МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСИС»

Институт информационных технологий и компьютерных наук

Кафедра инженерной кибернетики

Проектная работа

по дисциплине

«Базы данных»

на тему

«База данных для игрового мира»

Выполнил:
студент 1-го курса,
гр. БПМ-22 3- Киселев К.А.

Проверил:
Новицкий Д.Д.

Москва, 2023

Содержание

Введение

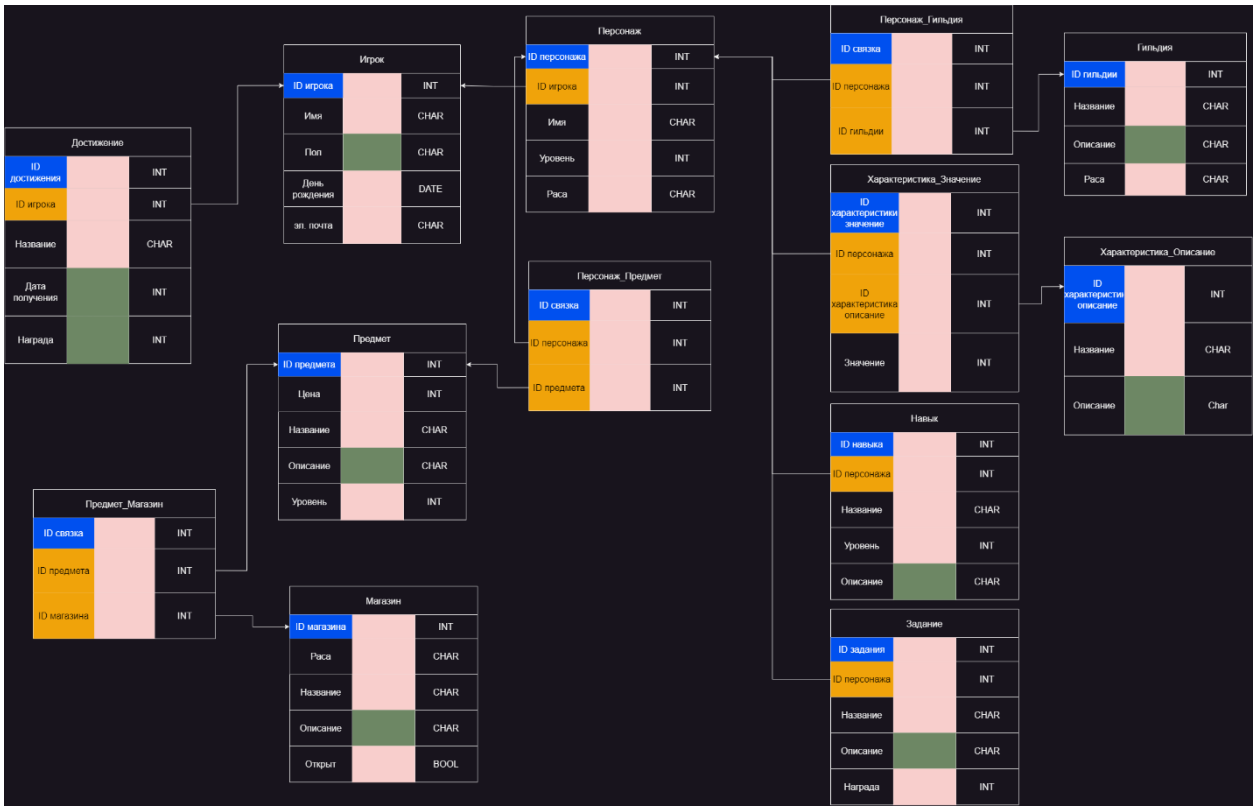
База данных представляет собой игровую систему, которая включает в себя различные элементы, такие как игроки, достижения, герои, гильдии, характеристики и навыки, задания, предметы и магазины.

