# Dungeons and Dragons Database

Group 42 - Aaron Ennis and Brad Powell

### URL to access web application

<http://flip3.engr.oregonstate.edu:4733/>

### Updates since last version

Changed the “armor” section to allow for NULL for classes (like wizard) who can’t equip any armor.

Updated the default for “level” in Character\_Class to 1 (the character wouldn’t be linked to the class if it didn’t have at least one level).

### Feedback by the peer reviewer

“Hey Brad, I like this idea for a project and I think that it works well as a database. I don't see any glaring issues, so here are a few things that I think could be changed:

* You might want to allow for more characters for the character's backstory, 100 characters isn't a lot.
* The character should have more unique attributes to it, like height or weight; just because the race has an average height or weight doesn't mean each character of that race is going to be exactly that same height.
* This might be outside the scope of your project, but I think it would be cool if you stored actual player statistics in your database; things like a characters dexterity or charisma ratings might make this database more functional to replace a character sheet, if that is your goal.
* I noticed that in your schema the arrow from character\_class and Class is misaligned, making it look like there are two arrows from character ID, when one should come from the class ID.

Good luck on your project!”

“I like the project it's very detailed. I know nothing about D&D but it seems complicated enough to be worthy of a database!

A couple of things I noticed:

The player role seems somewhat confusing, being that a player can be related to other players via one player being a DM. Those relationships also aren't apparent from the schema. To me it would make a lot more sense if there was Game entity, which would link with a single player ID as DM, (essentially a one-to-one relationship), and link to a table relating non-DM Players to Games. Essentially then you could remove "role" as a player attribute because if they are a DM they will be linked as such to a Game. Even if you are only planning your database to have one Game, then it may work out better to have a separate "DM" entity that can be linked to a player ID, instead of having that be an attribute. This would allow you to more easily enforce the "only one DM" rule.

To me it doesn''t seem like you want to have the relationship tables (Character\_Class and Race\_Special) in the ERD, Conceptually they don't really represent entities, they represent a relationship between two different entities. For example simply having Character and Class, or Race and Special, connected together with many-many connectors would covey the same high-level information and be a lot less cluttered. Then your schema, which gets more into the low-level implementation than the ERD, would show just how you would create those many-many relations with tables.”

“Awesome idea for your project, a true role-playing game classic. I really like the organization of the entities, their respective attributes, and the relationships between them. I actually think the inclusion of the Race\_Special and Character\_Class relationship entities help clarify the relationships, especially since the Character\_Class relationship has different relationships flanking it. For examples, the relationships involving the Character\_Class entity, a character must have at least one Character\_Class | < and each Character\_Class must have exactly one class | |.. This is also shown since I assume that as a character levels up, their character class may be updated and the class may as well. So in this sense, I think they help more than hinder the clarity of your ER diagram.

I noticed in the project outline description it stated, "keeps track of players, with whom the players play, and their characters". Based on the database, I can see how the players and their characters are kept track, but I'm a little unsure about how "with whom the players play" would be included since there is no entity that keeps a record/time stamp of who is playing with whom. If you wanted to, you could have another entity that acts as a record with attributes such as the players, time, and you could retrieve the stats (such as their level) of the player at that particular time. This sounds somewhat like an extension of the database you have now which primarily stores the players, and their different defining properties.”

### Actions based on the feedback

Re-aligned the arrow from Class ID in Character\_Class.

In 5th edition D&D, a “background” is a single word or short phrase descriptor of the character’s life before becoming an adventurer. These are things like “Soldier” or “Hermit.” We kept this attribute at 100 characters, but added clarification to the outline that this is a game mechanic, not a narrative backstory.

The purpose of the database is to show a snapshot of the players and the types of characters they play, so adding a character’s individual height/weight or the like isn’t very useful information to present. This goes for displaying the character’s stats as well.

We decided to keep the relationship tables in the ER diagram. One reviewer disliked them and another liked them. In particular, the Character\_Class table has additional information beyond linking up ID’s in a many-to-many relationship.

Two reviewers pointed out that the recursive relationship between regular players and DMs wasn’t clearly expressed in the schema, and one reviewer pointed out that the database is supposed to show who plays with whom. Instead of adding a new Game entity, we decided the most efficient way to convey this information is to replace the “role” attribute with a “DM” attribute that contains the Player ID of whomever is DMing for that player (or the DM him or herself in the case where the Player ID matches the DM attribute). A query on a particular DM can show who all is playing together.

### Upgrades to the Draft Version

Took the s’s off of the entities to keep everything singular and consistent.

Split player name into first and last name.

### Fixes based on Feedback from Step 1:

We have not received feedback from the grader for step 1.

### Project Outline

This is a database for many game elements for Dungeons and Dragons 5th Edition (D&D). While there are many items in the game that are out of the scope of this project, like individual weapons and armor, it does store information on the defining attributes of a D&D character. The database also keeps track of players, with whom the players play, and their characters.

