CSL 332 Course Project: Tower of Babel - Text based RPG

Babelonians:
Jyotesh Singh, 2011EE10554
Karan Goel, 2011EE50555
Tushar Gupta, 2011EE50564

Due date: April 20, 2014, 11:55pm IST

1 Project Description

Our project involves building a simple, non-graphical role-playing game. This was motivated by the variety of RPGs that the creators have frequently played, and by wondering whether we can successfully emulate at least a bare-bones RPG. Indeed, our team name "Babelonians" was inspired by our constant chatter (hence the Tower of Babel) and interest in fantasy/history/mythology (hence the Babylonians).

The complete system allows users to create password-enabled user accounts, after users are lead to their home screens. Users can choose to battle monsters, by purchasing a wide variety of weapons, armor and weapons from the in-game store, or they can choose to rest at the Inn, to recuperate their health points (hp) and energy. Our system keeps track of user statistics, including every single battle ever fought by a user, and this is made available for the user to access. As users play more, they achieve higher levels and get access to better weapons.

When a user registers, he is allowed to choose a race and is also asked to enter his unique username, email and password. After registration, users battle monsters which are chosen by our battle engine based on similar skill levels. Users have the option of attacking, defending against or fleeing from monsters. Every move that is ever made by any user is stored by us, for them to access at a later time. A user's energy depletes every time he fights a battle, which means he is forced to go to the Inn after 10 continuous battles. Our Inn provides users a variety of options to regain their health; in addition users can also buy a potion for hp regeneration at the weapon store. If a user beats a monster, he gains some gold and experience points (xp), that is dropped by the monster. Users level up when their xp crosses a certain point.

The entities & attributes that were designed by us are shown in Table 1, while the ER Diagram corresponding to this is shown in Figure 1.

2 Data Sources and Statistics

We are using Final Fantasy 6 data for the XP/HP per level
http://www.gamefaqs.com/ps/562865-final-fantasy-vi/faqs/39433
which is unchanged. The equipment data is from
http://shrines.rpgclassics.com/snes/ff6/
We cleaned up this data and tweaked the prices in order to normalize them according to attack/defence (using Wolfram Alpha for best fit curves) as we were not incorporating item drops or special abilities of equipment in our game variant. We also tweaked the attack/defence to make sure that player & monster abilities are well matched. Player data was taken from
http://mmnet.iis.sinica.edu.tw/dl/wowah/
and was structed as show in Table 2. We only extracted Avatar Id, Level & Race data from this table by writing a Python script to extract unique player details.
Since privacy meant player/guild names were replaced by numbers, we generated our own names from these dictionaries (http://home.hiwaay.net/ lkseitz/comics/herogen/herogen.cgi). To generate user email & password data, we generated csv data from http://www.mockaroo.com for 90000 odd players.

Entity	Attributes
users	playerid, username, password, email, create_time
player	playerid, username, racename, level, xp, gold, hp, energy, restid, rest_time, weaponid, armorid_b, armorid_h, armorid_s
moderators	moderatorid, modname, email, password, create_time
race	racename
armor	armorid, name, level, gold, defense, armortype
weapon	weaponid, name, level, gold, attack
potion	potionid, potion_name, level, gold, type, percent
level_data	level, hp, xp
rest	restid, restname, min_level, rest_seconds_for_max, gold
move	moveid, movename
inventory	playerid, equipmentid, equipmenttype
monster	monsterid, name, hp, xp, level, gold, attack, defense
battle_log	battleid, battle_time, playerid, monsterid, player_starthp, win, player_weapon, player_armorb, player_armorh, player_armors, dmg_rnd1_for, dmg_rnd2_for, dmg_rnd3_for, dmg_rnd4_for, dmg_rnd5_for, dmg_rnd1_against, dmg_rnd2_against, dmg_rnd3_against, dmg_rnd4_against, dmg_rnd5_against, player_rnd1_move, player_rnd2_move, player_rnd3_move, player_rnd4_move, player_rnd5_move, rnd1_potion, rnd2_potion, rnd3_potion, rnd4_potion, rnd5_potion

Table 1: List of Entities and Attributes

Attribute Name	Permitted Values
Query Time	Date between Jan 2006 and Jan 2009
Query Sequence	Integer >= 0
Avatar Id	Integer >= 0
Guild	An integer within [1,513]
Level	An integer within [1,80]
Race	Blood Elf, Orc, Tauren, Troll, Undead
Class	Death Knight, Druid, Hunter, Mage, Paladin, Priest, Rogue, Shaman, Warlock, Warrior
Zone	One of 229 zones in World of Warcraft worlds

Table 2: Structure of downloaded raw player data

Monster data was taken from
http://www.rpglegion.com/ff6/monsters/monstersa.htm
and was cleaned up by tweaking xp & gold earned (and changing names we didn't like!).