Структура базы данных включает следующие таблицы:

1) Player - хранит информацию об игроках, включая их ID, имя, пол, дату рождения и электронную почту.
2) Achievement - отслеживает достижения игроков, связанные с их ID и предоставляет дополнительные детали, такие как название, дата получения и количество очков.
3) Hero - хранит информацию о героях игроков, включая их ID, имя, уровень, расу и внешний ключ для связи с игроком.
4) Guild - хранит информацию о гильдиях, включая их ID, название, описание и расу.
5) Hero_Guild - связывает героев с гильдиями, используя внешние ключи для связи с героями и гильдиями.
6) Characteristic_Description и Characteristic_Value - хранят информацию о характеристиках героев, включая их ID, название и значение.
7) Skill - хранит информацию о навыках героев, включая их ID, имя, уровень и описание.
8) Task - хранит информацию о задачах героев, включая их ID, имя, описание и награду.
9) Item - хранит информацию о предметах, включая их ID, стоимость, имя, описание и уровень.
10) Hero_Item - связывает героев с предметами, используя внешние ключи для связи с героями и предметами.
11) Shop - хранит информацию о магазинах, включая их ID, расу, имя, описание и статус открытия.
12) Item_Shop - связывает предметы с магазинами, используя внешние ключи для связи с предметами и магазинами.

Все эти таблицы связаны между собой через внешние ключи, что позволяет управлять и анализировать данные на более высоком уровне. Например, можно отслеживать, какой игрок получил какое достижение, или какой герой принадлежит к какой гильдии.

Даталогическая модель:



1. Создание и заполнение таблиц

```
SET foreign_key_checks = 0;
DROP TABLE IF EXISTS player;
DROP TABLE IF EXISTS achievement;
DROP TABLE IF EXISTS hero;
DROP TABLE IF EXISTS guild;
DROP TABLE IF EXISTS hero_guild;
DROP TABLE IF EXISTS characteristic_description;
DROP TABLE IF EXISTS characteristic_value;
DROP TABLE IF EXISTS skill;
DROP TABLE IF EXISTS task;
DROP TABLE IF EXISTS item;
DROP TABLE IF EXISTS hero_item;
DROP TABLE IF EXISTS shop;
DROP TABLE IF EXISTS item_shop;

SET foreign_key_checks = 1;

create table player(
    id_player INT NOT NULL PRIMARY KEY auto_increment,
    name VARCHAR(100) not null,
    gender VARCHAR(100) null,
    birthday DATE not null,
    email varchar(100) not null
);
```

```sql
create table achievement(
    id_achievement INT NOT NULL PRIMARY KEY auto_increment,
    id_player INT not null,
    nname VARCHAR(100) null,
    date_receipt DATE not null,
    points INT not null,
    foreign key (id_player) references player(id_player)
    on delete cascade
    on update cascade
);

create table hero(
    id_hero INT NOT NULL PRIMARY KEY auto_increment,
    id_player INT not null,
    nname VARCHAR(100) null,
    llevel INT not null,
    race VARCHAR(100) not null,
    foreign key (id_player) references player(id_player)
    on delete cascade
    on update cascade
);

create table guild(
    id_guild INT NOT NULL PRIMARY KEY auto_increment,
    nname varchar(100),
    ddescription TEXT null,
    race VARCHAR(100) not null
);

create table hero_guild(
    id_bundle INT NOT NULL PRIMARY KEY auto_increment,
    id_hero INT not null,
    id_guild INT not null,
    foreign key (id_hero) references hero(id_hero)
    on delete cascade
    on update cascade,
    foreign key (id_guild) references guild(id_guild)
    on delete cascade
    on update cascade
);

create table characteristic_description(
    id_characteristic_description int not null primary key auto_increment,
    nname varchar(100) not null,
    ddescription TEXT null
);

create table characteristic_value(
    id_characteristic_value INT NOT NULL PRIMARY KEY auto_increment,
    id_hero INT not null,
    id_characteristic_description INT not null,
    vvalue int not null,
    foreign key (id_hero) references hero(id_hero)
    on delete cascade
    on update cascade,
    foreign key (id_characteristic_description) references
characteristic_description(id_characteristic_description)
    on delete cascade
    on update cascade
);

create table skill(
```

```
    id_skill INT NOT NULL PRIMARY KEY auto_increment,
    id_hero int not null,
    nname VARCHAR(100) not null,
    llevel INT not null,
     ddescription TEXT null,
     foreign key (id_hero) references hero(id_hero)
     on delete cascade
     on update cascade
);

create table task(
    id_task INT NOT NULL PRIMARY KEY auto_increment,
    id_hero int not null,
    nname VARCHAR(100) not null,
    ddescription TEXT null,
    reward int not null,
    foreign key (id_hero) references hero(id_hero)
    on delete cascade
    on update cascade
);

create table item(
    id_item INT NOT NULL PRIMARY KEY auto_increment,
    cost int not null,
    nname VARCHAR(100) not null,
    ddescription TEXT null,
    llevel int not null
);

create table hero_item(
    id_bundle INT NOT NULL PRIMARY KEY auto_increment,
    id_hero INT not null,
    id_item INT not null,
    foreign key (id_hero) references hero(id_hero)
    on delete cascade
    on update cascade,
    foreign key (id_item) references item(id_item)
    on delete cascade
    on update cascade
);

create table shop(
    id_shop INT NOT NULL PRIMARY KEY auto_increment,
    race VARCHAR(100) not null,
    nname VARCHAR(100) not null,
    ddescription TEXT null,
    is_open bool not null
);

create table item_shop(
    id_bundle INT NOT NULL PRIMARY KEY auto_increment,
    id_item INT not null,
    id_shop INT not null,
    foreign key (id_item) references item(id_item)
    on delete cascade
    on update cascade,
    foreign key (id_shop) references shop(id_shop)
    on delete cascade
    on update cascade
);

INSERT INTO player (name, gender, birthday, email)
```

```
VALUES
('John Doe', 'Male', '1990-05-15', 'johndoe@example.com'),
('Jane Smith', 'Female', '1995-09-22', 'janesmith@example.com'),
('Michael Johnson', 'Male', '1988-12-10', 'michaeljohnson@example.com');

INSERT INTO achievement (id_player, nname, date_receipt, points)
VALUES
(1, 'First Achievement', '2023-01-10', 100),
(1, 'Second Achievement', '2023-02-20', 75),
(2, 'First Achievement', '2023-03-05', 60),
(3, 'First Achievement', '2023-04-15', 40);

INSERT INTO hero (id_player, nname, llevel, race)
VALUES
(1, 'Warrior', 20, 'Human'),
(1, 'Mage', 15, 'Elf'),
(2, 'Hunter', 18, 'Dwarf'),
(3, 'Rogue', 22, 'Orc');

INSERT INTO guild (nname, ddescription, race)
VALUES
('Warriors of Light', 'A guild of brave warriors', 'Human'),
('Magical Order', 'A guild of powerful mages', 'Elf'),
('Dwarven Brotherhood', 'A guild of skilled hunters', 'Dwarf'),
('Shadow Assassins', 'A guild of stealthy rogues', 'Orc');

INSERT INTO hero_guild (id_hero, id_guild)
VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4);

INSERT INTO characteristic_description (nname, ddescription)
VALUES
('Strength', 'Measures physical power'),
('Intelligence', 'Measures magical abilities'),
('Agility', 'Measures speed and dexterity');

INSERT INTO characteristic_value (id_hero, id_characteristic_description, vvalue)
VALUES
(1, 1, 80),
(1, 2, 60),
(1, 3, 70),
(2, 1, 100),
(2, 2, 80),
(2, 3, 90),
(3, 1, 70),
(3, 2, 40),
(3, 3, 80),
(4, 1, 60),
(4, 2, 70),
(4, 3, 90);

INSERT INTO skill (id_hero, nname, llevel, ddescription)
VALUES
(1, 'Sword Slash', 5, 'A powerful sword attack'),
(1, 'Fireball', 3, 'Launches a fireball at the enemy'),
(2, 'Arrow Shot', 4, 'Shoots arrows with precision'),
(2, 'Ice Bolt', 2, 'Casts a freezing spell'),
(3, 'Stealth', 5, 'Becomes invisible to enemies'),
(3, 'Backstab', 4, 'Deals massive damage from behind'),
```

```
(4, 'Dual Wield', 6, 'Wields two weapons simultaneously'),
(4, 'Poison Dagger', 3, 'Poisons the enemy with a dagger');

INSERT INTO task (id_hero, nname, ddescription, reward)
VALUES
(1, 'Defeat the Dragon', 'Slay the fearsome dragon in the cave', 100),
(2, 'Retrieve the Lost Artifact', 'Find the ancient artifact hidden in the ruins',
80),
(3, 'Rescue the Hostages', 'Save the captured villagers from bandits', 70),
(4, 'Assassinate the Warlord', 'Eliminate the enemy warlord in his stronghold', 120);

INSERT INTO item (cost, nname, ddescription, llevel)
VALUES
(100, 'Health Potion', 'Restores health when consumed', 1),
(100, 'Mana Potion', 'Restores mana when consumed', 1),
(200, 'Sword of Power', 'A legendary sword that grants immense strength', 10),
(1100, 'Staff of Fire', 'A staff imbued with the power of fire magic', 8),
(80, 'Bow of Precision', 'A finely crafted bow that enhances accuracy', 6),
(120, 'Dagger of Shadows', 'A dagger that grants stealth and poison abilities', 7);

INSERT INTO hero_item (id_hero, id_item)
VALUES
(1, 1),
(1, 2),
(2, 3),
(2, 4),
(3, 5),
(4, 6);

INSERT INTO shop (race, nname, ddescription, is_open)
VALUES
('Human', 'Warriors Emporium', 'A shop specializing in weapons and armor for
warriors', true),
('Elf', 'Arcane Wonders', 'A shop offering a variety of magical items and spell
scrolls', true),
('Dwarf', 'Hunting Supplies', 'A shop providing equipment for hunters and trappers',
true),
('Orc', 'Shadows Black Market', 'An underground shop selling illegal and rare items',
false);


INSERT INTO item_shop (id_item, id_shop)
VALUES
(1, 1),
(2, 1),
(3, 2),
(4, 2),
(5, 3),
(6, 4);
```