### Database Outline

### The entities in our database are:

* Player -- These are the people playing our imaginary game of Dungeons and Dragons. Each player has the following attributes:
  + ID: The primary key by which we keep track of the players. This will be automatically assigned and auto-incrementing as players are added.
  + Player First Name: The first name of the person playing the game (as opposed to character name). This will be a string of up to 100 characters which can’t be left blank and has no default.
  + Player Last Name: The last name of the person playing the game. This will be a string of up to 100 characters which can’t be left blank and has no default.
  + DM: The Player ID of this player’s DM. If the particular player is the DM, then the Player ID and the DM attribute will match.
* Character -- Also known as Player Characters (PC’s), these are the in-game characters controlled by the players. Each character has the following attributes:
  + ID: A number automatically assigned and auto-incrementing to serve as the primary key.
  + Player ID: This is a foreign key from the Player entity that identifies which player owns the character.
  + Character Name: The name of the character, which is a string of up to 100 characters and can’t be left blank nor does it have a default.
  + Race: The species of the character, such as human, elf, or half-orc. This is a string of up to 10 characters and will contain a Race ID (as a foreign key) from the Race entity. It cannot be left blank and there is no default.
  + Background: Each character has a background to represent their lives before they became adventurers, whether they were soldiers, charlatans, or anything in between. This is a one word or short phrase descriptor, not a narrative backstory. This will be a string of up to 100 characters and can’t be left blank and there is no default.
* Character Class – Contains all of the classes that a character possesses. This is a composite/bridge entity that relates Character to Class using primary keys from those entities with a couple additional attributes describing the class further as it relates to the particular character as follows:
  + Level: The level held by a particular character for a particular class. This is an integer value. It cannot be blank, and defaults to 1.
  + Primary Class: Indicates if the tuple/row/entry represents the character’s primary class. This is a boolean value. It cannot be blank, and the default is FALSE.
* Class -- The “type” or “profession” of the character and the thing that defines the character’s playstyle the most. Each class has the following attributes:
  + ID: A number automatically assigned and auto-incrementing to serve as the primary key.
  + Class Name: The name of the class. This is a string of up to 20 characters. It cannot be left blank and there is no default. Examples include barbarian and rogue.
  + Hit Die: The die rolled each level to determine the character’s hit points. The higher the die the more hit points the character tends to have, so the more attacks they can take in combat. This will be an int of either 6, 8, 10, or 12 for the number of sides of the die. It cannot be left blank and there is no default.
  + Armor: The heaviest armor the class can wear. It will be either light, medium, heavy, or null (for classes like Wizard who can’t wear armor). It defaults to NULL.
  + Saving Throw 1: Each class has two “saving throws” tied to an attribute that helps them avoid damage or other adverse effects. This will be a string of up to 15 characters. It cannot be left blank and there is no default.
  + Saving Throw 2: The second saving throw the class has. This will be a string of up to 15 characters. It cannot be left blank and there is no default.
* Race -- The species of the character. Each race has the following attributes:
  + ID: A number automatically assigned and auto-incrementing to serve as the primary key.
  + Race Name: The named race. This is a string of up to 10 characters. It cannot be blank and there is no default. Examples include orc, elf, human, and halfling.
  + Lifespan: The average racial lifespan in years. This is an unsigned integer value. It is unsigned to allow for “immortal” designation with a value of -1. It cannot be blank, and there is no default.
  + Height: The average height for the race in meters. This is a floating point value. It cannot be left blank, and there is no default.
  + Weight: The average weight for the race in kilograms. This is a floating point value. It cannot be blank and there is no default value.
  + Speed: The average speed for the race in meters per second. This is a floating point value, and there is no default.
* Special -- Special attributes granted to races and their descriptions.
  + ID: A number automatically assigned and auto-incrementing to serve as the primary key.
  + Special Name: This is the name of the special racial attribute. It is a string of up to 50 characters. It cannot be blank, and there is no default.
  + Special Description: This is a brief description of the special attribute. It is a string of up to 255 characters. It cannot be blank.

The relationships in this database are:

* Players can have many characters – Players may create more than one character to round off the party or for a change of pace. This is a *one-to-many* relationship as each character belongs to only one player.
* Players can have other players – This a recursive *one-to-many­* relationship used to denote that a Dungeon Master (DM) can have many players, but a player can have at most one DM because in this project universe, it is considered base treachery for a player to have more than one DM.
* Dungeon Masters have several players – DMs and players cannot exist without each other. This is a *one-to-many* relationship, as it is considered base treachery in this Project Universe to have more than one DM.
* Characters have classes – Each time a character levels up, the player chooses a class in which to gain experience. A character starts with one class, but can gain others as the game goes on, so this is a *many-to-many* relationship.
* Characters have a race – A character can have only one race, but a race can be assigned to many characters, so this is a one-to-many relationship.
* Races have special attributes – Each race can have more than one special attribute, and it is possible that a special attribute can be assigned to more than one race. This is a many-to-many relationship.

### Entity-Relationship Diagram:



### Schema:

|  |  |  |  |
| --- | --- | --- | --- |
| Player | | | |
| player\_id | player\_first\_name | player\_last\_name | dm |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Character | | | | |
| character\_id | player\_id | character\_name | race\_id | background |

|  |  |  |  |
| --- | --- | --- | --- |
| Character\_Class | | | |
| character\_id | class\_id | level | primary\_class |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class | | | | | |
| class\_id | class\_name | hit\_die | armor | saving\_throw\_1 | saving\_throw\_2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Race | | | | | |
| race\_id | race\_name | lifespan | height | weight | speed |

|  |  |
| --- | --- |
| Race\_Special | |
| race\_id | special\_id |

|  |  |  |
| --- | --- | --- |
| Special | | |
| special\_id | special\_name | special\_description |