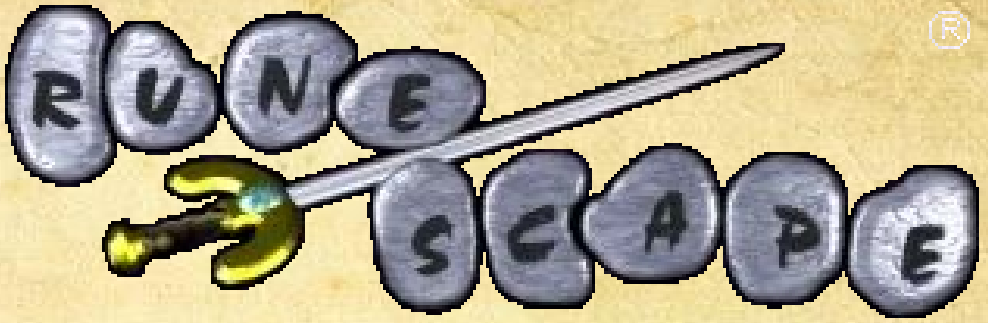

Runescape Database

By: Jason Hipolito

Table Of Contents

Executive Summary

Runescape is a MMORPG(massive multiplayer online role playing game) where adventurers go out and explore the world of Gielinor doing quests, training skills and slaying monsters.

Jagex, the creators of Runescape has contacted me to create a database to keep track of player progress. They want to be able to track character information like skill levels, quests completed/not completed, inventories, banks, and items received through monster drops.

ER Diagram

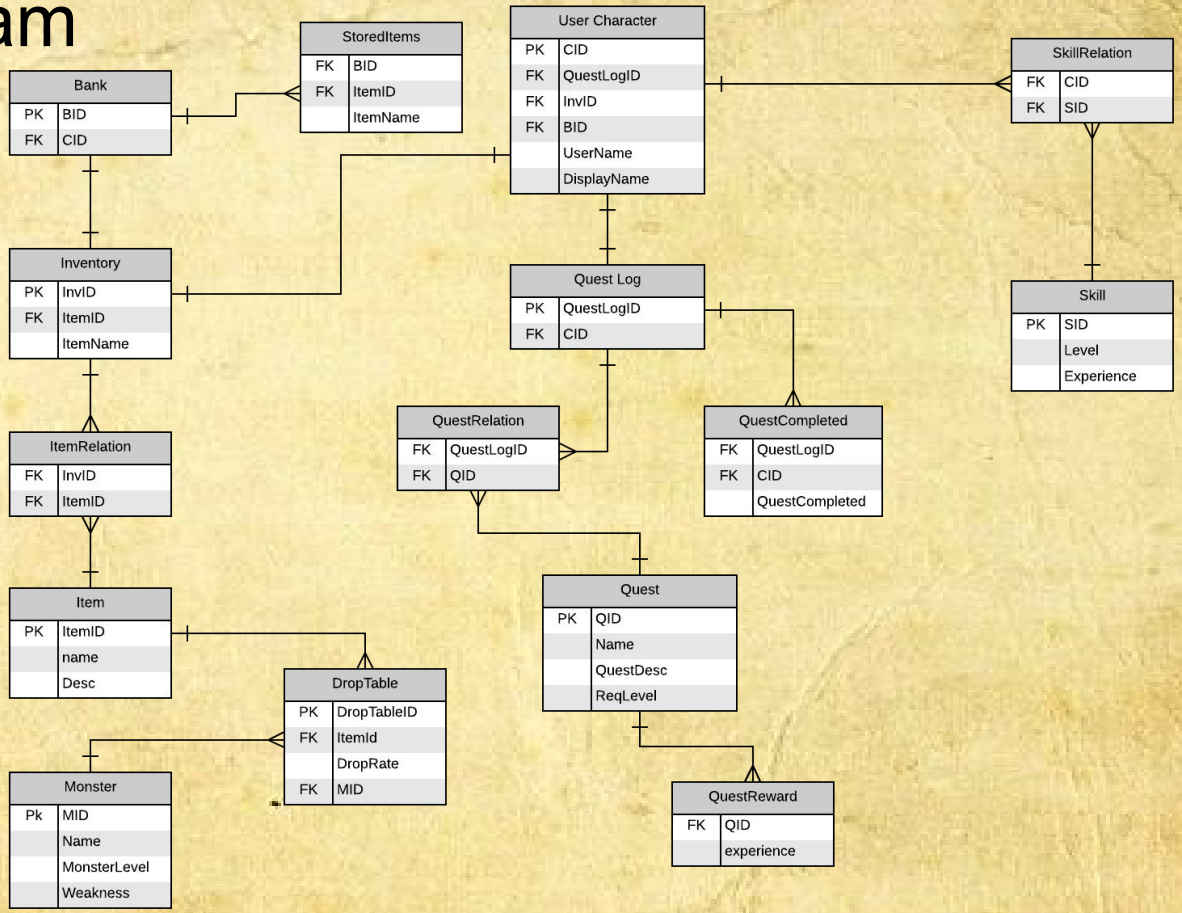
Bank
PK BID
FK CID
StoredItems
FK BID
FK ItemID
ItemName
User Character
PK CID
FK QuestLogID
FK InvID
FK BID
UserName
DisplayName
SkillRelation
FK CID
FK SID
Inventory
PK InvID
FK ItemID
ItemName
Quest Log
PK QuestLogID
FK CID
Skill
PK SID
Level
Experience
ItemRelation
FK InvID
FK ItemID
QuestRelation
FK QuestLogID
FK QID
QuestCompleted
FK QuestLogID
FK CID
QuestCompleted
Item
PK ItemID
name
Desc
Quest
PK QID
Name
QuestDesc
ReqLevel
DropTable
PK DropTableID
FK ItemId
DropRate
FK MID
Monster
Pk MID
Name
MonsterLevel
Weakness
QuestReward
FK QID
experience