Finally, we wrote a Python script to simulate battles for the 90000 registered players (to populate the battle_log table), and essentially wrote player game histories. This script simulated our battle engine.

Other smaller tables such as the rest table, containing a handful of tuples were handwritten by us.

Table 3 shows the statistics of our data.

3 Functionality and Working

1. **User's View of the System**

   (a) Home:

   The home page contains 3 major options; login, signup & moderator login. A user can directly login on this page (which leads to the 'piazza' which is the player home page), or can click on either of the signup/moderator login links.

   On pressing login, a query to verify the username & password details gets fired to the database. We have utilized MD5 hashing in our database for passwords.

   (b) Signup:

   The signup page asks the player to enter account details - username, password & email address, as well as a race. On successfully signing up, the player is lead to the 'piazza' once again.

Table	No. of Tuples	Time to Load	Raw Dataset Size	Raw Dataset Size (Clean)
users	88195	3948.697 ms	2 GB	9.2 MB
player	88195	4473.113 ms	2 GB	5.5 MB
race	5	3.977 ms	38 bytes	38 bytes
armor	61	5.122 ms	2 KB	2 KB
inventory	352780	5137.343 ms	-	5.4 MB
weapon	67	5.602 ms	2 KB	2 KB
potion	20	7.663 ms	500 bytes	532 bytes
rest	5	4.104 ms	-	104 bytes
battle_log	158755171	1601025.655 ms	-	16.25 GB
monster	360	7.996 ms	13 KB	13 KB
move	7	6.710 ms	-	154 bytes
level_data	99	12.564 ms	1 KB	1 KB
moderators	1	-	-	-

Table 3: Data Statistics

On signing up, a query is fired to update the user & player tables with the player's details and default values for level, xp, hp, energy & weapons.

(c) Moderator:

The moderator signin page asks for authentication for the moderator (which is cross checked by a query like the player's signin), and then displays options for the moderator.

(d) Piazza:

The piazza displays the player's current status (name, level, hp, xp, energy, equipped weapon & armor names), and provides options to do/go to the following:

i. Signout
ii. Battle Home
iii. Inventory
iv. Inn
v. Shop
vi. Leaderboard
vii. Settings

(e) Battle Home:

At the Battle Home, the player can firslty choose to use an energy potion from his inventory (a drop down list), whereupon queries will retrieve the energy effect of the potion, delete it from the player's inventory and increase the player's energy in the player table.

Second, the player can go to the reports page to view his battle repots and the last option is the player can choose to battle. In this case, the player is given 2 options - either battle monsters better than his skill level, or those worse than his skill level. After this check, a query is generated to check that the player is not resting in the inn (restid in the player table), and also has sufficient energy to start another battle. (Note: Players must actively click 'Wake up' at the Inn to stop resting.

(f) Battle Start:

A query is first fired to select a monster and then queries to get monster details from table monster, player's stats and the attack/defense/heal potions available in the player's inventory are loaded.

The battle proceeds as follows; players can choose to select 'Attack', 'Parry' or 'Flee' whereupon the battle engine calculates damage done, and depending on fled or knockout (or not), it relates how much damage was done to which side, accounts for the effect of potions that are used in rounds, etc. Players are allowed upto a maximum of 5 rounds to play against a monster. It also keeps the player table, and the battle_log updated with damage/move/potion used stats (for the battle_log), as well as hp/xp/gold/energy for the player table.

Once the battle is over, the current battle report is generated.

(g) Battle Report:

This calculates xp, gold, etc. by pulling monster & battle details (from the battle stats stored in battle_log using the current battle's id). It updates these values and the player's energy.

The entire battle is packaged into 1 transaction and is committed at the end, and then the battle report for the current battle is shown in exactly the same fashion as on the player's Reports page.

