The Solace Database

Kevin Haupt and Andrew Goldman

12/9/15

Databases - CMPT 307

**Project Introduction**

The Solace database is a information browser and backup for a virtual tabletop roleplaying game (TTRPG). The game in question is to be the senior project of Kevin Haupt, so to help explain the problem of the database, we will paraphrase the proposal for the senior project.

At this point you might be wondering what a role playing game (RPG) is. The most simple definition is a game in which a player assumes a ‘role’ or plays as if they were another person, a character, in some fictionalized setting. These can take place across a variety of mediums. These include [video games such as the Final Fantasy](https://en.wikipedia.org/wiki/Final_Fantasy) series, a collection which in itself contains different genres. Most are offline single player games, while there is a massively multiplayer online RPG (MMORPG, or MMO) within the intellectual property as well. TTRPG’s, which predate the advent of personal computing, have carried down traditions of being played with pencil and paper.

At their core, RPG’s contain information about a character, representing how strong, fast, or magical the character is, representing these values with numbers. These numbers are then used to calculate the success or power of the actions that a player makes their character perform. RPG’s often contain elements of chance which sets them apart from games like chess. In chess, a player’s move is either legal, or illegal, but they will never fail. In RPG’s, it is typically possible that an action taken by a player can fail or misfire in some fashion. The most common method of adding chance to these games is by rolling dice.

The attributes and data of RPGs lend themselves naturally to database design. A game has characters, items, spells, etc. While most of these game files will be common between each installation of the game, it makes sense to save user created content like characters and items in a database, both for data backup, browsing, and analysis.

One problem this database exists to solve is the loss of game data on a client computer. Ideally, the database should contain at least the minimum of information necessary to reconstruct a game that was lost locally. An additional goal is to aggregate statistics about characters across different games in order to determine player preference and possibly the balance or fairness of game mechanics. A third opportunity is to provide a way for players to view what others have done with the game and thereby giving them the opportunity to improve their experience by introducing new ideas and concepts.

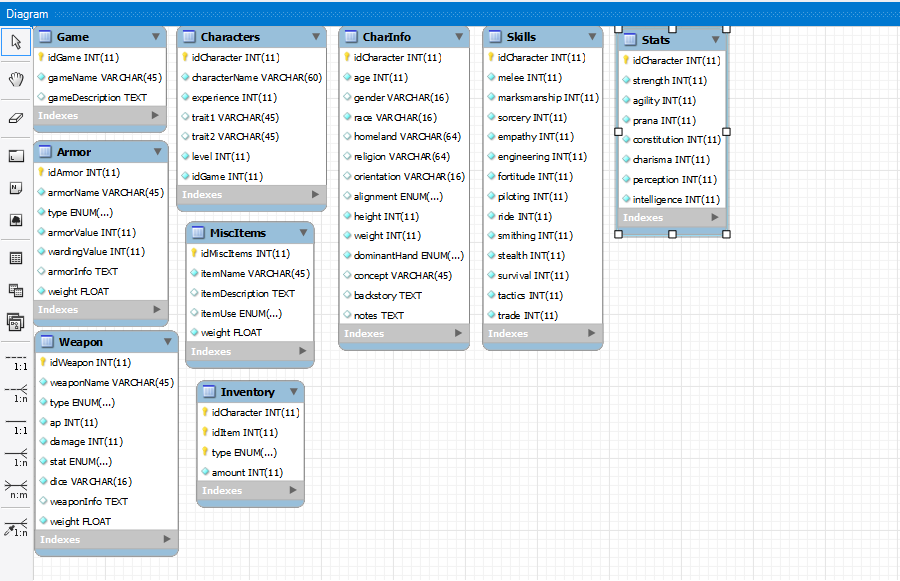
**Data Discovery and Modeling Phase**

As the game itself does not exist, this project was done entirely on data created by us. We filled all tables with at least 10 rows of information to ensure that our structures and data were modeled correctly.

**Database Modeling and Creation**

We used the following tables: Character; a table that holds an id number for a character, and some very basic information, like their experience and level. Character Information; a table that holds information about a character that is more detail-oriented and less relevant to gameplay. Stats; a table that holds the stats of a given characters. Skills; a table that holds the skills of a given character. Game; a table that holds a game id with character ids from characters within that game. Weapon; a table to hold weapons. Armor; a table to hold armor. Miscellaneous Items; a table that holds all items in the game that aren’t weapons or armor. Inventory; a linking table that pairs characters with items they possess.

This is our EER Diagram. You will notice that there are no foreign keys or connections between tables. The reason for this was mysterious MySQL errors that pervaded all attempts at creating them, whether in the schema, or with ALTER queries. This is why the idCharacter field is implemented so many times. It acts as a foreign key by joining tables on that field.