Table - Bank

This table combines a bank ID with a character ID.

Create Statement:

```
CREATE TABLE Bank (
    BID serial UNIQUE NOT NULL,
    CID integer references Item(ItemID) NOT NULL,
PRIMARY KEY (BID)
) ;
```

Functional Dependencies:

BID → CID

bid integer	cid integer
1	1
2	2
3	3
4	4
5	5
6	6

Table - Stored Items

Shows what items are stored in what bank. There can be duplicated bank ID's and items because a player can own multiple of the same items.

Create Statement:

```
CREATE TABLE StoredItems (
    BID integer references Bank(BID) NOT NULL,
    ItemID integer references Item(ItemID) NOT NULL,
    ItemName text
);
```

Functional Dependencies:

BID , ItemID → itemName

bid integer	itemid integer	itemname text
1	1	Spade
1	4	Bandos Chestplate
1	1	Spade
2	3	Rune Scimitar
2	4	Bandos Chestplate
4	3	Rune Scimitar
3	5	Dragon Boots
3	4	Bandos Chestplate
5	11	Arclight
6	3	Rune Scimitar
6	4	Bandos Chestplate
4	5	Dragon Boots
2	3	Rune Scimitar
3	4	Bandos Chestplate
3	9	Ahrim Staff

Table - Item

This table stores items that are in the game.

Create Statement:

```
CREATE TABLE Item (
    ItemID serial UNIQUE NOT NULL,
    name text,
    itemDesc text,
PRIMARY KEY (ItemID)
);
```

Functional Dependencies:

itemID → name, itemDesc

itemid integer	name text	itemdesc text
1	Spade	Used to dig
2	Pickaxe	Used to mine
3	Rune Scimitar	Slash Weapon
4	Bandos Chestplate	Defensive chestplate
5	Dragon Boots	Strong Boots
6	Dragon Dagger	Special Weapon
7	Abyssal Whip	an Abyssal Demon Spine
8	legends Cape	Cape for legends
9	Ahrim Staff	Anchient Staff
10	ashes	its ash
11	arclight	a Strong weapon agianst demons
12	Bones	Bury these
13	Big Bones	Bury these
14	Death Rune	Magical Runes
15	Chaos Rune	Magical Runes
16	Air Rune	Magical Runes
17	Fire Rune	Magical Runes

Table - Inventory

This is the player inventory where they can hold items that are picked up.

