



Министерство науки и высшего образования Российской
Федерации Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский государственный технический
университет имени Н.Э. Баумана
(национальный исследовательский
университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Теоретическая информатика и компьютерные технологии

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ
ПО КУРСУ БАЗЫ ДАННЫХ НА
ТЕМУ:

Разработка базы данных ролевой
многопользовательской онлайн игры

Студент
подпись, дата

Баев Д.А.
фамилия, и.о.

Научный руководитель
подпись, дата

Домрачева А.Б.
фамилия, и.о.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1. Описание предметной области.....	5
2. Структура базы данных.....	6
2.1. Используемые сущности и их взаимосвязь.....	6
2.2. Преобразование модели «сущность-связь» в реляционную модель.....	10
2.3. Связи между сущностями.....	13
2.4. Свойства отношений.....	15
3. Реализация базы данных.....	27
4. Реализация клиент-серверного приложения.....	35
5. Тестирование.....	40
ЗАКЛЮЧЕНИЕ.....	42
СПИСОК ЛИТЕРАТУРЫ.....	43
Приложение А.....	44

ВВЕДЕНИЕ

В современном мире компьютерные игры, особенно ролевые многопользовательские онлайн игры, занимают особое место в развлекательной индустрии. Они предоставляют игрокам возможность погрузиться в увлекательные фантастические миры, где каждый может стать частью эпических приключений, развивать своих персонажей и взаимодействовать с другими игроками из разных уголков мира. Для обеспечения бесперебойной и удовлетворительной игровой среды, необходимо иметь эффективную и надежную базу данных, которая бы обеспечивала хранение, обработку и передачу игровой информации.

Разработка подобной базы данных позволит разработчикам игровой механики хранить всю информацию, связанную с аккаунтами игроков, их персонажами и всеми сведениями об этих персонажах: их предметы, изученные способности, полученные и выполненные задания, гильдия, в которую они вступили, список друзей, полученные и отправленные сообщения.

Это обеспечит возможность "восстановления" персонажа игрока между игровыми сессиями без потери достижений и полученных наград, без чего невозможно функционирование самой многопользовательской онлайн игры. Следовательно, тема курсовой работы актуальна в наши дни.

Таким образом, целью данной курсовой работы является: создание базы данных для хранения и обработки всей информации, необходимой для закрепления прогресса пользователей онлайн игры, а также тестирование этой базы данных с помощью имитации клиента и сервера игры.

Для достижения этой цели требуется:

- 1) Определить, какие аспекты ролевой многопользовательской онлайн игры будут включены в базу данных.
- 2) Спроектировать реляционную базу данных.
- 3) Реализовать клиент-серверное приложение, которое будет имитировать базовый игровой процесс.
- 4) Протестировать работоспособность.

1. Описание предметной области

За основу модели игры были взяты популярные ролевые многопользовательские онлайн игры: World of Warcraft и Аллоды Онлайн.

Центральным элементом предметной области являются персонажи. Эти персонажи создаются и управляются игроками, поэтому нужно поддерживать регистрацию и авторизацию аккаунтов этих игроков.

Каждый персонаж обладает набором уникальных способностей, которые они могут развивать по мере продвижения в игре.

В игре существует множество заданий, которые игроки могут выполнять, чтобы получать опыт, награды и продвигаться в сюжете игры.

Мир игры богат разнообразными предметами, которые персонажи могут собирать и использовать. Это включает в себя оружие, броню, зелья и многое другое.

Персонажи могут объединяться в гильдии для совместных приключений, соревнований и общения. Также они могут добавлять других персонажей в свой список друзей, чтобы легко взаимодействовать и играть вместе. Поэтому очень важна система чата, которая позволяет игрокам общаться, координировать действия, делиться информацией и обсуждать игровые события.

2. Структура базы данных

2.1. Используемые сущности и их взаимосвязь

После изучения предметной области можно выделить следующие основные сущности для базы данных:

Сначала выделим сущность Игрок (Player), которая будет хранить информацию о всех аккаунтах игроков: логин, пароль, дата регистрации, электронная почта и персонажи этого аккаунта.

Далее нужна сущность Персонаж (Character), которая отвечает за основную информацию о всех персонажах игроков, такую как: имя, фракция, раса, пол, класс, уровень, текущий опыт, текущее здоровье, текущая мана, количество золота, локацию, на которой находится персонаж, координаты персонажа на этой локации и гильдию, в которую вступил персонаж. Также нужно содержать информацию о персонажах-друзьях, поэтому для данной сущности присутствует рекурсивная связь.

Следом рассмотрим сущность Способность (Skill). База данных должна содержать информацию о всех способностях, которые персонаж может использовать в игре: название способности, сколько маны затрачивает эта способность, описание способности, время восстановления, класс, который может выучить способность. Некоторые способности можно изучить, только если изучены другие способности, поэтому у этой сущности также присутствует рекурсивная связь.

Следующей выделяется сущность Гильдия (Guild), ответственная за информацию о всех гильдиях в игре: название, участники, описание, минимальный уровень для вступления, лидер.

Далее рассмотрим сущность Сообщение (Message). Она хранит информацию о всех сообщениях: дата и время отправки, текст, отправитель, получатель. Сообщения могут быть адресованы в разные места: другому персонажу, гильдии или в общий чат локации. Поэтому эта сущность категория-подтип, дискриминатором которой является тип получателя.

Следом нужна сущность Задание (Quest). Она содержит информацию, о всех заданиях, которые доступны персонажам в игре: название, описание, необходимый уровень, является ли задание сюжетным, награды за выполнение.

Последней основной сущностью является сущность Предмет (Item). Она отвечает за информацию всех предметах, которые могут быть получены в игре: название, уровень, редкость, цена, описание. Предметы могут быть разных типов: оружие, элементы брони, другое. Поэтому эта сущность категории-подтип, дискриминатором которой является тип предмета. Предметы могут быть наградами за выполнение задания, поэтому между этими сущностями присутствует связь.

Помимо основных сущностей присутствует также несколько сопрягающих сущностей, которые обеспечивают связь между персонажами и некоторыми другими основными сущностями с дополнительной информацией о этой связи, а именно:

- 1) Выученная способность (Skill_learned): связывает персонажа и выученную им способность. Дополнительная информация: уровень способности.
- 2) Взятое задание (Quest_taken): связывает персонажа и взятое им задание. Дополнительная информация: является ли задание выполненным.
- 3) Предметы в инвентаре (Item_in_inventory): связывает персонажа и предмет в его инвентаре. Дополнительная информация: количество, является ли предмет экипированным (в случае, если он является оружием или элементом брони)

Итого, на основе всех требований была создана логическая модель данных - модель «сущность-связь». Эта модель представлена на Рисунках 1 - 3.

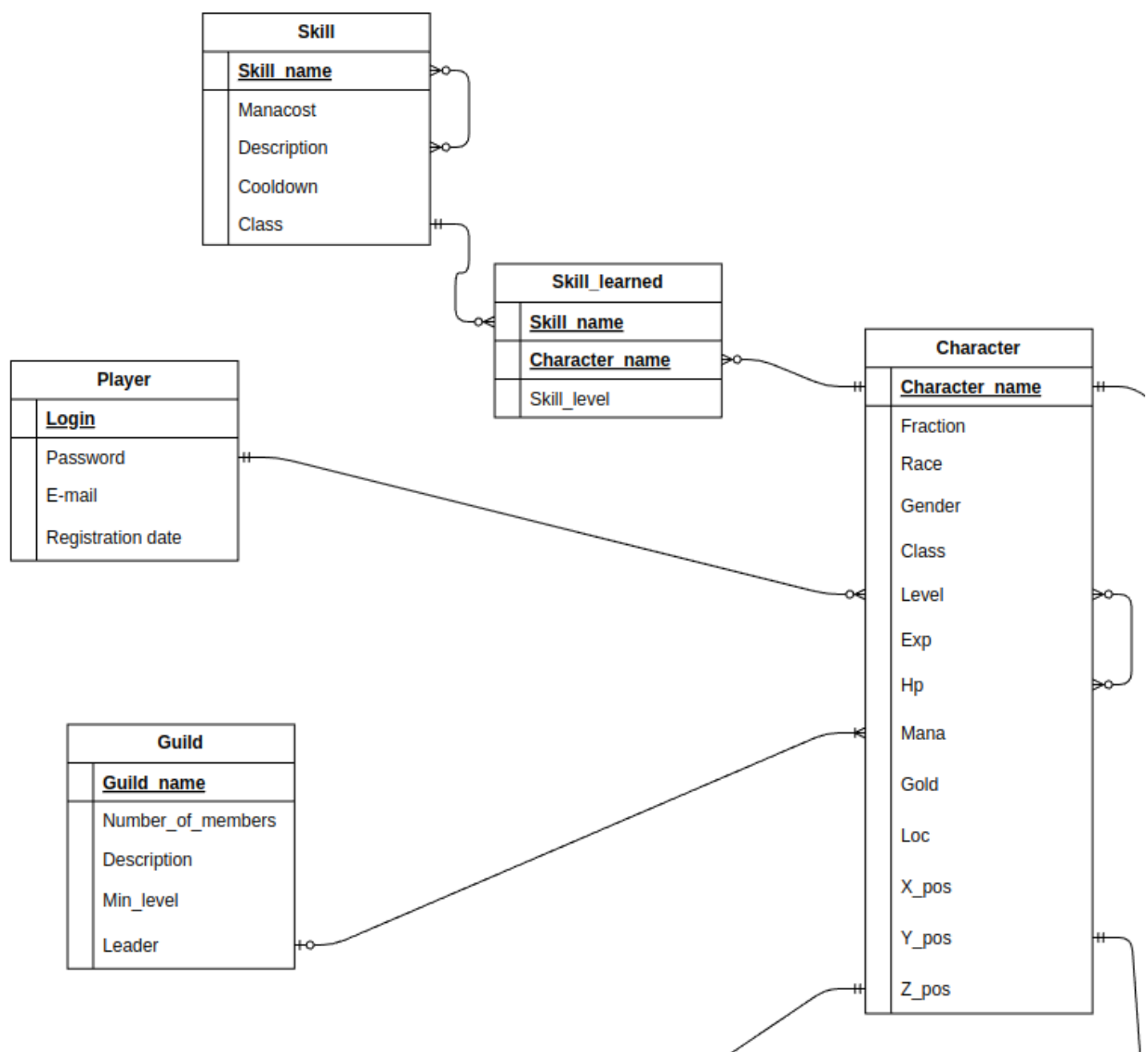


Рисунок 1 - Сущности Character, Skill, Player, Guild, Skill_learned.