2. Создание процедур

1). Процедура для получения информации об игроках, которые получили определенное достижение. Эта процедура будет использовать соединение (JOIN) между таблицами player и achievement.

```
DELIMITER //
CREATE PROCEDURE GetPlayersWithAchievement(IN achievementName VARCHAR(100))
BEGIN
   SELECT p.name, p.email, a.nname AS achievement_name, a.date_receipt
   FROM player p
   JOIN achievement a ON p.id_player = a.id_player
   WHERE a.nname = achievementName;
END //
DELIMITER ;
```

Для вызова этой процедуры, используйте следующий синтаксис:

```
CALL GetPlayersWithAchievement('First Achievement');
```

2) Процедуру которая принимает название задания и возвращает список персонажей, у которых оно есть. Эта процедура будет использовать соединение (JOIN) между таблицами hero и task.

```
DELIMITER //
CREATE PROCEDURE GetHeroesWithTask(IN task_name VARCHAR(100))
BEGIN
 SELECT t.nname, h.nname, t.reward
 FROM hero h
 JOIN task t ON h.id_hero = t.id_hero
 WHERE t.nname = task_name;
END //
DELIMITER ;
```

Для вызова этой процедуры используйте следующий синтаксис:

```
CALL GetHeroesWithTask('Defeat the Dragon');
```

3) Процедура для получения информации о героях в определенной гильдии. Эта процедура будет использовать соединение (JOIN) между таблицами hero, hero_guild и guild:

```
DELIMITER //
CREATE PROCEDURE GetGuildMembers(IN guildName VARCHAR(100))
BEGIN
   SELECT h.nname, h.llevel, g.nname AS guild_name
   FROM hero h
   JOIN hero_guild hg ON h.id_hero = hg.id_hero
   JOIN guild g ON hg.id_guild = g.id_guild
   WHERE g.nname = guildName;
END //
DELIMITER ;
```

Для вызова этой процедуры используйте следующий синтаксис:

```
CALL GetGuildMembers('Warriors of Light');
```

4) Процедура для получения списка игроков с суммой их очков, полученных за достижения

```
DELIMITER //
CREATE PROCEDURE GetPlayerPoints()
BEGIN
   SELECT * FROM (
       WITH PlayerSummary AS (
           SELECT p.id_player, p.name, SUM(a.points) as TotalPoints
```

```
            FROM player p
            JOIN achievement a ON p.id_player = a.id_player
            GROUP BY p.id_player
        )
        SELECT ps.id_player, ps.name, ps.TotalPoints
        FROM PlayerSummary ps
    ) AS FinalResult;
END//
DELIMITER ;
```