Create Statement:

```
CREATE TABLE Inventory (
    InvID serial UNIQUE NOT NULL,
    ItemID integer references Item(ItemID) NOT NULL,
    ItemName text,
PRIMARY KEY (InvID)
);
```

Functional Dependencies:

InvID → itemID, itemName

invid integer	itemid integer	itemname text
1	1	Spade
2	15	Chaos Rune
3	15	Chaos Rune
4	15	Chaos Rune
5	9	Ahrim Staff
6	13	Big Bones
7	4	Bandos Chestplate
8	2	pickaxe

Table - ItemRelation

This creates a many to many relationship between the inventory table and item table for a many to many relationship.

Create Statement:

```
CREATE TABLE ItemRelation (
    InvID integer references Inventory(InvID) NOT NULL,
    ItemID integer references Item(ItemID) NOT NULL
);
```

Functional Dependencies:

InvID → itemID

invid integer	itemid integer
1	1
1	15
2	15
3	15
4	15
5	9
6	13
7	4
8	2

Table - DropTable

This table shows the drop rate of items dropped from monsters as a percent where 100 is 100% and .25 is 0.25% therefore being very rare.

Create Statement:

```
Create TABLE DropTable (
    DropTableID serial UNIQUE NOT NULL,
    ItemID integer references Item(ItemID) NOT NULL,
    MID integer references monster(MID) NOT NULL,
    DropRate decimal(5,2)
);
```

Functional Dependencies:

DropTableID → itemID, MID, dropRate

droptableid integer	itemid integer	mid integer	droprate numeric (5,2)
1	1	1	40.00
2	3	1	20.00
3	4	1	20.00
4	15	1	60.00
5	7	2	0.25
6	13	1	100.00
7	14	2	30.00
8	10	2	100.00
9	15	2	30.00
10	16	2	30.00
11	12	4	100.00
12	2	4	2.50
13	17	4	25.00
14	16	4	25.00
15	14	5	45.00

Table - Monster

Shows the monster's information like name level and their weakness.

Create Statement:

```
Create TABLE Monster (
    MID serial UNIQUE NOT NULL,
    Name text,
    MonsterLevel integer,
    Weakness text,
    Primary Key(MID)
);
```

Functional Dependencies:

MID → name, monsterLevel, weakness

mid integer	name text	monsterlevel integer	weakness text
1	General Graardor	642	Stab
2	Abyssal Demon	124	ArcLight
4	Goblin	5	Slash
5	Spiritual mage	83	Ranged
6	Ahrim the Blighted	98	Ranged

Table - Skill

This is the player's skill level(what level they are).

Create Statement:

```
CREATE TABLE Skill (
    SID serial UNIQUE NOT NULL,
    Level integer,
    Experience integer,
    Primary Key (SID)
);
```

Functional Dependencies:

SID → level, experience

sid integer	level integer	experience integer
1	12	1584
2	96	9684577
3	55	166636
4	99	13034431
5	88	4385776
6	65	449428

Table - QuestLog

Keeps track of which quest log belongs to which user.

Create Statement:

```
CREATE TABLE QuestLog (
    QuestLogID serial UNIQUE NOT NULL,
    CID integer references Item(ItemID) NOT NULL,
    Primary Key (QuestLogID)
);
```

Functional Dependencies:

QuestLogID → CID

questlogid integer	cid integer
2	2
1	1
3	3
4	4
5	5
6	6

Table - Quest Completed

This table shows what quest each player has already completed.

Create Statement:

```
Create TABLE QuestCompleted (
    QuestlogID integer references Questlog(QuestlogID ) NOT NULL,
    CID integer references UserCharacter(CID) NOT NULL,
    Questcompleted text
);
```

Functional Dependencies:

QuestLogID → CID, QuestCompleted

questlogid integer	cid integer	questcompleted text
2	2	Dragon Slayer
2	2	Vampire Slayer
2	2	Cooks Assisstant
2	2	Loost Tribe
3	3	Vampire Slayer
3	3	Lost Tribe
5	5	Another Slice of H.A.M.
4	4	Another Slice of H.A.M.
5	5	Lost Tribe
6	6	Another Slice of H.A.M.
6	6	Vampire Slayer
6	6	Lost Tribe
6	6	Cooks Assisstant
6	6	Dragon Slayer
6	6	Fight Arena

Table - Quest

Create Statement:

```
CREATE TABLE Quest (
    QID serial UNIQUE NOT NULL,
    Name text,
    QuestDesc text,
    ReqLevel integer,
PRIMARY KEY (QID)
);
```

This is a list of available quests that can be completed and their required level.