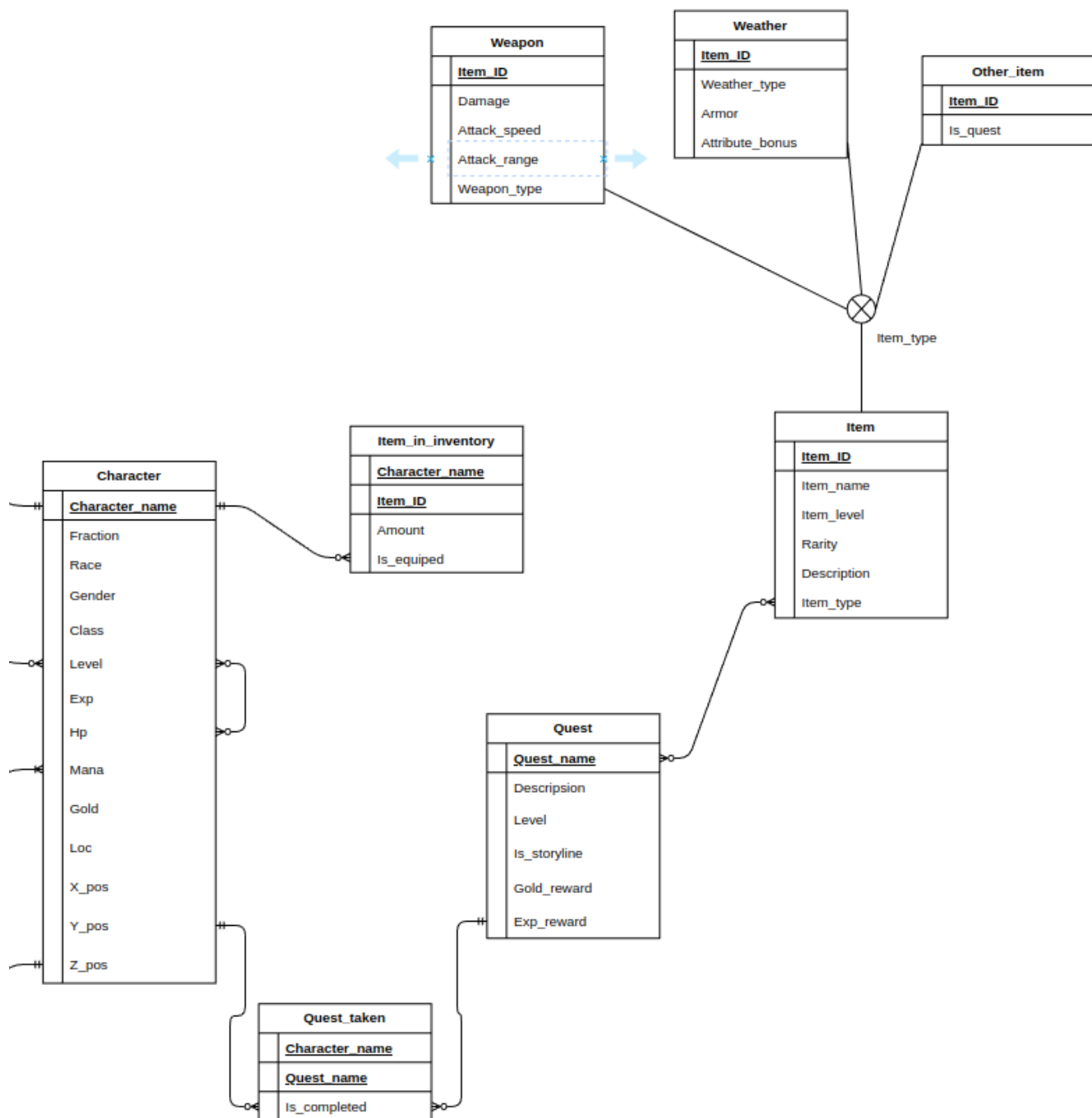


Рисунок 2 - Сущности Item, Quest, Quest_taken, Item_in_inventory.

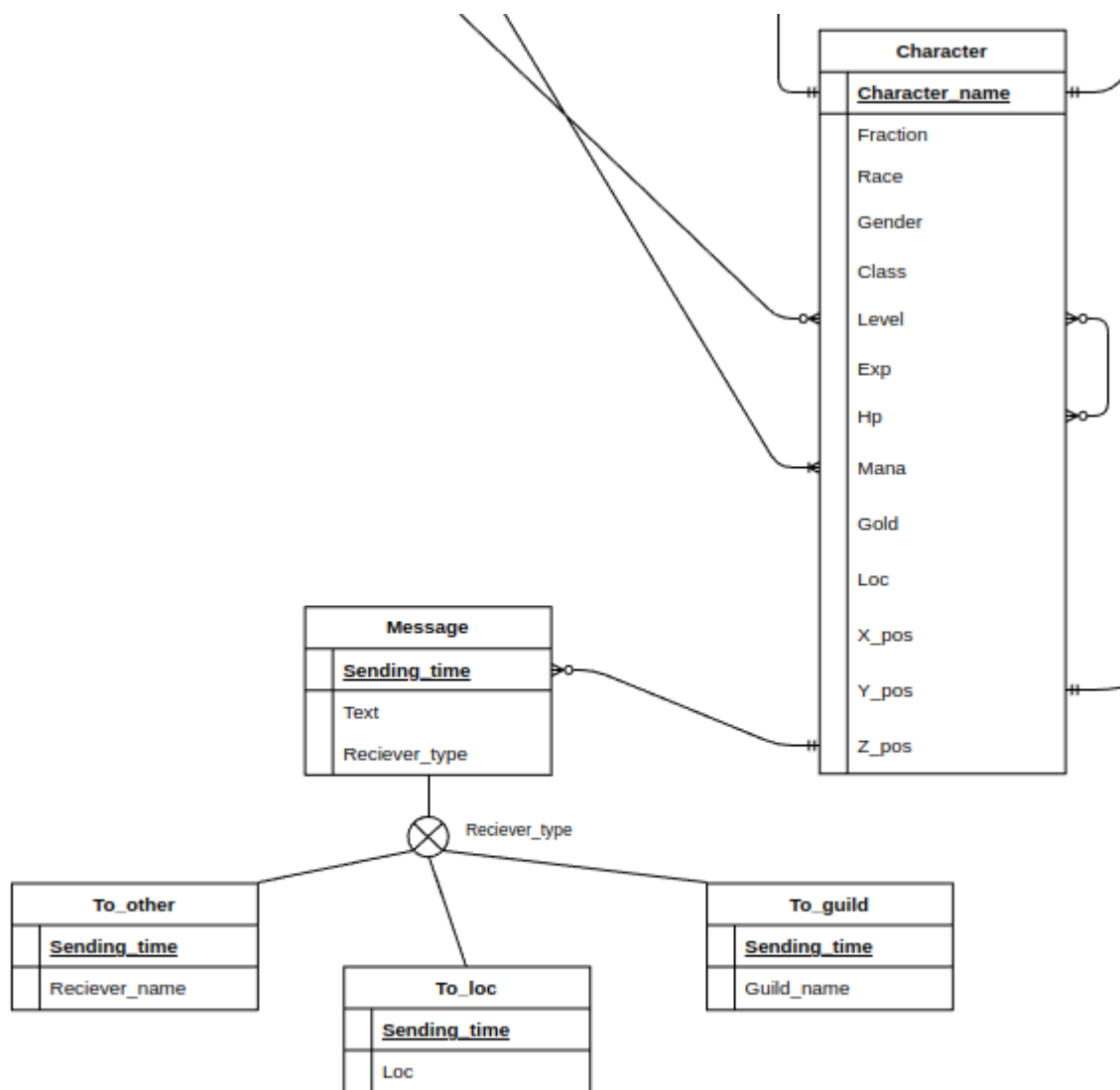


Рисунок 3 - Сущность Message.

2.2. Преобразование модели «сущность-связь» в реляционную модель

Из-за связи многие-ко-многим получают следующие вспомогательные сущности: Предмет-награда (Item_reward): для связи предметов и заданий, Персонажи-друзья (Character_Friends): для рекурсивной связи между персонажами, Нужные способности (Skill_need): для рекурсивной связи между способностями.