Here is the original schema used for the creation of the database.

-- -----------------------------------------------------

-- Schema solace

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `solace` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci ;

USE `solace` ;

-- -----------------------------------------------------

-- Table `solace`.`Game`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `solace`.`Game` ;

CREATE TABLE IF NOT EXISTS `solace`.`Game` (

`idGame` INT NOT NULL,

`gameName` VARCHAR(45) NOT NULL DEFAULT 'A Solace Game',

`gameDescription` TEXT NULL,

PRIMARY KEY (`idGame`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `solace`.`Character`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `solace`.`Characters` ;

CREATE TABLE IF NOT EXISTS `solace`.`Characters` (

`idCharacter` INT ZEROFILL NOT NULL AUTO\_INCREMENT,

`characterName` VARCHAR(60) NOT NULL,

`experience` INT NOT NULL,

`trait1` VARCHAR(45) NULL,

`trait2` VARCHAR(45) NULL,

`level` INT NOT NULL,

`idGame` INT NOT NULL,

PRIMARY KEY (`idCharacter`));

-- -----------------------------------------------------

-- Table `solace`.`CharInformation`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `solace`.`CharInfo` ;

CREATE TABLE IF NOT EXISTS `solace`.`CharInfo` (

`idCharacter` INT NOT NULL,

`age` INT NOT NULL DEFAULT 30,

`gender` VARCHAR(16) NULL,

`race` VARCHAR(16) NOT NULL,

`homeland` VARCHAR(64) NULL,

`religion` VARCHAR(64) NULL,

`orientation` VARCHAR(16) NULL,

`alignment` ENUM('LG','LN','LE','NG','N','NE','CG','CN','CE') NULL,

`height` INT NOT NULL DEFAULT 180,

`weight` INT NOT NULL DEFAULT 65,

`dominantHand` ENUM('right','left','ambidextrous') NULL,

`concept` VARCHAR(45) NULL,

`backstory` TEXT NULL,

`notes` TEXT NULL,

PRIMARY KEY (`idCharacter`));

-- -----------------------------------------------------

-- Table `solace`.`Skills`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `solace`.`Skills` ;

CREATE TABLE IF NOT EXISTS `solace`.`Skills` (

`idCharacter` INT NOT NULL,

`idSkills` INT NOT NULL,

`melee` INT NOT NULL DEFAULT 0,

`marksmanship` INT NOT NULL DEFAULT 0,

`sorcery` INT NOT NULL DEFAULT 0,

`empathy` INT NOT NULL DEFAULT 0,

`engineering` INT NOT NULL DEFAULT 0,

`fortitude` INT NOT NULL DEFAULT 0,

`piloting` INT NOT NULL DEFAULT 0,

`ride` INT NOT NULL DEFAULT 0,

`smithing` INT NOT NULL DEFAULT 0,

`stealth` INT NOT NULL DEFAULT 0,

`survival` INT NOT NULL DEFAULT 0,

`tactics` INT NOT NULL DEFAULT 0,

`trade` INT NOT NULL DEFAULT 0,

PRIMARY KEY (`idSkills`));

-- -----------------------------------------------------

-- Table `solace`.`Weapon`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `solace`.`Weapon` ;

CREATE TABLE IF NOT EXISTS `solace`.`Weapon` (

`idWeapon` INT NOT NULL,

`weaponName` VARCHAR(45) NOT NULL,

`type` ENUM('unarmed','melee','reach','ranged','other') NOT NULL DEFAULT 'unarmed',

`ap` INT NOT NULL DEFAULT 2,

`damage` INT NOT NULL DEFAULT 0,

`stat` ENUM('strength','agility','prana') NOT NULL DEFAULT 'strength',

`idCharacter` INT NOT NULL,

PRIMARY KEY (`idWeapon`));

-- -----------------------------------------------------

-- Table `solace`.`Armor`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `solace`.`Armor` ;

CREATE TABLE IF NOT EXISTS `solace`.`Armor` (

`idArmor` INT NOT NULL,

`armorName` VARCHAR(45) NOT NULL,

`type` ENUM('plate','mail','leather','cloth','other') NOT NULL DEFAULT 'cloth',

`armorValue` INT NOT NULL DEFAULT 0,

`wardingValue` INT NOT NULL DEFAULT 0,

`otherProperties` TEXT NULL,

`idCharacter` INT NOT NULL,

PRIMARY KEY (`idArmor`));

-- -----------------------------------------------------

-- Table `solace`.`MiscItems`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `solace`.`MiscItems` ;

