
--
ALTER TABLE `Campioni`
  ADD PRIMARY KEY (`IdCampione`),
  ADD KEY `IdRuolo` (`IdRuolo`);

--
-- Indici per le tabelle `ElencoAbilita`
--
ALTER TABLE `ElencoAbilita`
  ADD PRIMARY KEY (`IdAbilita`),
  ADD KEY `IdCampione` (`IdCampione`);

--
-- Indici per le tabelle `ElencoTransazioni`
--
ALTER TABLE `ElencoTransazioni`
  ADD PRIMARY KEY (`IdTransazione`),
  ADD KEY `IdGiocatore` (`IdGiocatore`),
  ADD KEY `IdArticolo` (`IdArticolo`);

--
-- Indici per le tabelle `Equipaggiamento`
--
ALTER TABLE `Equipaggiamento`
  ADD PRIMARY KEY (`IdEquip`),
  ADD KEY `IdRuolo` (`IdRuolo`);

--
-- Indici per le tabelle `Giocatore`
--
ALTER TABLE `Giocatore`
  ADD PRIMARY KEY (`IdGiocatore`);

--
-- Indici per le tabelle `Negozio`
--
ALTER TABLE `Negozio`
  ADD PRIMARY KEY (`IdArticolo`),
  ADD KEY `Negozio_ibfk_1` (`IdAspetto`);

--
-- Indici per le tabelle `PartiteGiocate`
--
ALTER TABLE `PartiteGiocate`
  ADD PRIMARY KEY (`IdPartita`),
  ADD KEY `IdGiocatore` (`IdGiocatore`);
```

Aggiunta di chiavi esterne:

```sql
ALTER TABLE `Aspetti`
  ADD CONSTRAINT `Aspetti_ibfk_1` FOREIGN KEY (`IdCampione`) REFERENCES `Campioni` (`IdCampione`);

--
-- Limiti per la tabella `Campioni`
--
ALTER TABLE `Campioni`
  ADD CONSTRAINT `Campioni_ibfk_1` FOREIGN KEY (`IdRuolo`) REFERENCES `Ruolo` (`IdRuolo`);

--
-- Limiti per la tabella `ElencoAbilita`
--
ALTER TABLE `ElencoAbilita`
  ADD CONSTRAINT `ElencoAbilita_ibfk_1` FOREIGN KEY (`IdCampione`) REFERENCES `Campioni` (`IdCampione`);

--
-- Limiti per la tabella `ElencoTransazioni`
--
ALTER TABLE `ElencoTransazioni`
  ADD CONSTRAINT `ElencoTransazioni_ibfk_1` FOREIGN KEY (`IdGiocatore`) REFERENCES `Giocatore` (`IdGiocatore`),
  ADD CONSTRAINT `ElencoTransazioni_ibfk_2` FOREIGN KEY (`IdArticolo`) REFERENCES `Negozio` (`IdArticolo`);

--
-- Limiti per la tabella `Equipaggiamento`
--
ALTER TABLE `Equipaggiamento`
  ADD CONSTRAINT `Equipaggiamento_ibfk_1` FOREIGN KEY (`IdRuolo`) REFERENCES `Ruolo` (`IdRuolo`);

--
-- Limiti per la tabella `Negozio`
--
ALTER TABLE `Negozio`
  ADD CONSTRAINT `Negozio_ibfk_1` FOREIGN KEY (`IdAspetto`) REFERENCES `Aspetti` (`IdAspetto`) ON DELETE RESTRICT ON UPDATE RESTRICT;

--
-- Limiti per la tabella `PartiteGiocate`
--
ALTER TABLE `PartiteGiocate`
  ADD CONSTRAINT `PartiteGiocate_ibfk_1` FOREIGN KEY (`IdGiocatore`) REFERENCES `Giocatore` (`IdGiocatore`);
```

Per la struttura grafica del database ho utilizzato il sito: phpmyadmin

QUERY SQL

Ho creato delle query su mysql per testare la correttezza, per testare le relazioni tra tabelle e chiavi esterne.

1) Trovare tutti i nomi dei giocatori che possiedono più di 50 di oro:

```
mysql> SELECT Giocatore.Nome
    -> FROM Giocatore
    -> WHERE Giocatore.OroTotalePosseduto
    -> > 50;
+--------------+
| Nome         |
+--------------+
| Treecher     |
| RiseUp       |
| EarthMother  |
| KaboomView   |
| HoltHamlet   |
| Gerbilator   |
| Oblique      |
| TrueGrit     |
| Mutant       |
| Bulletproof  |
| Mania00      |
| ImpPlant     |
| FrenzyMan    |
| Explosssive  |
| BigDip       |
| CarloXXVII   |
| Alkanoid     |
| Plover       |
| ZetanChamp   |
| MuttonChops  |
| Incandescent |
| Astropower   |
| Octopi       |
| Warlockk     |
| Belizard     |
| Minkx        |
| CrosStorm    |
| Indira       |
| HeroineIsm   |
| Outfielder   |
| Profusser    |
| Sceptre      |
| Quibble      |
| Overseer     |
| Salamandrine |
| BBGun        |
| Commandame   |
| Possumiss    |
| MusicMiss    |
| Seashanty    |
```

2) Trovare tutti i nomi dei campioni che fanno parte del ruolo “Corsia Centrale”:

```
mysql> SELECT NomeCampione
    -> FROM Campioni, Ruolo
    -> WHERE Campioni.IdRuolo = Ruolo.IdRuolo
    -> AND NomeRuolo = "Corsia Centrale";
+--------------+
| NomeCampione |
+--------------+
| Ahri         |
| Annie        |
| Brand        |
| Cassiopeia   |
| Ekko         |
| Katarina     |
| Lissandra    |
| Neeko        |
| Orianna      |
| Viktor       |
| Zoe          |
+--------------+
11 rows in set (0,00 sec)
```

3) Trovare il nome dei giocatori che hanno vinto almeno una partita:

```
mysql> SELECT DISTINCT(Nome)
    -> FROM Giocatore,PartiteGiocate
    -> WHERE Giocatore.IdGiocatore = PartiteGiocate.IdGiocatore
    -> AND EsitoPartita = 'Vittoria';
+--------------+
| Nome         |
+--------------+
| Treecher     |
| EarthMother  |
| KaboomView   |
| Redshock     |
| Helixo       |
| Oblique      |
| HornofPlenty |
| Berserker    |
| Shotgunner   |
| Bulletproof  |
| Mania00      |
| Sw449Sn4ky   |
| ImpPlant     |
| Tweedlex     |
| FrenzyMan    |
| BigDip       |
| SmokinGun    |
| Alkanoid     |
| Ouster       |
| Incandescent |
| Astropower   |
| Octopi       |
| JossStick    |
| Minkx        |
| CrosStorm    |
| Outfielder   |
| Sceptre      |
| Overseer     |
| SepiaTone    |
| Opally       |
| MusicMiss    |
+--------------+
31 rows in set (0,00 sec)
```

4) Trovare il nome delle rune apposite per il ruolo “Giungla”:

```
mysql> SELECT NomeRuna
    -> FROM Rune,Ruolo
    -> WHERE Rune.IdRuolo = Ruolo.IdRuolo
    -> AND NomeRuolo = 'Giungla'
    -> ;
+---------------+
| NomeRuna      |
+---------------+
| Primo Attacco |
| Scatto fasico |
+---------------+
2 rows in set (0,00 sec)
```

5) Trovare il nome dei giocatori che non hanno mai vinto una partita

```
mysql> SELECT DISTINCT(Nome)
    -> FROM Giocatore,PartiteGiocate
    -> WHERE Giocatore.IdGiocatore = PartiteGiocate.IdGiocatore
    -> AND Giocatore.IdGiocatore NOT IN (
    -> SELECT PartiteGiocate.IdGiocatore
    -> FROM PartiteGiocate
    -> WHERE EsitoPartita = 'Vittoria');
+--------------+
| Nome         |
+--------------+
| RiseUp       |
| Sharkgirl    |
| BatBoy       |
| HoltHamlet   |
| Gerbilator   |
| TrueGrit     |
| Mutant       |
| Nightshade   |
| Masher       |
| Dynamite     |
| BamSnap      |
| RebRK800     |
| Explosssive  |
| Belizard     |
| CarloXXVII   |
| Plover       |
| ZetanChamp   |
| MuttonChops  |
| Warlockk     |
| Indira       |
| HeroineIsm   |
| Profusser    |
| Quibble      |
| Salamandrine |
| BBGun        |
| Commandame   |
| Possumiss    |
| Seashanty    |
+--------------+
28 rows in set (0,00 sec)
```

6) Trovare gli ID dei campioni che hanno come salute campione meno di 650:

```
mysql> SELECT Campioni.IdCampione
    -> FROM Campioni,Statistiche
    -> WHERE Campioni.IdCampione = Statistiche.IdCampione
    -> AND SaluteCampione < 650;
+------------+
| IdCampione |
+------------+
|          2 |
|          4 |
|          6 |
|          8 |
|          9 |
|         10 |
|         11 |
|         15 |
|         17 |
|         18 |
|         19 |
|         22 |
|         23 |
|         24 |
|         25 |
|         26 |
|         29 |
|         31 |
|         33 |
|         34 |
|         35 |
|         38 |
|         41 |
|         44 |
|         45 |
|         47 |
|         50 |
+------------+
27 rows in set (0,00 sec)
```