Подтипы сущностей Сообщение и Предмет также должны быть вынесены в отдельные таблицы.

Реляционная модель представлена на Рисунках 4 - 6.

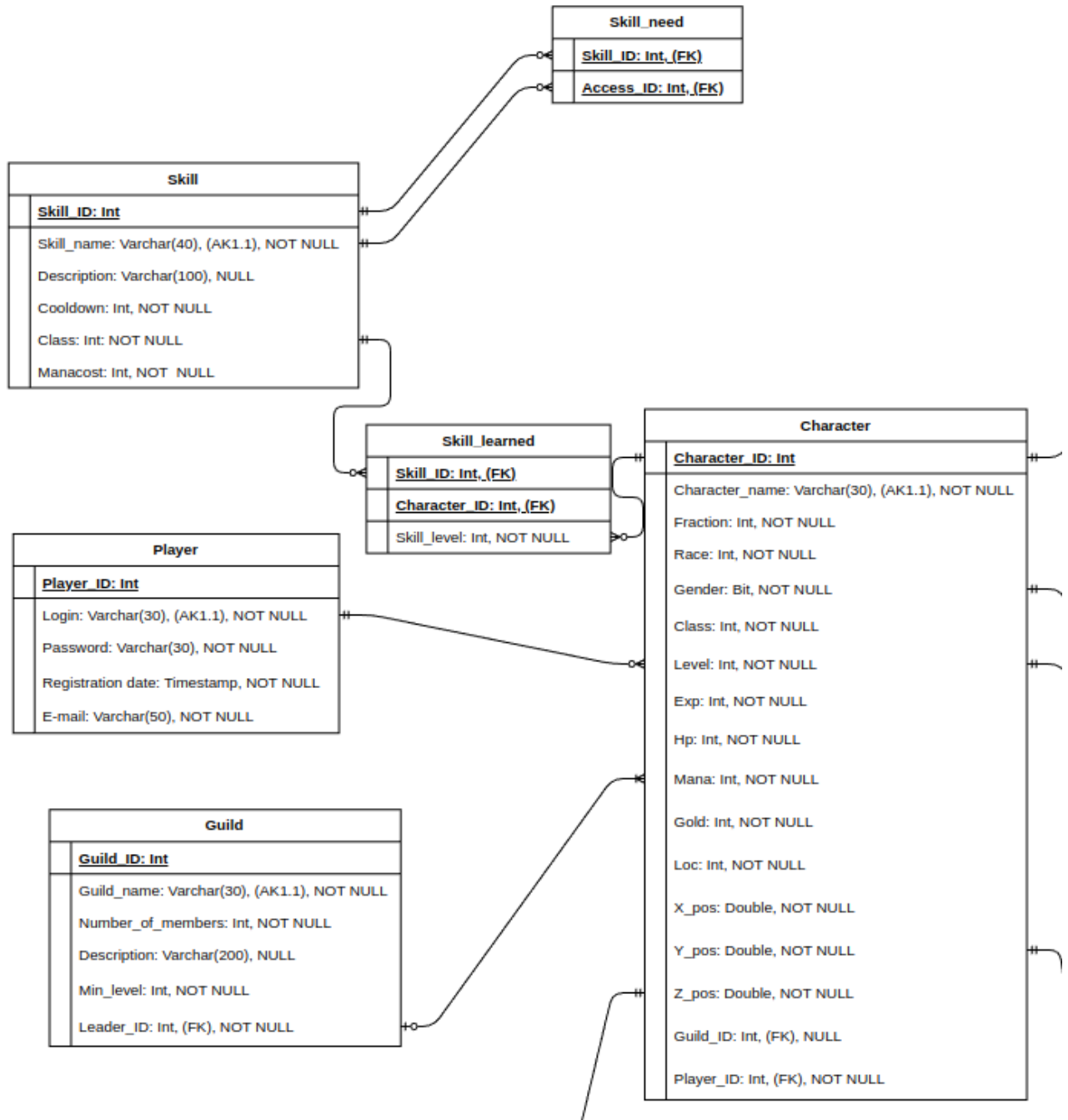


Рисунок 4 - Отношения Character, Player, Guild, Skill, Skill_learned, Skill_need.

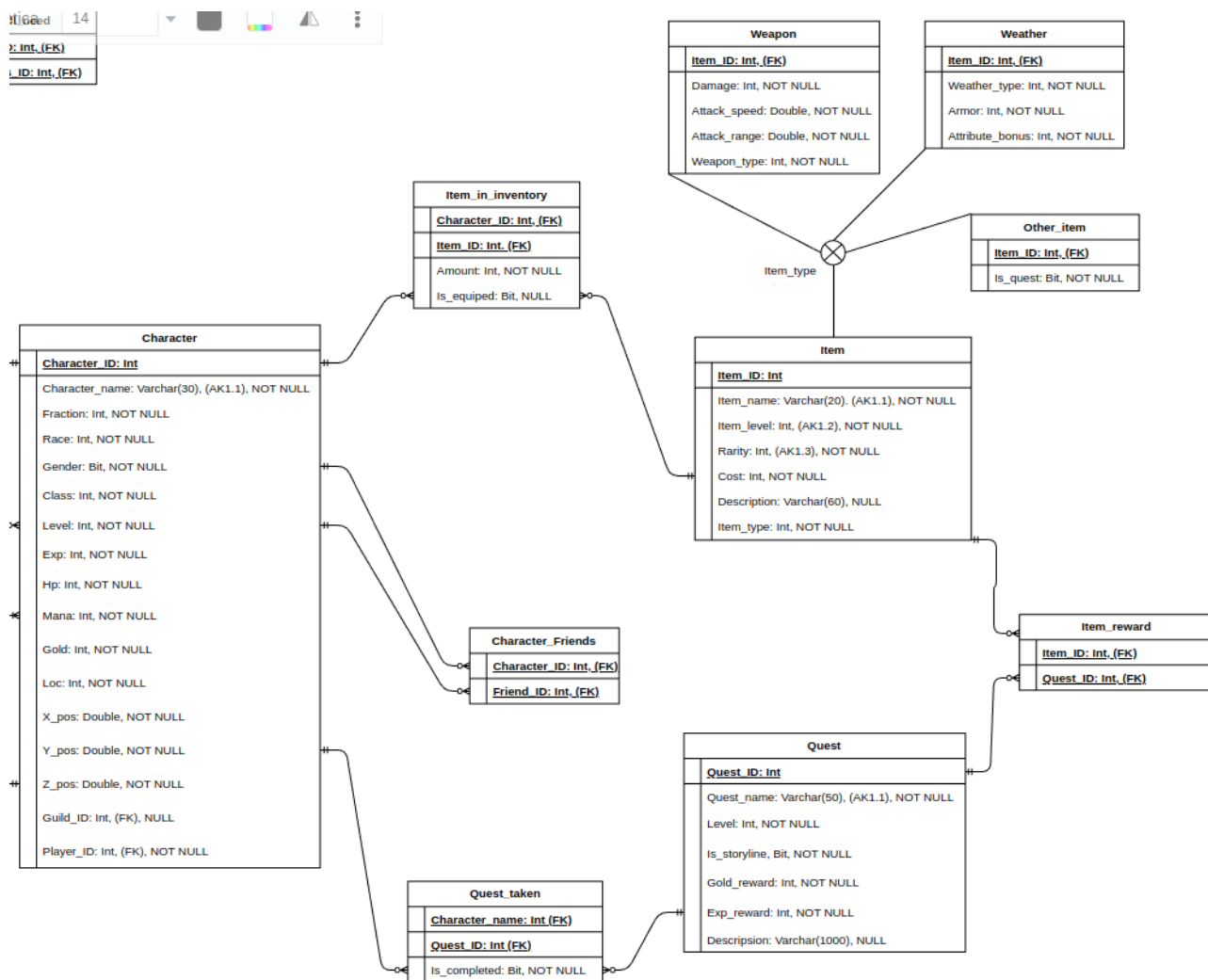


Рисунок 5 - Отношения Item, Weapon, Weather, Other_item, Quest, Quest_taken, Item_in_inventory, Item_reward, Character_friends.