CREATE TABLE IF NOT EXISTS `solace`.`MiscItems` (

`idMiscItems` INT NOT NULL,

`itemName` VARCHAR(45) NOT NULL,

`itemAmount` INT NOT NULL,

`itemDescription` TEXT NULL,

`idCharacter` INT NOT NULL,

PRIMARY KEY (`idMiscItems`));

-- -----------------------------------------------------

-- Table `solace`.`Stats`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `solace`.`Stats` ;

CREATE TABLE IF NOT EXISTS `solace`.`Stats` (

`idStats` INT NOT NULL,

`idCharacter` INT NOT NULL,

`strength` INT NOT NULL DEFAULT 3,

`agility` INT NOT NULL DEFAULT 3,

`prana` INT NOT NULL DEFAULT 3,

`constitution` INT NOT NULL DEFAULT 3,

`charisma` INT NOT NULL DEFAULT 3,

`perception` INT NOT NULL DEFAULT 3,

`intelligence` INT NOT NULL DEFAULT 3,

PRIMARY KEY (`idStats`));

-- -----------------------------------------------------

-- Table `solace`.`Inventory`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `solace`.`Inventory` ;

CREATE TABLE IF NOT EXISTS `solace`.`Inventory` (

`idCharacter` INT NOT NULL,

`idItem` INT NOT NULL,

`amount` INT NOT NULL DEFAULT 1,

`type` ENUM(`weapon`,`armor`,`misc`) NOT NULL DEFAULT `other`,

PRIMARY KEY (`idCharacter`,`idItem`));

Some things to note here; modifications have been made to the data structure as problems were discovered. Most notably, every field which has the idCharacter field now uses that as all or part of its primary key, and most other primary keys were dropped. There are exceptions to this, as the Weapon, Armor, and MiscItem tables had their idCharacter fields dropped in favor of the linking table Inventory.The Game table also uses a dual field primary key. Because the inventory table links a single table to three other tables, it uses a triple field primary key including the ‘type’ field, allowing a weapon, armor, and miscitem with the same ID to be linked with a single character.

There were several fields that were added as time went on. One of these was a weight for items. It may be necessary to compute the total weight of a player's inventory to determine if they can actually carry that weight. This is known as carrying capacity. There were other fields, mostly descriptive, which allowed for more information to be held.

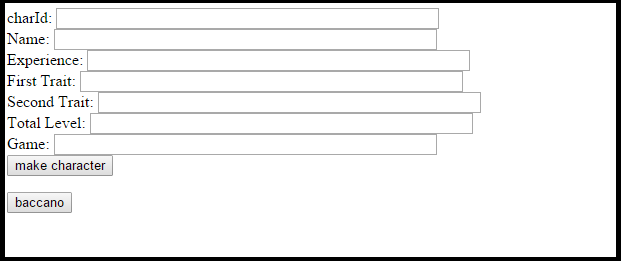
**Database Interface/Application**

Our databases primary purpose is to save enough datum about a game that it could be rebuilt when local game files are lost. Being more of a data backup, our interface and application is somewhat limited because the actual game this is supposed to be used for has not been coded yet.

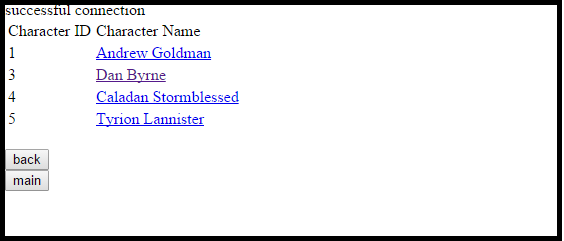
We do have some limited php and html that allow us to add information to the database as well as browsing characters that are currently in there. Here is a screenshot of our launch page, after logging in.



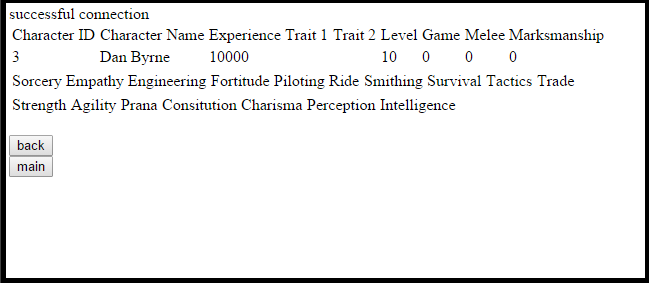
This is an example of an input page.



And here is an example of searching for a character, given the string “an”



Finally, if we click on Dan Byrne, we get this page.



This is as far as we made it with our online interface. As you can see, there are many fields which still do not contain data, as we were struggling to get php to fill those fields. However, the database side of things is very robust and effective at its job. We will easily be able to re-create the basic elements of a game on a new system or fresh application with the data given.