(h) Reports:

This page first extracts player battle reports, 20 at a time in order of most recent first. The xp and gold gained are recalculated for each report instead of being stored in the battle_log table, and then the entire statistics are shown on the screen.

The gold gained, and xp gained are calculated according to the following formula:

Gain = monstergold * (max(totaldamagedone,monstertotalhp))/monstertotalhp

(i) Inventory:

In the inventory, players can view equipped weapons, armor & owned potions. The equipped inventory is queried from the player table where it is stored directly (with name information from the static tables). This is in addition to the ability to change the equipped weapons of the player (through a drop down list), to other weapons purchased by the player.

These other weapons are displayed by querying the inventory table by playerid, and the current equipment change is done by performing an update on the player table.

The other option that is available is to go back to the Piazza.

(j) Inn:

In the inn, players can sleep or wake depending on their current resting state. Sleeps can be chosen between one of 5 variants (Against a Wall, Chair, Attic, Bed, Suite) which increase energy and heal damage at different rates. When a player sleeps, his state is changed in the player table by a query and he explicitly needs to wake himself to battle.

(k) Shop:

Players can buy or sell weapons/armor/potions at the shop. Weapons that are available to the player to purchase (according to level and gold) are queried from the static tables, and once a weapon is purchased from a player, his gold is updated in the player table, while the new weapon is added to his inventory.

Players can also sell any number of weapons, except the currently equipped weapons (to prevent the player from becoming weaponless/armorless). This is done by performing deletes on the inventory table.

(l) Leaderboard:

The leaderboard displays the top 50 players in the entire game, based on xp. This is implemented as a simple query on the player table.

(m) Settings:

Players are allowed to delete their accounts, as well as change their passwords. The query for account deletion executes a trigger to delete the player information from the inventory & battle_log tables. The query for password change updates the users table with the new MD5 hash of the user's password.

2. **Special Functionality**

(a) Indexes:

Our battle_log table has over 150 million rows, so we indexed the table on playerid using a B-Tree index (the PostgreSQL website indicates that hash indexes are not recommended for use) for fast lookups on battles fought by a particular player.

In addition, we also indexed the timestamps (also using a B-Tree index), to allow us to display pages of entries (1-50, 51-100, etc.) at the browser end quickly.

(b) Constraints:

Aside from primary key constraints on every table, there are uniqueness constraints on users(username, email), player(username), race(racename), armor(name), weapon(name), potion(potion_name), rest(restname), move(movename), monster(name).

There is also a referential constraint on playerid in users & player.

(c) Sequences:

We have several sequences which keep track of the next playerid/battleid that is to be handed out when a new player joins or a new battle commences. Using

```
SELECT nextval('sequence_name');
```

at the PHP front-end, we extract a new unique value that is next in the sequence. This also ensures that no 2 players are assigned the same playerid due to concurrency issues as the sequence lock is first acquired and then the sequence is incremented in this statement.

(d) Views

We created a materialized view for equipment (equipmentid, equipmentname, effect, gold, level, type), to ease access of displaying available equipment in the shop, as well as inventory querying. This view includes data from weapon, armor & potion.

We also created views for displaying leaderboard data which takes too long to query instantaneously.

(e) Triggers

We implemented the following triggers to ensure player values of hp, energy are updated when he levels up, and also to ensure that when a player signs up, he is assigned default inventory in the inventory table.

```
CREATE FUNCTION update_xp_gold() RETURNS trigger as $update_xp_gold$
BEGIN
        UPDATE player SET hp = (SELECT hp from level_data where level = NEW.level),
        energy = 100;
        RETURN NULL;
END;
$update_xp_gold$ language plpgsql;

CREATE FUNCTION newplayerinventory() RETURNS trigger as $newplayerinventory$
BEGIN
        INSERT into inventory
        VALUES
        (NEW.playerid, 1, 'Weapon'),
        (NEW.playerid, 1, 'Bodywear'),
        (NEW.playerid, 1, 'Helmet'),
        (NEW.playerid, 1, 'Shield')
        RETURN NULL;
END;
$newplayerinventory$ language plpgsql;

CREATE TRIGGER newplayerinventory
AFTER INSERT on users
EXECUTE PROCEDURE update_inventory();

CREATE TRIGGER levelupdate
AFTER UPDATE OF level ON player
EXECUTE PROCEDURE update_xp_gold();
```