Functional Dependencies:

QID → name, QuestDesc, ReqLevel

qid integer	name text	questdesc text	reqlevel integer
5	Lost Tribe	there is a disturbance in lumbride castle basment, go investigate	55
4	Vampire Slayer	Kill the vampire harassing Draynor	20
1	Dragon Slayer	Venture the the is of cranor and slay a dragon	32
2	Cooks Assisstant	The cook of lumbridge needs an adventurer to get ingredients	10
3	Fight Arena	Save a mothers children from the fight arena	45
6	Another Slice of H.A.M.	H.A.M kidnaped my brother can you rescue him?	65

Table - Quest Relation

Allows a many to many connection with Quest log and Quest table.

Create Statement:

```
CREATE TABLE QuestRelation (
    QuestLogID integer references questLog(QuestLogID) NOT NULL,
    QID integer references Quest(QID) NOT NULL
);
```

Functional Dependencies:

QuestLogID → QID

questlogid integer	qid integer
1	1
2	1
3	2
4	2
5	4
6	5

Table - Quest Reward

A table with experience that is awarded to the player after completing the quest.

Create Statement:

```
CREATE TABLE QuestReward (
    QID integer references Quest(QID) NOT NULL,
    Experience integer
);
```

Functional Dependencies:

QID → Experience

qid integer	experience integer
2	150
3	3000
4	400
1	30000
6	50000
5	23000

Table - UserCharacter

Create Statement:

```
CREATE TABLE UserCharacter (
    CID serial UNIQUE NOT NULL,
        QuestLogID integer references QuestLog(QuestLogID) NOT NULL,
    InvID integer references Inventory(InvID) NOT NULL,
    BID integer references Bank(BID) NOT NULL,
    UserName VarChar(12),
    DisplayName VarChar(12),
    Primary Key(CID)
);
```

Functional Dependencies:

CID → questLogID, InvID,BID,
UserName, DisplayName

This is a table with information about that character.

cid integer	questlogid integer	invid integer	bid integer	username character varying (12)	displayname character varying (12)
1	1	1	1	Jay	Dark Jay53
2	2	2	2	FishSlayer90	CaptnFish
3	3	3	3	DragonBeast	PonyBoy
4	4	4	4	L33tNoob	Getgud
5	5	5	5	LordAlan	Inferno
6	6	6	6	kitty4life	DogsrCool

Table - Skill Relation

Connects the user and skill table together.

Create Statement:

```
CREATE TABLE SkillRelation (
    CID integer references UserCharacter(CID) NOT NULL,
        SID integer references Skill(SID) NOT NULL
);
```

CID → SID

cid integer	sid integer
1	1
2	2
3	3
4	4
5	5
6	6

View - User Information

Shows information about a user and what they are currently holding in a table that is easy to read at a quick glance.

```
CREATE OR REPLACE VIEW UserInfo AS
SELECT  usercharacter.username,
            usercharacter.displayname,
            skill.level,
            skill.experience,
            inventory.itemName
FROM  usercharacter INNER JOIN skillrelation ON usercharacter.CID = skillrelation.CID
                                LEFT JOIN Skill ON skillrelation.SID = Skill.SID
                                LEFT JOIN inventory ON usercharacter.invid = inventory.invid;
```

View - User Information

username character varying (12)	displayname character varying (12)	level integer	experience integer	itemname text
Jay	Dark Jay53	12	1584	Spade
FishSlayer90	CaptnFish	96	9684577	Chaos Rune
DragonBeast	PonyBoy	55	166636	Chaos Rune
L33tNoob	Getgud	99	13034431	Chaos Rune
LordAlan	Inferno	88	4385776	Ahrim Staff
kitty4life	DogsrCool	65	449428	Big Bones