Для вызова этой процедуры используйте следующий синтаксис:

```
CALL GetPlayerDetails();
```

5) Процедуру, которая извлекает список героев, обладающих определенной характеристикой. Например, мы могли бы получить всех героев, имеющих определенный уровень интеллекта.

```
DELIMITER //
CREATE PROCEDURE GetHeroesWithCharacteristic(IN id_characteristic_description INT, IN
min_value INT)
BEGIN
 SELECT * FROM (
     WITH CharacteristicHeroes AS (
         SELECT h.id_hero, h.nname, cd.nname as CharacteristicName, cv.vvalue
         FROM hero h
         JOIN characteristic_value cv ON h.id_hero = cv.id_hero
         JOIN characteristic_description cd ON cv.id_characteristic_description =
cd.id_characteristic_description
         WHERE cd.id_characteristic_description = id_characteristic_description AND
cv.vvalue >= min_value
     )
     SELECT ch.*
     FROM CharacteristicHeroes ch
 ) AS FinalResult;
END//
DELIMITER ;
```

Для вызова этой процедуры используйте следующий синтаксис:

```
CALL GetHeroesWithCharacteristic(2, 50);
```

3. Создание функций:

1) Функция по названию предмета, показывает магазин, в котором он продается

```
1) DELIMITER $$
CREATE FUNCTION FindItemShop(item_name VARCHAR(100))
RETURNS VARCHAR(100) DETERMINISTIC
BEGIN
 DECLARE shop_name VARCHAR(100);

 SELECT shop.nname INTO shop_name
 FROM item
 JOIN item_shop ON item.id_item = item_shop.id_item
 JOIN shop ON item_shop.id_shop = shop.id_shop
 WHERE item.nname = item_name;

 RETURN shop_name;
```

```
END$$
DELIMITER ;
```

Для вызова этой функции можно использовать следующий синтаксис:

```
SELECT FindItemShop('Sword of Power') AS ShopName;
```

2) Функция, которая по id игрока, возвращает количество его достижений:

```
DELIMITER $$
CREATE FUNCTION GetPlayerAchievements(playerId INT) RETURNS INT
READS SQL DATA
BEGIN
  DECLARE totalAchievements INT;
  SELECT COUNT(*) INTO totalAchievements
  FROM achievement JOIN player on player.id_player = achievement.id_player
  WHERE player.id_player = playerId;
  RETURN totalAchievements;
END$$
DELIMITER ;
```

Для вызова этой функции можно использовать следующий синтаксис:

```
SELECT GetPlayerAchievements(1) AS CountOfAchievements;
```

3) Функция, по id игрока, возвращает название задания, за которое даю больше всего награды

```
DELIMITER $$
CREATE FUNCTION getTaskByHeroId(id_hero INT) RETURNS VARCHAR(100)
READS SQL DATA
BEGIN
   DECLARE task_name VARCHAR(100);

   SELECT task.nname INTO task_name
   FROM hero
   JOIN task ON hero.id_hero = task.id_hero
   WHERE hero.id_hero = id_hero;

   RETURN task_name;
END$$
DELIMITER ;
```

Для вызова этой функции можно использовать следующий синтаксис:

```
SELECT getTaskByHeroId(1);
```

4) Функция принимает название расы в качестве входного параметра и возвращает список имен героев этой расы.

```
DELIMITER $$
CREATE FUNCTION get_heroes_by_race(race_name VARCHAR(255))
RETURNS VARCHAR(255)
READS SQL DATA
BEGIN
  DECLARE heroes_list VARCHAR(255);
  WITH heroes_cte AS (
    SELECT h.nname
    FROM hero h
    WHERE h.race = race_name
  )
  SELECT GROUP_CONCAT(nname) INTO heroes_list
  FROM heroes_cte;
  RETURN heroes_list;
```

```
END$$
DELIMITER ;
```

Для вызова этой функции можно использовать следующий синтаксис:

```
SELECT get_heroes_by_race('Elf');
```