(f) Access Privileges:

We implemented 3 user modes: normal, moderator & superuser. For each of these 3 modes, we created the following access privileges:

i. Normal: This mode is only allowed to query our database using SELECT queries and any player side queries which change dynamic tables (Eg - battle_log, gold, xp, hp in player table, password in users etc.) and not static tables in the database.

ii. Moderator: Mods are allowed to delete players from the user/player tables and are also allowed to make insertions into the weapon, armor, potion, monster, level, race & rest tables.

iii. Superuser: The superuser has unlimited access privileges to the database.

3. **List of Queries**

(a) Login Page:

i. `SELECT password FROM users WHERE username = $1;`

(b) Moderator:

i. `SELECT password from moderators WHERE modname = $1;`

ii.
```
INSERT INTO monster
VALUES ($1, $2, $3, $4, $5, $6, $7, $8);
```

iii. `DELETE FROM users where username = $1;`

iv.
```
UPDATE monster set attack = $1, xp = $2, gold = $3, defense = $4
,hp = $5 where monsterid = $6;
```

(c) Piazza:

i. `select * from player where playerid = $1;`

ii. `select name from weapon where weaponid = $1;`

iii. `select name from armor where armorid = $1 and armortype = 'Bodywear';`

iv. `select name from armor where armorid = $1 and armortype = 'Helmet';`

v. `select name from armor where armorid = $1 and armortype = 'Shield';`

(d) Inventory:

i.
```
update player set armorid_b = $1, armorid_h = $2, armorid_s = $3
where playerid = $4;
```

ii.
```
select name, attack, weaponid from inventory, weapon
where playerid = $1 and equipmenttype='Weapon' and equipmentid=weaponid;
```

iii.
```
select potion_name, type, percent from inventory, potion
where playerid = $1 and equipmenttype='Potion' and equipmentid=potionid;
```

iv.
```
select name, defense, armorid from inventory, armor
where playerid = $1 and equipmenttype='Helmet' and equipmentid=armorid
and armortype = 'Helmet';
```

v.
```
select name, defense, armorid from inventory, armor
where playerid = $1 and equipmenttype='Shield' and equipmentid=armorid
and armortype = 'Shield';
```

vi.
```
select name, defense, armorid from inventory, armor
where playerid = $1 and equipmenttype='Bodywear'
and equipmentid=armorid and armortype = 'Bodywear';
```

(e) Inn:

i. `select level, player.gold, rest.gold, min_level from player, rest where playerid = $1 and rest.restid = $2;`

ii. `update player set restid = $1, rest_time = ".date('Y-m-d H:i:s')." where playerid = $2;`

iii. `select restid, rest_time, energy, player.hp as hp, level_data.hp as maxhp from player, level_data where player.level = level_data.level and playerid = $1;`

iv. `select rest_seconds_for_max, restname from rest where restid = $1;`

(f) Shop:

i. `select * from equipment where equipmenttype = $1;`

ii. `insert into inventory values ($1, $2, $3);`

iii. `delete from inventory where equipmentid = $1 and playerid = $2 and equipmenttype = $3;`

iv. `update player set gold = gold + $goldTransferred;`

(g) Leaderboard:

i. `select * from player order by xp desc limit 50;`

ii. This returns the top 100 players in terms of win % who have fought at least 10 battles.

```
create materialized view winpercent as
select username, avg(win::int::double precision) * 100 as percentagewins
from battle_log, users where battle_log.playerid = users.playerid
group by username having count(playerid) > 10
order by percentagewins desc limit 100;
```

iii. This returns the top 100 players in terms of average damage dealt per battle.

```
create materialized view maxdmg as
select username, avg(dmg_rnd1_for+dmg_rnd2_for+
dmg_rnd3_for+dmg_rnd4_for+dmg_rnd5_for) as avgdmg
from battle_log, users where battle_log.playerid = users.playerid
group by username order by avgdmg desc limit 100;
```