View - Monster Info & drops

It's important to keep track of what monsters drop and how often they drop these items so that very strong items don't come into the game to quickly/often as this can mess up the economy around that item. So this view will tell you what items are dropped by what monster and how often it is dropped. The same monster comes up multiple times because some monsters drop more than one item. You will also see that some items like bones and ashes are a 100% drop rate because when a monster/demon dies they will always drop that plus another item.

```
CREATE OR REPLACE VIEW MonsterInfoDropRate AS
SELECT monster.name,
            monster.monsterLevel,
            monster.weakness,
            item.name,
            dropTable.droprate
FROM    Monster LEFT JOIN DropTable ON monster.MID = DropTable.MID
                        LEFT JOIN item ON DropTable.itemID = item.itemID;
```

View - Monster Info & drops

monstername text	monsterlevel integer	weakness text	itemname text	droprate numeric (5,2)
General Graardor	642	Stab	Spade	40.00
General Graardor	642	Stab	Rune Scimitar	20.00
General Graardor	642	Stab	Bandos Chestplate	20.00
General Graardor	642	Stab	Chaos Rune	60.00
Abyssal Demon	124	ArcLight	Abyssal Whip	0.25
General Graardor	642	Stab	Big Bones	100.00
Abyssal Demon	124	ArcLight	Death Rune	30.00
Abyssal Demon	124	ArcLight	ashes	100.00
Abyssal Demon	124	ArcLight	Chaos Rune	30.00
Abyssal Demon	124	ArcLight	Air Rune	30.00
Goblin	5	Slash	Bones	100.00
Goblin	5	Slash	Pickaxe	2.50
Goblin	5	Slash	Fire Rune	25.00
Goblin	5	Slash	Air Rune	25.00
Spiritual mage	83	Ranged	Death Rune	45.00
Spiritual mage	83	Ranged	Pickaxe	1.00
Spiritual mage	83	Ranged	arclight	3.00
Spiritual mage	83	Ranged	Bones	100.00
Ahrim the Blighted	98	Ranged	Ahrim Staff	2.00
Ahrim the Blighted	98	Ranged	ashes	100.00
Ahrim the Blighted	98	Ranged	Rune Scimitar	15.00
Ahrim the Blighted	98	Ranged	Dragon Dagger	10.00

View - Users Bank

Sometimes it's important to see what items are in the game for the sake of having data. For example if a new monster where to be released and the devs see that there are already a lot of very powerful items from that monster in the game very early on they will know for future updates to lower the drop rate or nerf the drop rate of the current monster.

```
CREATE OR REPLACE VIEW PlayerBank AS
SELECT    usercharacter.Displayname AS Display_Name,
          StoredItems.ItemName AS Item,
          StoredItems.itemID
FROM      StoredItems LEFT JOIN Bank ON StoredItems.BID = bank.bid
                      LEFT JOIN Inventory ON Bank.CID = Inventory.invID
                      LEFT JOIN Usercharacter ON Inventory.invID = Usercharacter.invid
ORDER BY StoredItems.ItemName;
```

View - Users Bank

display_name character varying (12)	item text	itemid integer
PonyBoy	Ahrim Staff	9
CaptnFish	Ahrim Staff	9
DogsrCool	Ahrim Staff	9
Inferno	Arclight	11
Getgud	Arclight	11
Inferno	Arclight	11
DogsrCool	Bandos Chestplate	4
Dark Jay53	Bandos Chestplate	4
CaptnFish	Bandos Chestplate	4
PonyBoy	Bandos Chestplate	4
PonyBoy	Bandos Chestplate	4
PonyBoy	Bandos Chestplate	4
DogsrCool	Bandos Chestplate	4
PonyBoy	Dragon Boots	5
Getgud	Dragon Boots	5
CaptnFish	Rune Scimitar	3
Getgud	Rune Scimitar	3
CaptnFish	Rune Scimitar	3
DogsrCool	Rune Scimitar	3
Dark Jay53	Spade	1
Dark Jay53	Spade	1