5) Функция принимает название характеристики в качестве входного параметра и возвращает среднее значение этой характеристики по всем героям.

```
DELIMITER $$
CREATE FUNCTION get_average_characteristic_value(characteristic_name VARCHAR(255))
RETURNS DECIMAL(10,2)
READS SQL DATA
BEGIN
  DECLARE avg_value DECIMAL(10,2);
  WITH characteristic_cte AS (
    SELECT cv.vvalue
    FROM characteristic_description cd
    JOIN characteristic_value cv ON cd.id_characteristic_description =
cv.id_characteristic_description
    WHERE cd.nname = characteristic_name
  )
  SELECT AVG(vvalue) INTO avg_value
  FROM characteristic_cte;
  RETURN avg_value;
END$$
DELIMITER ;
```

Для вызова этой функции можно использовать следующий синтаксис:

```
SELECT get_average_characteristic_value('Strength');
```

4. Создание триггеров:

1) Триггер для таблицы player: Этот триггер будет автоматически обновлять поле birthday на следующий день после того, как игрок достигнет 18 лет.

```
DELIMITER $$
CREATE TRIGGER age_check AFTER UPDATE ON player
FOR EACH ROW
BEGIN
  IF (YEAR(CURDATE()) - YEAR(NEW.birthday)) >= 18 THEN
     UPDATE player SET birthday = DATE_ADD(birthday, INTERVAL 1 DAY);
  END IF;
END$$
DELIMITER ;
```

2) Триггер для таблицы achievement: Этот триггер будет автоматически увеличивать уровень героя на 1, когда он получает новую награду.

```
DELIMITER $$
CREATE TRIGGER level_up AFTER INSERT ON achievement
FOR EACH ROW
BEGIN
  UPDATE hero SET llevel = llevel + 1 WHERE id_hero = NEW.id_player;
END$$
DELIMITER ;
```

3) Триггер для таблицы hero_guild: Этот триггер будет автоматически обновлять описание гильдии, когда игрок присоединяется к ней.

```
DELIMITER $$
CREATE TRIGGER guild_join AFTER INSERT ON hero_guild
FOR EACH ROW
BEGIN
  UPDATE guild SET ddescription = CONCAT(ddescription, ', ', (SELECT nname FROM hero
WHERE id_hero = NEW.id_hero)) WHERE id_guild = NEW.id_guild;
END$$
DELIMITER ;
```

4) Триггер будет проверять, что возраст игрока не меньше 14 лет перед вставкой нового игрока в таблицу player

```
DELIMITER //
CREATE TRIGGER player_age_check
BEFORE INSERT
ON player
FOR EACH ROW
BEGIN
   DECLARE current_date_var DATE;
   SET current_date_var = CURDATE();
   IF TIMESTAMPDIFF(YEAR, NEW.birthday, current_date_var) < 14 THEN
       SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Player must be older than 14.';
   END IF;
END; //
DELIMITER ;
DELIMITER ;
```

5) Триггер будет проверять, что количество очков достижения больше или равно нулю перед вставкой нового достижения в таблицу achievement.

```
DELIMITER //
CREATE TRIGGER achievement_points_check
BEFORE INSERT
ON achievement
FOR EACH ROW
BEGIN
   IF NEW.points < 0 THEN
       SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Points cannot be negative.';
   END IF;
END; //
DELIMITER ;
```

5. Создание ролей

1) Создание роли Admin и предоставление полных привилегий

```
CREATE ROLE 'Admin';
GRANT ALL PRIVILEGES ON *.* TO 'Admin';
```

2) Роль "Чтение" для всех таблиц: Эта роль предоставляет права только на чтение для всех таблиц в базе данных.

```
CREATE ROLE read_only;
GRANT SELECT ON *.* TO 'read_only';
```

3) Роль "Менеджер игроков": Эта роль имеет полный контроль над таблицей player.

```
CREATE ROLE player_manager;
GRANT ALL PRIVILEGES ON player.* TO 'player_manager';
```

4) Роль "Администратор достижений": Эта роль имеет полный контроль над таблицей achievement.

```
CREATE ROLE achievement_admin;
GRANT ALL PRIVILEGES ON achievement.* TO 'achievement_admin';
```