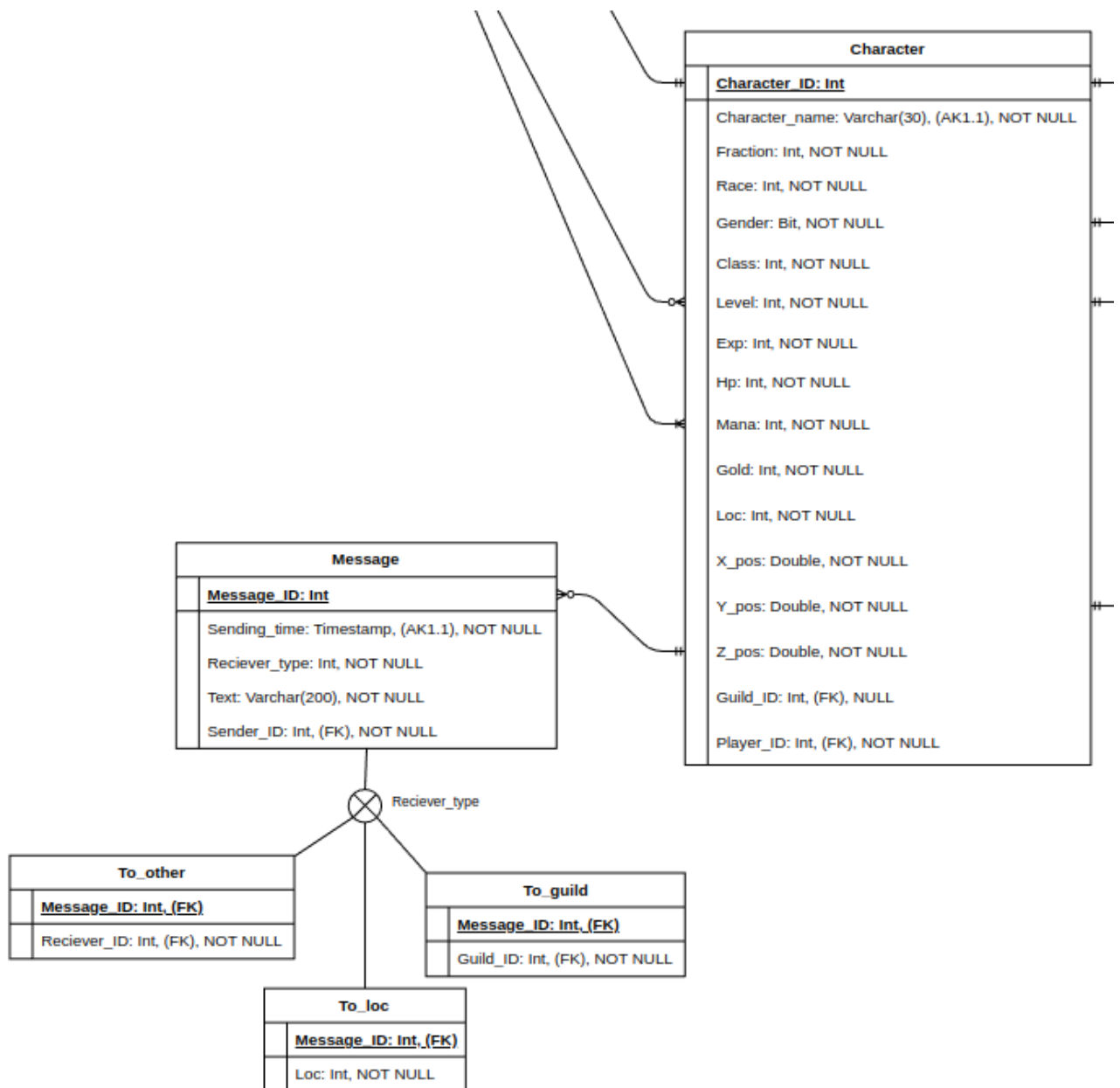


Рисунок 6 - Отношения Message, To_other, To_guild, To_loc.

2.3. Связи между сущностями

Связи между сущностями приведены в Таблице 1.

- 1) Игроки и персонажи: каждый персонаж обязан принадлежать одному игроку, у каждого игрока может быть 0 или более персонажей.
- 2) Персонажи и гильдии: персонаж может состоять максимум в 1 гильдии или не состоять ни в одной. В каждой гильдии должен быть по крайней мере один персонаж (по совместительству ее лидер).

- 3) Персонажи и способности: у персонажа может быть много способностей, а может быть и не одной (например, персонаж только был создан). Способность может быть изучена множеством персонажей, а может быть не изучена никаким (например, если способность только недавно была добавлена в игру)
- 4) Персонажи и предметы: у персонажа может быть много предметов, а может быть и не одного. Предмет может быть у множества персонажей, а может быть и не одного.
- 5) Персонажи и задания: у персонажа может быть много заданий, а может быть и не одного. Задание может быть выдано множеству персонажей, а может быть не выдано никому.
- 6) Персонажи и сообщения: персонаж может отправлять 0 или более сообщений. Каждое сообщение отправлено только одним персонажем.
- 7) Персонажи и персонажи: у каждого персонажа может быть 0 или более друзей.
- 8) Способности и способности: для изучения способности может требоваться 0 или более способностей. Каждая способность может влиять на изучение 0 или более способностей
- 9) Предметы и задания: за задание может быть выдано в награду 0 или более предметов. Каждый предмет может быть выдан в награду сразу за несколько заданий, а может и не выдаваться в награду за задания.

Parent	Child	Cardinality
Player	Character	0:N
Guild	Character	1:N
Character	Skill_learned	0:N
Skill	Skill_learned	0:N

Character	Item_in_inventory	0:N
Item	Item_in_inventory	0:N
Character	Quest_taken	0:N
Quest	Quest_taken	0:N
Character	Message	0:N
Character	Character_Friends	0:N
Skill	Skill_need	0:N
Item	Item_reward	0:N
Quest	Item_reward	0:N

Таблица 1 - Связи между сущностями.

2.4. Свойства отношений

Итого, база данных имеет 19 отношений:

Player (Таблица 2): содержит в себе информацию об аккаунтах игроков.

Содержит следующие атрибуты:

- Идентификатор (Player_ID)
- Имя пользователя (Login)
- Пароль (Password)
- Дата регистрации (Registration date)
- Адрес электронной почты (E-mail)

На имя пользователя и адрес электронной почты нужно наложить ограничения с использованием регулярных выражений, чтобы множество имен и множество адресов не пересекались и адрес был представлен в соответствии с общепринятым шаблоном.

Column Name	Type	Key	Null Status
-------------	------	-----	-------------

Player_ID	Int	Primary Key	NOT NULL
Login	Varchar (30)	Alternate Key	NOT NULL
Password	Varchar (30)	No	NOT NULL
Registration date	Date	No	NOT NULL
E-mail	Varchar (50)	No	NOT NULL

Таблица 2 - Отношение Player.

Character (Таблица 3): содержит в себе информацию о персонажах.

Содержит следующие атрибуты:

- Идентификатор (Character_ID)
- Имя персонажа (Character_name)
- Фракция персонажа (Fraction)
- Раса персонажа (Race)
- Пол персонажа (Gender)
- Класс персонажа (Class)
- Уровень персонажа (Level)
- Текущий опыт (Exp)
- Текущее здоровье (Hp)
- Текущая мана (Mana)
- Текущий запас золота (Gold)
- Локация, на которой находится персонаж (Loc)
- Позиция персонажа на локации по оси X (X_pos)
- Позиция персонажа на локации по оси Y (Y_pos)
- Позиция персонажа на локации по оси Z (Z_pos)
- Идентификатор гильдии персонажа (Guild_ID)
- Идентификатор аккаунта игрока, которому принадлежит персонаж (Player_ID)

Column Name	Type	Key	Null Status
Character_ID	Int	Primary Key	NOT NULL
Character_name	Varchar (30)	Alternate Key	NOT NULL
Fraction	Int	No	NOT NULL
Race	Int	No	NOT NULL
Gender	Int	No	NOT NULL
Class	Int	No	NOT NULL
Level	Int	No	NOT NULL
Exp	Int	No	NOT NULL
Hp	Int	No	NOT NULL
Mana	Int	No	NOT NULL
Gold	Int	No	NOT NULL
Loc	Int	No	NOT NULL
X_pos	Double	No	NOT NULL
Y_pos	Double	No	NOT NULL
Z_pos	Double	No	NOT NULL
Guild_ID	Int	Foreign Key	NULL
Player_ID	Int	Foreign Key	NOT NULL

Таблица 3 - Отношение Character.

Guild (Таблица 4): содержит в себе информацию о гильдиях. Содержит следующие атрибуты:

- Идентификатор (Guild_ID)

- Название гильдии (Guild_name)
- Текущее количество участников (Number_of_members) - значение должно быть от 1 до 50
- Описание (Description)
- Минимальный уровень, необходимый для вступления в гильдию (Min_level)
- Идентификатор лидера гильдии (Leader_ID)

Column Name	Type	Key	Null Status
Guild_ID	Int	Primary Key	NOT NULL
Guild_name	Varchar (30)	Alternate Key	NOT NULL
Number_of_members	Int	No	NOT NULL
Description	Varchar (200)	No	NULL
Min_level	Int	No	NOT NULL
Leader_ID	Int	Foreign Key	NOT NULL

Таблица 4 - Отношение Guild.

Skill (Таблица 5): содержит в себе информацию о способностях.

Содержит следующие атрибуты:

- Идентификатор (Skill_ID)
- Описание (Description)
- Время восстановления способности (Cooldown)
- Класс персонажа, необходимый для разучивания способности (Class)
- Количество маны, необходимое для применения способности (Manacost)

Column Name	Type	Key	Null Status
Skill_ID	Int	Primary Key	NOT NULL
Skill_name	Varchar (40)	Alternate Key	NOT NULL
Description	Varchar (100)	No	NULL
Cooldown	Int	No	NOT NULL
Class	Int	No	NOT NULL
Manacost	Int	No	NOT NULL

Таблица 5 - Отношение Skill.

Skill_need (Таблица 6): содержит в себе информацию о том, какие способности необходимы для изучения других способностей. Содержит следующие атрибуты:

- Идентификатор способности (Skill_ID)
- Идентификатор способности, для изучения которой нужно сперва изучить способность с идентификатором Skill_ID (Access_ID)

Column Name	Type	Key	Null Status
Skill_ID	Int	Primary Key, Foreign Key	NOT NULL
Access_ID	Int	Primary Key, Foreign Key	NOT NULL

Таблица 6 - Отношение Skill_need.