(h) Battle Home:

i. `select energy, percent from player, potion where playerid = $1 and potionid = $2;`

ii. `update player set energy = energy + $energyToIncrease where playerid = $1;`

iii. `delete from inventory where playerid = $1 and equipmentid = $2 and equipmenttype = 'Potion';`

iv. `select level, restid, rest_time, energy, hp, username from player where playerid = $1;`

v. `select hp from level_data where level = \$1;`

vi. `select potionid, potion_name, type, percent from potion, inventory where playerid = $1 and equipmenttype = 'Potion' and potionid=equipmentid and type = 'E';`

vii. `select rest_seconds_for_max from rest where restid = $1;`

(i) Battle Start:

i. Randomly selects a monster of the appropriate level; where random() is handled by the database at the back.

```
select * from monster where level = $1 limit 1
offset (select trunc(random()*(num-1)+1) from (select count(*) as num
from monster where level = $1) as temp);
```

ii. `select nextval('battleid_seq');`

iii.
```
begin;
insert into battle_log values ($1, $2, $3, $4,
NULL, $5, $6, $7, $8,
NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL);
```

iv. `delete from inventory where playerid = $1 and equipmentid = $2 and equipmenttype = 'Potion';`

v. `update battle_log set rnd$1_potion = $2 where battleid = $3;`

vi.
```
update battle_log set win = $1, dmg_rnd<Round no.>_for = $2,
dmg_rnd<Round no.>_against = $3,
player_rnd<Round no.>_move = $4 where battleid = $5;
```

vii.
```
select (dmg_rnd1_for+dmg_rnd2_for+dmg_rnd3_for+dmg_rnd4_for+dmg_rnd5_for)
as dmgFor,
(dmg_rnd1_against+dmg_rnd2_against+dmg_rnd3_against
+dmg_rnd4_against+dmg_rnd5_against) as dmgAgainst from battle_log
where battleid = $1;
```

viii. `select potionid, potion_name, type, percent from potion, inventory where playerid = $1 and equipmenttype = 'Potion' and potionid=equipmentid and type <> 'E';`

(j) Battle Result & Reports:

i.
```
select * from battle_log where playerId = $1
order by timestamp desc limit $2 offset $3;
```

ii. `select name, xp, gold, hp from monster where monsterid = $1;`

iii. `select name from weapon where weaponid = $1;`

iv. `select name from armor where armorid = $1 and armortype = $2;`

v. `select potion_name, type, percent from potion where potionid = $1;`

vi. `select * from battle_log where battleid = $1;`

vii. **update** player **set** xp = xp + $xpGained, gold = gold + $goldGained **where** playerid = $1;

viii. **select** player.xp, level_data.xp, player.energy **from** player, level_data **where** playerid =$1 **and** level_data.**level** = player.**level**+1;

ix. **update** player **set** **level** = **level** + 1 **where** playerid = $1;

x. **update** player **set** energy = energy − 10 **where** playerid = $1;

xi. Used to end the battle transaction once a battle is over and has not been interrupted (logout/crash).

commit;

4. **Query Times**

Table 4 is a list of query times (averaged over 3 runs) of queries from Section 2.