5) Роль "Менеджер героев": Эта роль будет иметь полный контроль над таблицами hero, hero_guild, characteristic_value, skill, task, hero_item.

```
CREATE ROLE 'hero_manager';
GRANT ALL PRIVILEGES ON hero.* TO 'hero_manager';
GRANT ALL PRIVILEGES ON hero_guild.* TO 'hero_manager';
GRANT ALL PRIVILEGES ON characteristic_value.* TO 'hero_manager';
GRANT ALL PRIVILEGES ON skill.* TO 'hero_manager';
GRANT ALL PRIVILEGES ON task.* TO 'hero_manager';
GRANT ALL PRIVILEGES ON hero_item.* TO 'hero_manager';
```

6. Создание пользователей

Создание пользователей и присваивание им ролей, описанных выше

```
CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';
GRANT 'Admin' TO 'user1'@'localhost';
CREATE USER 'user2'@'localhost' IDENTIFIED BY 'password2';
GRANT 'Admin' TO 'user2'@'localhost';
CREATE USER 'user3'@'localhost' IDENTIFIED BY 'password3';
GRANT 'read_only' TO 'user3'@'localhost';
CREATE USER 'user4'@'localhost' IDENTIFIED BY 'password4';
GRANT 'read_only' TO 'user4'@'localhost';
CREATE USER 'user5'@'localhost' IDENTIFIED BY 'password5';
GRANT 'player_manager' TO 'user5'@'localhost';
CREATE USER 'user6'@'localhost' IDENTIFIED BY 'password6';
GRANT 'player_manager' TO 'user6'@'localhost';
CREATE USER 'user7'@'localhost' IDENTIFIED BY 'password7';
GRANT 'achievement_admin' TO 'user7'@'localhost';
CREATE USER 'user8'@'localhost' IDENTIFIED BY 'password8';
GRANT 'achievement_admin' TO 'user8'@'localhost';
CREATE USER 'user9'@'localhost' IDENTIFIED BY 'password9';
GRANT 'hero_manager' TO 'user9'@'localhost';
CREATE USER 'user10'@'localhost' IDENTIFIED BY 'password10';
GRANT 'hero_manager' TO 'user10'@'localhost';
```

7. Создание копии структуры БД

```
DROP DATABASE IF EXISTS game_world_copy;
CREATE DATABASE game_world_copy;
```

```
CREATE TABLE game_world_copy.player like game_world.player;
CREATE TABLE game_world_copy.achievement like game_world.achievement;
CREATE TABLE game_world_copy.hero like game_world.hero;
CREATE TABLE game_world_copy.guild like game_world.guild;
CREATE TABLE game_world_copy.hero_guild like game_world.hero_guild;
CREATE TABLE game_world_copy.characteristic_description like
game_world.characteristic_description;
CREATE TABLE game_world_copy.characteristic_value like
game_world.characteristic_value;
CREATE TABLE game_world_copy.skill like game_world.skill;
CREATE TABLE game_world_copy.task like game_world.task;
CREATE TABLE game_world_copy.item like game_world.item;
CREATE TABLE game_world_copy.hero_item like game_world.hero_item;
CREATE TABLE game_world_copy.shop like game_world.shop;
CREATE TABLE game_world_copy.item_shop like game_world.item_shop;
```

8. Создание копии данных БД

```
INSERT INTO game_world_copy.player SELECT * FROM game_world.player;
INSERT INTO game_world_copy.achievement SELECT * FROM game_world.achievement;
INSERT INTO game_world_copy.hero SELECT * FROM game_world.hero;
INSERT INTO game_world_copy.guild SELECT * FROM game_world.guild;
INSERT INTO game_world_copy.hero_guild SELECT * FROM game_world.hero_guild;
INSERT INTO game_world_copy.characteristic_description SELECT * FROM
game_world.characteristic_description;
INSERT INTO game_world_copy.characteristic_value SELECT * FROM
game_world.characteristic_value;
INSERT INTO game_world_copy.skill SELECT * FROM game_world.skill;
INSERT INTO game_world_copy.task SELECT * FROM game_world.task;
INSERT INTO game_world_copy.item SELECT * FROM game_world.item;
INSERT INTO game_world_copy.hero_item SELECT * FROM game_world.hero_item;
INSERT INTO game_world_copy.shop SELECT * FROM game_world.shop;
INSERT INTO game_world_copy.item_shop SELECT * FROM game_world.item_shop;
```