Skill_learned (Таблица 7): содержит в себе информацию о том, какие способности выучены у персонажей. Содержит следующие атрибуты:

- Идентификатор способности (Skill_ID)
- Идентификатор персонажа (Character_ID)

- Уровень способности (Skill_level).

Column Name	Type	Key	Null Status
Skill_ID	Int	Primary Key, Foreign Key	NOT NULL
Character_ID	Int	Primary Key, Foreign Key	NOT NULL
Skill_level	Int	No	NOT NULL

Таблица 7 - отношение Skill_learned.

Message (Таблица 8): содержит в себе информацию о сообщениях, которые могут отправлять персонажи. Содержит следующие атрибуты:

- Идентификатор (Message_ID)
- Время отправки сообщения (Sending_time)
- Тип получателя (Reciever_type) - есть три типа: 0 - другой игрок, 1 - чат гильдии, 2 - чат локации
- Текст сообщения (Text)
- Идентификатор отправителя (Sender_ID)

Column Name	Type	Key	Null Status
Message_ID	Int	Primary Key	NOT NULL
Sending_time	Timestamp	Alternate Key	NOT NULL
Reciever_type	Int	No	NOT NULL
Text	Varchar (200)	No	NOT NULL
Sender_ID	Int	Foreign Key	NOT NULL

Таблица 8 - отношение Message.

To_other (Таблица 9): содержит в себе информацию о подтипе сообщения - другому игроку. Содержит следующие атрибуты:

- Идентификатор сообщения (Message_ID)
- Идентификатор получателя (Reciever_ID)

Column Name	Type	Key	Null Status
Message_ID	Int	Primary Key, Foreign Key	NOT NULL
Reciever_ID	Int	Foreign Key	NOT NULL

Таблица 9 - отношение To_other.

To_guild (Таблица 10): содержит в себе информацию о подтипе сообщения - в чат гильдии. Содержит следующие атрибуты:

- Идентификатор сообщения (Message_ID)
- Идентификатор гильдии (Guild_ID)

Column Name	Type	Key	Null Status
Message_ID	Int	Primary Key, Foreign Key	NOT NULL
Guild_ID	Int	Foreign Key	NOT NULL

Таблица 10 - Отношение To_guild.

To_loc (Таблица 11): содержит в себе информацию о подтипе сообщения - в чат локации. Содержит следующие атрибуты:

- Идентификатор сообщения (Message_ID)
- Локация (Loc)

Column Name	Type	Key	Null Status
-------------	------	-----	-------------

Message_ID	Int	Primary Key, Foreign Key	NOT NULL
Loc	Int	No	NOT NULL

Таблица 11 - Отношение To_loc.

Character_Friends (Таблица 12): содержит в себе информацию о друзьях персонажей. Содержит следующие атрибуты:

- Идентификатор персонажа (Character_ID)
- Идентификатор его друга (Friend_ID)

Column Name	Type	Key	Null Status
Character_ID	Int	Primary Key, Foreign Key	NOT NULL
Friend_ID	Int	Primary Key, Foreign Key	NOT NULL

Таблица 12 - Отношение Character_Friends.

Quest (Таблица 13): содержит в себе информацию о заданиях, которые получают персонажи. Содержит следующие атрибуты:

- Идентификатор (Quest_ID)
- Название (Quest_name)
- Уровень, необходимый для получения задания (Level)
- Является ли задание сюжетным (Is_storyline)
- Золото, получаемое в награду (Gold_reward)
- Опыт, получаемый в награду (Exp_reward)
- Описание задания (Description)

Column Name	Type	Key	Null Status
-------------	------	-----	-------------

Quest_ID	Int	Primary Key	NOT NULL
Quest_name	Varchar (50)	Alternate Key	NOT NULL
Level	Int	No	NOT NULL
Is_storyline	Int	No	NOT NULL
Gold_reward	Int	No	NOT NULL
Exp_reward	Int	No	NOT NULL
Description	Varchar (1000)	No	NULL

Таблица 13 - Отношение Quest.

Quest_taken (Таблица 14): содержит в себе информацию о том, какие задания взяты у персонажей. Содержит следующие атрибуты:

- Идентификатор персонажа (Character_ID)
- Идентификатор задания (Quest_ID)
- Выполнено ли задание (Is_completed)

Column Name	Type	Key	Null Status
Character_ID	Int	Primary Key, Foreign Key	NOT NULL
Quest_ID	Int	Primary Key, Foreign Key	NOT NULL
Is_completed	Int	No	NOT NULL

Таблица 14 - Отношение Quest_taken.

Item (Таблица 15): содержит в себе информацию о предметах. Содержит следующие атрибуты:

- Идентификатор (Item_ID)
- Название (Item_name)

- Уровень предмета (Item_level)
- Редкость (Rarity)
- Описание предмета (Description)
- Тип предмета (Item_type) - есть три типа: 0 - оружие, 1 - элемент брони, 2 - все остальное

Column Name	Type	Key	Null Status
Item_ID	Int	Primary Key	NOT NULL
Item_name	Varchar (20)	Alternate Key	NOT NULL
Item_level	Int	Alternate Key	NOT NULL
Rarity	Int	Alternate Key	NOT NULL
Description	Varchar (60)	No	NULL
Item_type	Int	No	NOT NULL

Таблица 15 - Отношение Item.

Weapon (Таблица 16): содержит в себе информацию о типе предмета - оружиях. Содержит следующие атрибуты:

- Идентификатор предмета (Item_ID)
- Урон оружия (Damage)
- Скорость атаки оружия (Attack_speed)
- Дальность атаки оружия (Attack_range)
- Тип оружия (Weapon_type)

Column Name	Type	Key	Null Status
Item_ID	Int	Primary Key, Foreign Key	NOT NULL

Damage	Int	No	NOT NULL
Attack_speed	Double	No	NOT NULL
Attack_range	Double	No	NOT NULL
Weapon_type	Int	No	NOT NULL

Таблица 16 - Отношение Weapon.

Weather (Таблица 17): содержит в себе информацию о типе предмета - элементах брони. Содержит следующие атрибуты:

- Идентификатор предмета (Item_ID)
- Тип элемента брони (Weather_type)
- Прибавка к броне (Armor)
- Прибавка к атрибуту (Attribute_bonus)

Column Name	Type	Key	Null Status
Item_ID	Int	Primary Key, Foreign Key	NOT NULL
Weather_type	Int	No	NOT NULL
Armor	Int	No	NOT NULL
Attribute_bonus	Int	No	NOT NULL

Таблица 17 - Отношение Weather.

Other_item (Таблица 18): содержит в себе информацию о типе предмета - остальные предметы. Содержит следующие атрибуты:

- Идентификатор предмета (Item_ID)
- Необходим ли этот предмет для выполнения задания (Is_quest)

Column Name	Type	Key	Null Status
-------------	------	-----	-------------

Item_ID	Int	Primary Key, Foreign Key	NOT NULL
Is_quest	Bit	No	NOT NULL

Таблица 18 - Отношение Other_item.

Item_in_inventory (Таблица 19): содержит в себе информацию о том, какие предметы есть у персонажей в инвентаре. Содержит следующие атрибуты:

- Идентификатор персонажа (Character_ID)
- Идентификатор предмета (Item_ID)
- Количество (Amount)
- Экипирован ли предмет на персонажа (Is_equipped) - должен быть NULL, если предмет является Other_item. У персонажа может быть экипировано не более одного оружия и не более одного элемента брони каждого типа.

Column Name	Type	Key	Null Status
Character_ID	Int	Primary Key, Foreign Key	NOT NULL
Item_ID	Int	Primary Key, Foreign Key	NOT NULL
Amount	Int	No	NOT NULL
Is_equipped	Bit	No	NULL

Таблица 19 - Отношение Item_in_inventory.

Item_taken (Таблица 20): содержит в себе информацию о том, какие предметы выдаются в награду за задания. Содержит следующие атрибуты:

- Идентификатор предмета (Item_ID)

- Идентификатор задания, в награду за которое выдается предмет (Quest_ID)

Column Name	Type	Key	Null Status
Item_ID	Int	Primary Key, Foreign Key	NOT NULL
Quest_ID	Int	Primary Key, Foreign Key	NOT NULL

Таблица 20 - Отношение Item_taken.

3. Реализация базы данных

Для создания базы данных была выбрана свободная реляционная система управления базами данных (СУБД) - PostgreSQL [1].

Это мощная СУБД с открытым исходным кодом. Она предоставляет разработчикам богатый набор функциональных возможностей и инструментов для эффективного хранения, организации и управления данными.

PostgreSQL поддерживает стандартный язык структурированных запросов SQL. Также он обеспечивает хорошую производительность благодаря оптимизации запросов, многозадачности и возможности распараллеливания запросов и поддерживает основные свойства транзакций: атомарность, согласованность, изоляция, устойчивость.

В результате, использование PostgreSQL обеспечивает надежную и мощную основу для базы данных ролевой многопользовательской онлайн игры, удовлетворяя всем необходимым требованиям для подобного типа игр.

Простые ограничения реализуются с помощью CHECK (как, например, в Листинге 1)

```
1. create table players (  
2.   Player_ID serial primary key,  
3.   Login varchar(30) unique not null CHECK (Login ~* '^[^@]+$'),  
4.   Password varchar(30) not null,  
5.   Registration_date date not null,  
6.   E_mail varchar(50) unique not null CHECK (E_mail ~*  
7.     '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$')
```

Листинг 1 - Создание таблицы players.