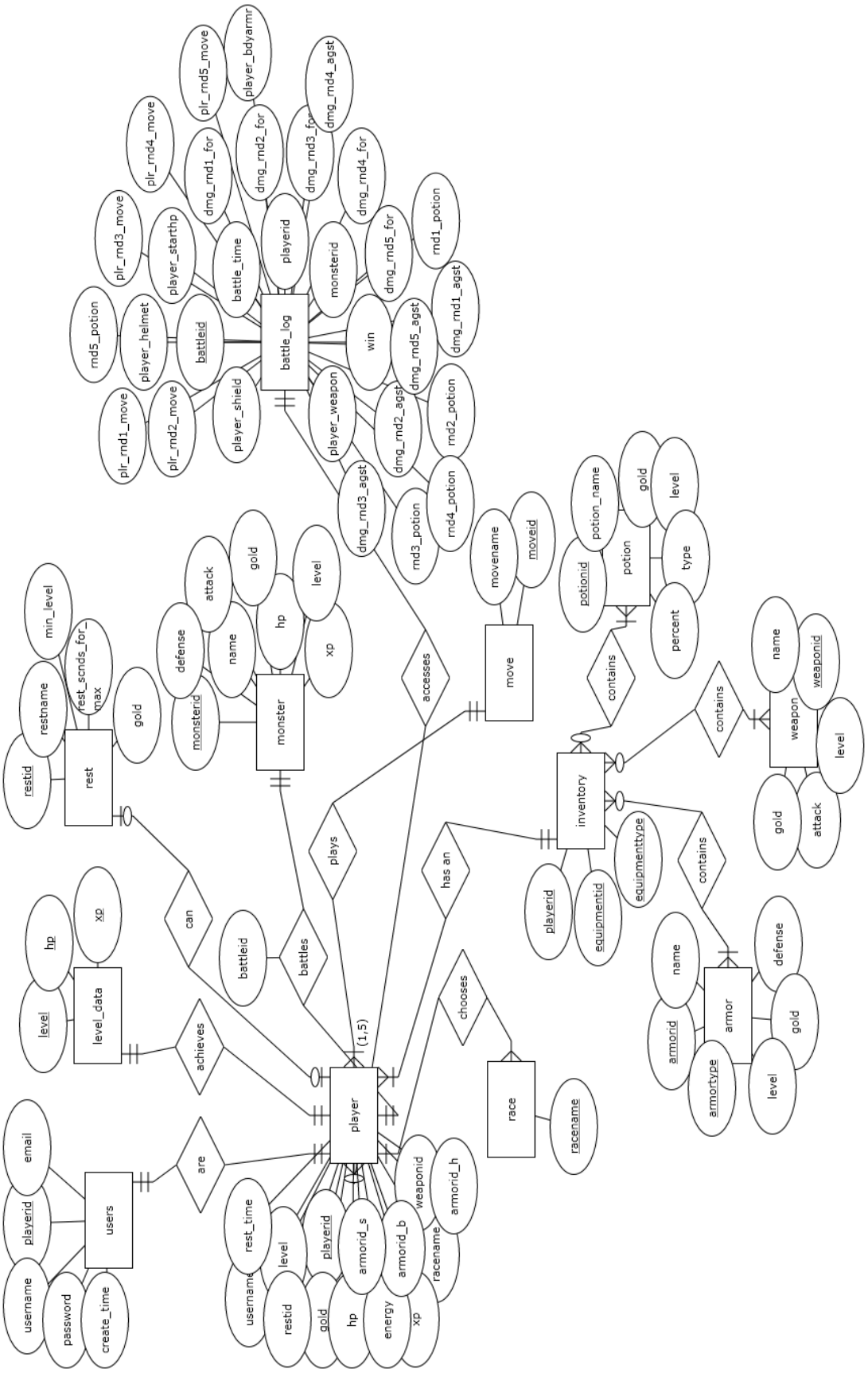

Figure 1: ER Diagram

Query Number:	Average Running Time
(a) i	2.775ms
(b) i	2.973ms
(b) ii	8.015ms
(b) iii	0.372ms
(b) iv	4.122ms
(c) i	0.619ms
(c) ii	3.283ms
(c) iii	3.780ms
(c) iv	3.819ms
(c) v	3.698ms
(d) i	2.971ms
(d) ii	3.886ms
(d) iii	3.091ms
(d) iv	3.556ms
(d) v	3.982ms
(d) vi	3.586ms
(e) i	3.065ms
(e) ii	5.130ms
(e) iii	5.159ms
(e) iv	0.372ms
(f) i	3.328ms
(f) ii	5.027ms
(f) iii	3.605ms
(f) iv	3.443ms
(g) i	101.894ms
(g) ii	141425.233ms
(g) iii	1231903.074ms
(h) i	9.305ms
(h) ii	0.594ms
(h) iii	3.912ms
(h) iv	0.572ms
(h) v	3.885
(h) vi	3.416ms
(h) vii	3.430ms
(i) i	13.529ms
(i) ii	3.262ms
(i) iii	8.822ms
(i) iv	5.351ms
(i) v	0.818ms
(i) vi	0.901ms
(i) vii	2.628ms
(i) viii	2.731ms
(j) i	23.5ms
(j) ii	2.623ms
(j) iii	3.559ms
(j) iv	3.143ms
(j) v	2.243ms
(j) vi	0.559ms
(j) vii	5.920ms
(j) viii	3.823ms
(j) ix	2.461ms
(j) x	4.621ms

Table 4: Average Query Running Times