Reports - Average Level & Items Stored

Shows The average level of all player so the devs know what Level content they should most likely be working on for future updates.

```
SELECT Trunc (Avg(skill.level),1) AS Average_Player_Level
FROM SKill;
```

average_player_level numeric
69.16

Shows how many of each items are stored.

```
SELECT itemname, count(storeditems.itemname)
FROM Storeditems
GROUP BY itemname
ORDER BY count(storeditems.itemname);
```

itemname text	count bigint
Dragon Boots	2
Spade	2
Ahrim Staff	3
Arclight	3
Rune Scimitar	4
Bandos Chestplate	7

Stored Procedures - User Search

```
create or replace function SeachUserName(text, REFCURSOR)
returns refcursor as $$
declare
  SearchUser text      := $1;
  resultset   REFCURSOR := $2;
begin
  open resultset for
    select *
    from   usercharacter
    Where  userName like SearchUser;
  return resultset;
end;
$$
language plpgsql;
```

```
SELECT SeachUserName('L%', 'results');
FETCH ALL FROM results;
```

cid integer	questlogid integer	invid integer	bid integer	username character varying (12)	displayname character varying (12)
4	4	4	4	L33tNoob	Getgud
5	5	5	5	LordAlan	Inferno

Stored Procedures - Monster Search

```
create or replace function MonsterSearch(int, int, REFCURSOR)
returns refcursor as $$
declare
  MonsterMinLevel  int      := $1;
  MonsterMaxLevel  int      := $2;
  resultset       REFCURSOR := $3;
begin
  open resultset for
    select *
    from   Monster
    Where  MonsterLevel > MonsterMinLevel
    AND    MonsterLevel < MonsterMaxLevel;
  return resultset;
end;
$$
language plpgsql;
```

```
SELECT MonsterSearch(80, 100, 'results');
FETCH ALL FROM results;
```

mid integer	name text	monsterlevel integer	weakness text
5	Spiritual mage	83	Ranged
6	Ahrim the Blighted	98	Ranged

Triggers

```
CREATE OR REPLACE FUNCTION New_Username() RETURNS TRIGGER AS
$$
DECLARE
 Username_ text := cast(new.username as text);
BEGIN
 IF Username_ is NULL THEN
RAISE EXCEPTION 'Enter a Username';
 END IF;
 IF (SELECT length(Username_)
 FROM Usercharacter
 WHERE username = new.Username) < 13 THEN
 RAISE EXCEPTION 'Username must be 12 or less characters';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Triggers

```
CREATE TRIGGER Username_check
 AFTER INSERT
 ON Usercharacter
 FOR EACH ROW
 EXECUTE PROCEDURE New_Username();
```

Security

Admin has power to edit all information in the DB

```
create role admin;
grant all on all tables in schema public to admin;
```

Monster_Developer gets access to the monster table droptable and item table because he need to be able to create the monster and edit what types of items they are droping/creating new items and making sure they will go into the bank.

```
create role Monster_Developer;
GRANT SELECT, INSERT, DELETE
ON Monster, DropTable, Item, ItemRelation, Inventory, Bank, StoredItems
TO Monster_Developer;
```

Security

Quest_Developer writes and designs epic adventures for the players of runescape to do so he needs access to all parts of the quest tables.

```
create role Quest_Developer;
GRANT SELECT, INSERT, DELETE
ON Quest, QuestReward, QuestRelation, QuestLog, QuestCompleted
TO Quest_Developer;
```

Bot_Moderator. Sadly in runescape there is a big problem with cheaters and gold farmers running bots to get an unfair advantage in the game. Therefore we need a moderator to check accounts for suspicious activity and review the account for any signs of cheating or "Botting".

```
create role Bot_Moderator;
GRANT SELECT, INSERT, DELETE
ON Usercharacter, questlog, questcompleted, skillrelation, skill, inventory, bank, storeditems
TO Bot_Moderator;
```

Implementation Notes & Known Problems

- On Usercharacter table you need to add the cid manually because i first entered it manually and now the Serial command on the create statements don't work properly.

- I had some issues with getting the Triggers working properly.

Future Enhancements

- Add a Members table to show who is a member and who isn't.
- Add a table that shows the items a player has equipped.
- Create triggers for display name to make sure it's not inappropriate.
- Add a bank pin to the bank and recovery questions if needed by the player. This will ensure that if someone hacks their account, they won't be able to access their items.