Более сложные ограничения, а также некоторые действия, которые требует механика игры, реализуются с помощью триггеров. Триггеры в PostgreSQL - это специальные хранимые процедуры, которые автоматически выполняются в ответ на определенные события, происходящие в базе данных (до или после них).

В листинге 2 приведен триггер `update_guild_id`. Он срабатывает, когда персонаж вступает в гильдию или покидает ее, и выполняет сразу несколько функций:

- Проверяет соответствие минимальному уровню гильдии
- Проверяет, что персонаж не вступит в гильдию, если она полностью заполнена.
- Автоматически регистрирует добавление или удаление члена гильдии (поле `number_of_members`).
- Если гильдию покидает лидер, то автоматически назначает лидером участника с самым высоким уровнем.
- Удаляет гильдию, если ее покидает последний участник.

```
1.  CREATE OR REPLACE FUNCTION update_guild_id()
2.  RETURNS TRIGGER AS $$
3.  DECLARE
4.      new_guild_id INT;
5.      current_members INT;
6.      new_leader_id INT;
7.  BEGIN
8.      new_guild_id := NEW.Guild_ID;
9.
10.     IF new_guild_id IS NOT NULL THEN
11.         IF OLD.level < (SELECT Minimal_level from guilds where
12.             Guild_ID = NEW.guild_id) then
13.             RAISE EXCEPTION 'Your level is less than minimal in this
14.             guild';
15.         end if;
16.
17.         SELECT Number_of_members INTO current_members
18.         FROM guilds
19.         WHERE Guild_ID = new_guild_id;
20.
21.         IF current_members >= 50 THEN
22.             RAISE EXCEPTION 'The guild already has the maximum number
23.             of members (50)';
24.         END IF;
```

```

22.
23.             IF OLD.Character_ID != (SELECT Leader_ID FROM guilds WHERE
Guild_ID = NEW.Guild_ID) THEN
24.                 UPDATE guilds
25.                     SET Number_of_members = Number_of_members + 1
26.                     WHERE Guild_ID = new_guild_id;
27.             end if;
28.
29.         END IF;
30.
31.     IF new_guild_id IS NULL THEN
32.         IF OLD.Guild_ID IS NOT NULL THEN
33.             SELECT Number_of_members INTO current_members
34.             FROM guilds
35.             WHERE Guild_ID = OLD.Guild_ID;
36.
37.             IF current_members > 1 THEN
38.                 UPDATE guilds
39.                     SET Number_of_members = Number_of_members - 1
40.                     WHERE Guild_ID = OLD.Guild_ID;
41.             ELSE
42.                 DELETE FROM guilds
43.                 WHERE Guild_ID = OLD.Guild_ID;
44.             END IF;
45.
46.             IF OLD.Character_ID = (SELECT Leader_ID FROM guilds WHERE
Guild_ID = OLD.Guild_ID) THEN
47.                 SELECT Character_ID INTO new_leader_id
48.                 FROM characters
49.                 WHERE Guild_ID = OLD.Guild_ID
50.                 ORDER BY Level DESC
51.                 LIMIT 1;
52.
53.                 UPDATE guilds
54.                     SET Leader_ID = new_leader_id
55.                     WHERE Guild_ID = OLD.Guild_ID;
56.             END IF;
57.         END IF;
58.     END IF;
59.
60.     RETURN NEW;
61. END;
62. $$ LANGUAGE plpgsql;

```

```

63.
64.
65. CREATE TRIGGER update_guild_id_trigger
66. AFTER UPDATE ON characters
67. FOR EACH ROW
68. WHEN ((NEW.Guild_ID IS NULL AND OLD.Guild_ID IS NOT NULL) or
        ((NEW.Guild_ID IS NOT NULL AND OLD.Guild_ID IS NULL)))
69. EXECUTE FUNCTION update_guild_id();

```

Листинг 2 - Триггер update_guild_id.

Также существует триггер, который выполняет схожую логику, если персонаж покидает гильдию по причине того, что его удалили.

В листинге 3 приведен триггер, задача которого запрещать вставку повторных друзей в Characters_friends.

```

1. CREATE OR REPLACE FUNCTION check_duplicate_friendship()
2. RETURNS TRIGGER AS $$
3. BEGIN
4.     IF EXISTS (
5.         SELECT 1 FROM characters_friends
6.         WHERE (Character_ID = NEW.Friend_ID AND Friend_ID =
NEW.Character_ID)
7.         OR (Character_ID = NEW.Character_ID AND Friend_ID = NEW.Friend_ID)
8.     ) THEN
9.         RAISE EXCEPTION 'Duplicate friendship';
10.    END IF;
11.    RETURN NEW;
12. END;
13. $$ LANGUAGE plpgsql;
14.
15. CREATE TRIGGER prevent_duplicate_friendship
16. BEFORE INSERT ON characters_friends
17. FOR EACH ROW
18. EXECUTE FUNCTION check_duplicate_friendship();

```

Листинг 3 - Триггер prevent_duplicate_friendship.

В листинге 4 приведен триггер, который запрещает изучать способности в соответствии с таблицей Skill_need.

```

1.  CREATE OR REPLACE FUNCTION check_need_learn()
2.  RETURNS TRIGGER AS $$
3.  BEGIN
4.      IF NOT EXISTS (
5.          SELECT 1
6.          FROM skill_need sn
7.          LEFT JOIN skill_learned sl ON sn.Skill_ID = sl.Skill_ID AND
8.          sl.Character_ID = NEW.Character_ID
9.          WHERE sn.Access_ID = NEW.Skill_ID
10.         AND sl.Skill_ID IS NULL
11.     ) THEN
12.         RETURN NEW;
13.     ELSE
14.         RAISE EXCEPTION 'Character does not have access to the
15.         required skill';
16.     END IF;
17. END;
18. $$ LANGUAGE plpgsql;
19.
20. CREATE TRIGGER check_character_learned_trigger
21. BEFORE INSERT ON skill_learned
22. FOR EACH ROW
23. EXECUTE FUNCTION check_need_learn();

```

Листинг 4 - Триггер check_character_learned_trigger.

В листинге 5 приведен триггер, который запрещает персонажам надевать предметы уровнем выше их и обеспечивает логику того, что одновременно могут быть экипированы только одно оружие и один элемент брони каждого типа.

```

1.  CREATE OR REPLACE FUNCTION update_equipped()
2.  RETURNS TRIGGER AS $$
3.  DECLARE
4.      item_level INT;
5.      character_level INT;
6.  BEGIN
7.      IF NEW.Is_equipped = 1 THEN
8.          SELECT items.Item_level INTO item_level
9.          FROM items
10.         WHERE Item_ID = NEW.Item_ID;

```



```

11.
12.         SELECT characters.Level INTO character_level
13.         FROM characters
14.         WHERE Character_ID = NEW.Character_ID;
15.
16.         IF item_level > character_level THEN
17.             RAISE EXCEPTION 'You can not equip item with a level
higher than the character level';
18.         end if;
19.
20.         IF (SELECT Item_type FROM items WHERE Item_ID = NEW.Item_ID) =
0 THEN
21.             UPDATE item_in_inventory
22.             SET Is_equipped = 0
23.             WHERE Character_ID = NEW.Character_ID
24.             AND Item_ID != NEW.Item_ID
25.             AND (SELECT Item_type FROM items WHERE Item_ID =
Item_in_inventory.Item_ID) = 0;
26.         END IF;
27.
28.         IF (SELECT Item_type FROM items WHERE Item_ID = NEW.Item_ID) =
1 THEN
29.             UPDATE item_in_inventory
30.             SET Is_equipped = 0
31.             WHERE Character_ID = NEW.Character_ID
32.             AND Item_ID != NEW.Item_ID
33.             AND (SELECT Item_type FROM items WHERE Item_ID =
Item_in_inventory.Item_ID) = 1
34.             AND (SELECT Weather_type FROM weather WHERE Item_ID =
Item_in_inventory.Item_ID) = (SELECT Weather_type FROM weather WHERE Item_ID
= NEW.Item_ID);
35.         END IF;
36.
37.         IF (SELECT Item_type FROM items WHERE Item_ID = NEW.Item_ID) =
2 THEN
38.             RAISE EXCEPTION 'This item can not be equipped';
39.         END IF;
40.     END IF;
41.     RETURN NEW;
42. END;
43. $$ LANGUAGE plpgsql;
44.
45. CREATE TRIGGER equipped_trigger

```

```
46. BEFORE UPDATE ON item_in_inventory
47. FOR EACH ROW
48. WHEN (NEW.Is_equipped != OLD.Is_equipped)
49. EXECUTE FUNCTION update_equipped();
```

Листинг 5 - Триггер equipped_trigger.

4. Реализация клиент-серверного приложения

Для разработки клиент-серверного приложения использовался язык программирования Python [2]. Python - это интерпретируемый, высокоуровневый язык программирования с простым и понятным синтаксисом, который имеет огромное количество библиотек для самых разных задач.

Для разработки части, которая должна имитировать клиент игры, использовалась библиотека Tkinter [3]. Это стандартная библиотека Python, предназначенная для создания графических пользовательских интерфейсов (ГПИ) на базе библиотеки Tk. Tkinter предоставляет простой и интуитивно понятный интерфейс для создания оконных приложений с графическими элементами и поэтому является отличным выбором для создания простой оконной модели клиента.

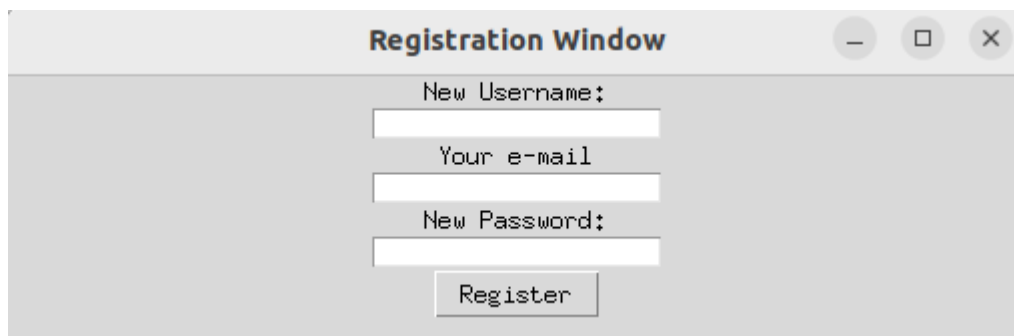
Серверная часть, в свою очередь, использует библиотеку Psycopg2 [4]. Psycopg2 - это библиотека для Python, предназначенная для работы с базой данных PostgreSQL. Она позволяет устанавливать соединение между Python-приложением и сервером PostgreSQL, выполнять SQL запросы, создавать курсоры для обработки результатов запросов, а также управлять транзакциями.

Для взаимодействия клиентской и серверной частей приложения используется библиотека Socket [5]. Она предоставляет средства для создания и управления сетевыми соединениями. Это мощный инструмент для работы с сетью в Python, который позволяет создавать разнообразные сетевые приложения.

Клиент и сервер общаются между собой, передавая сообщения в формате JSON (JavaScript Object Notation) [6]. Это легкий текстовый формат обмена данными, который широко используется для передачи данных между клиентом и сервером.

В приложении А приведены: листинг подключения к базе данных и листинги некоторых запросов.

На Рисунке 6 приведено окно регистрации.



The image shows a 'Registration Window' with a title bar containing standard window controls. The window has a light gray background. It contains three text input fields stacked vertically, each with a label above it: 'New Username:', 'Your e-mail', and 'New Password:'. Below these fields is a single button labeled 'Register'.

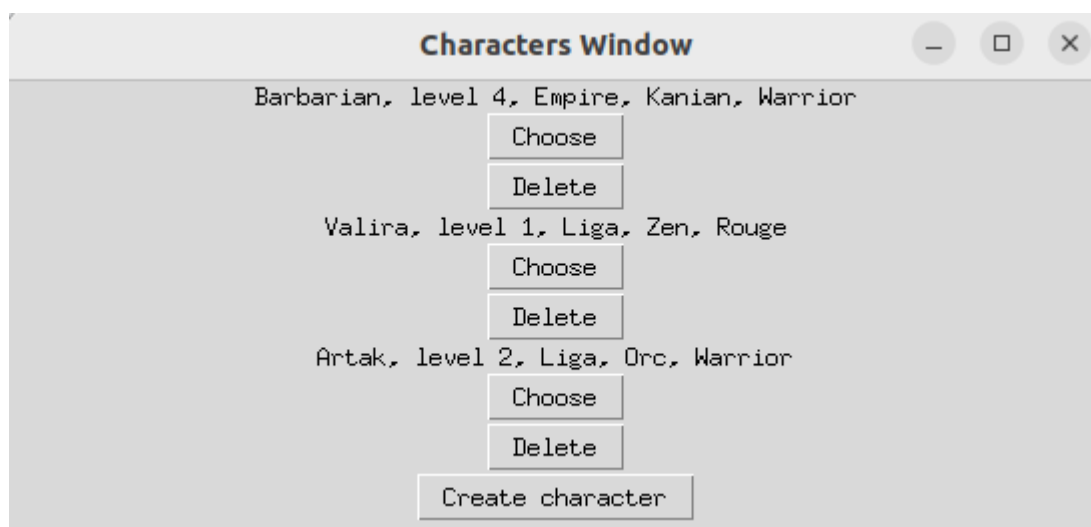
Рисунок 6 - Окно регистрации.

После регистрации можно авторизоваться (Рисунок 7) и попасть в меню выбора персонажей (Рисунок 8).



The image shows an 'Authorization Window' with a title bar containing standard window controls. The window has a light gray background. It contains two text input fields stacked vertically, each with a label above it: 'Login or e-mail:' and 'Password:'. The 'Login or e-mail' field contains the text 'p3v'. The 'Password' field contains a single asterisk '*'. Below these fields are three buttons stacked vertically: 'Login', 'Register', and 'Delete'.

Рисунок 7 - Окно авторизации



The image shows a 'Characters Window' with a title bar containing standard window controls. The window has a light gray background. It displays a list of characters with their details and two buttons for each character. The characters are: 'Barbarian, level 4, Empire, Kanian, Warrior', 'Valira, level 1, Liga, Zen, Rouge', and 'Artak, level 2, Liga, Orc, Warrior'. For each character, there are 'Choose' and 'Delete' buttons. At the bottom of the window is a button labeled 'Create character'.

Рисунок 8 - Окно выбора персонажа.

В меню можно создать персонажа (Рисунок 9) или попасть в имитацию игрового мира (Рисунок 10).

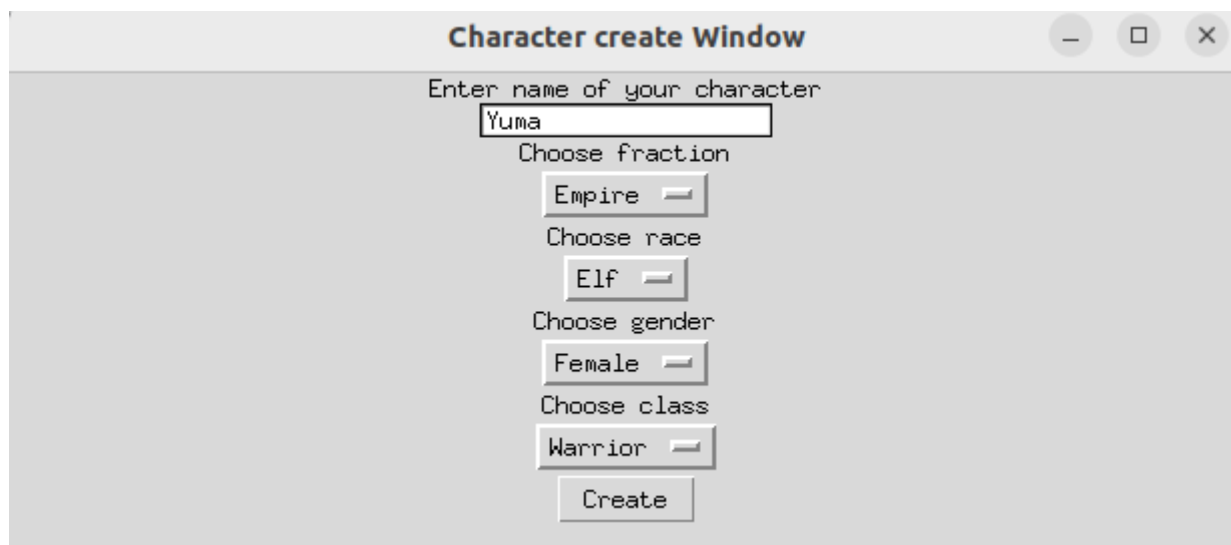


Рисунок 9 - Окно создания персонажа.



Рисунок 10 - Окно «игры».

Далее на рисунках 11 - 16 будут приведены окна отдельных элементов игры.

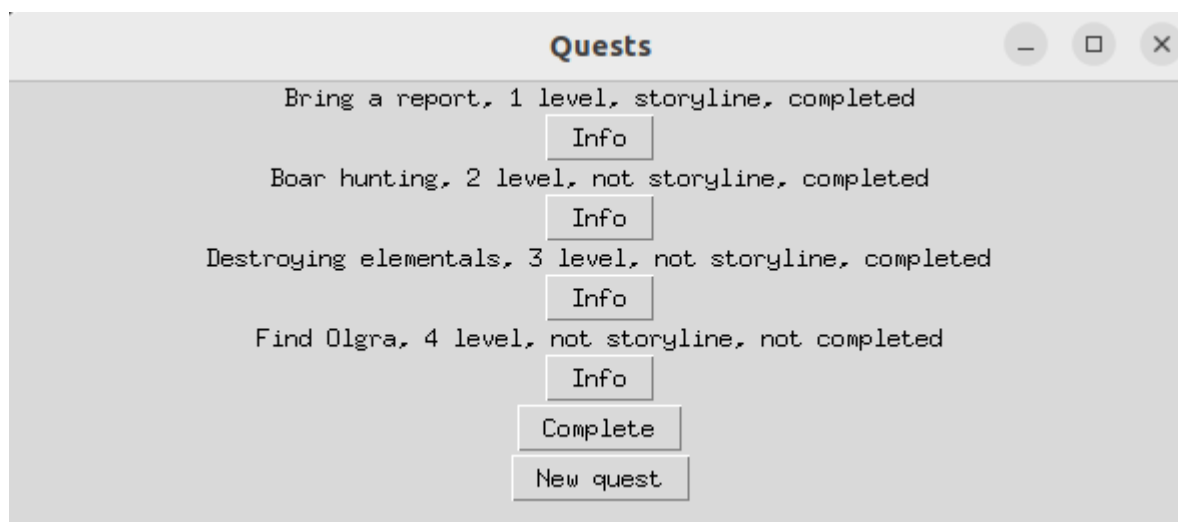


Рисунок 11 - Окно заданий.

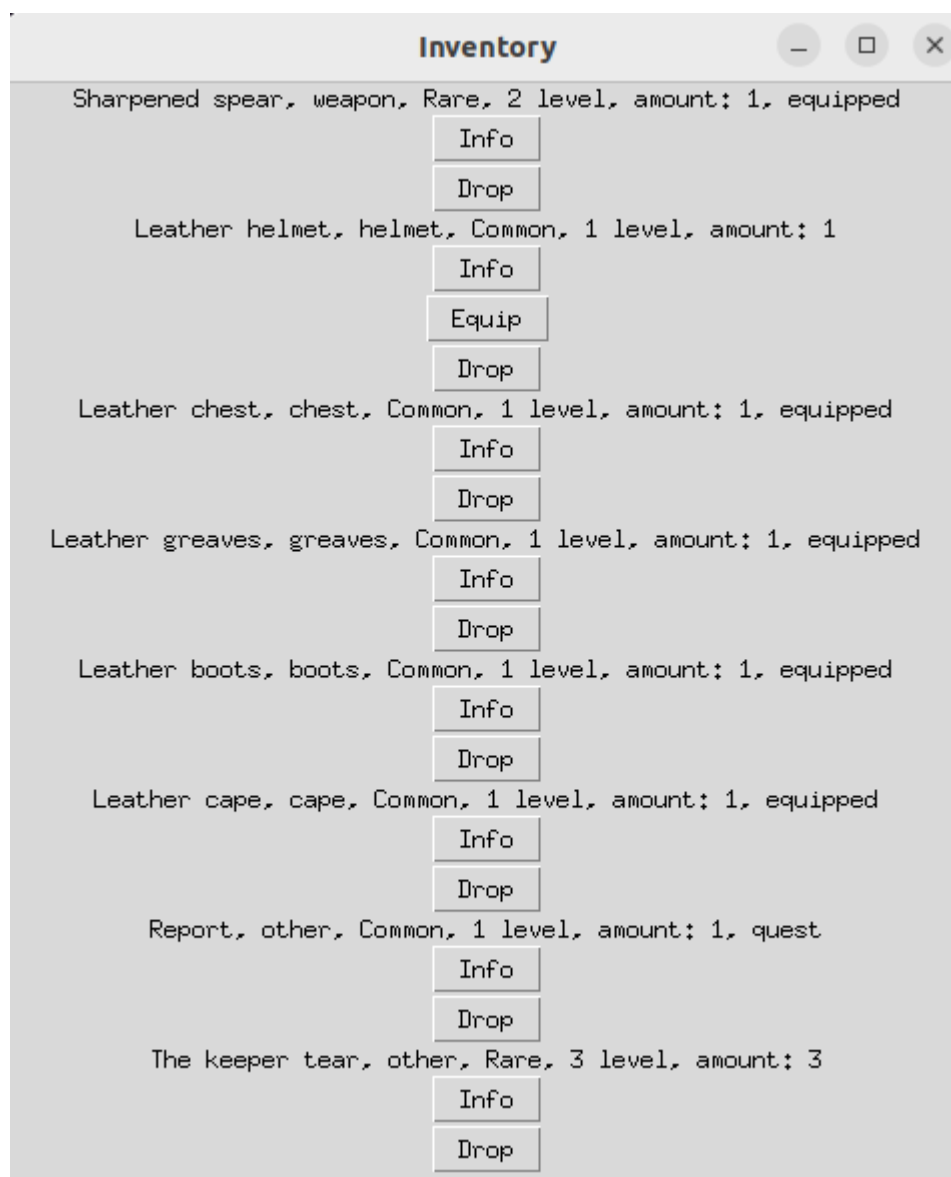


Рисунок 12 - Окно инвентаря.

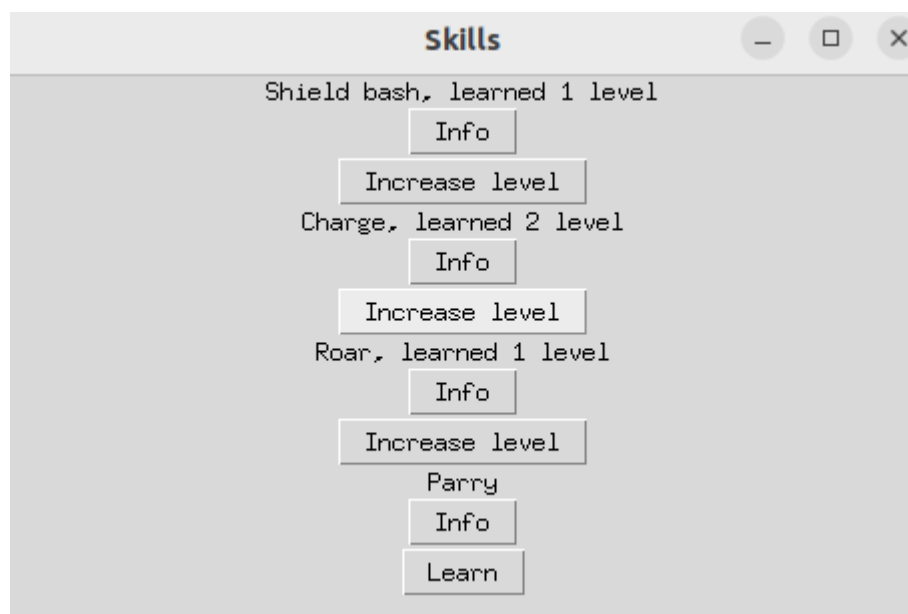


Рисунок 13 - Окно способностей.

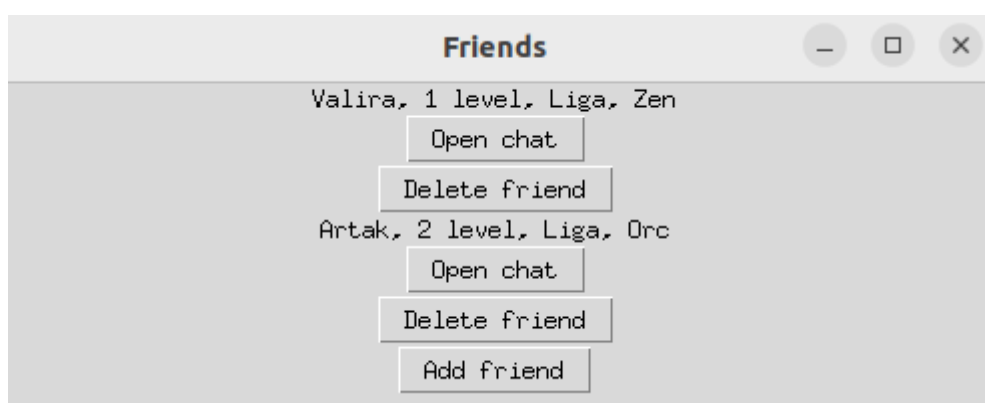


Рисунок 14 - Окно списка друзей.

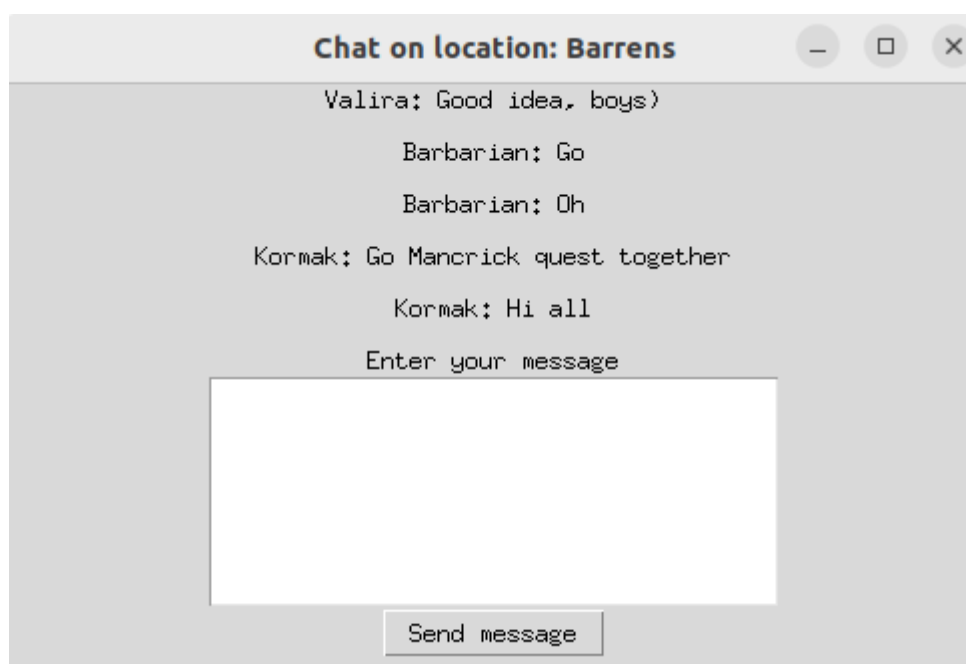


Рисунок 15 - Окно чата локации.

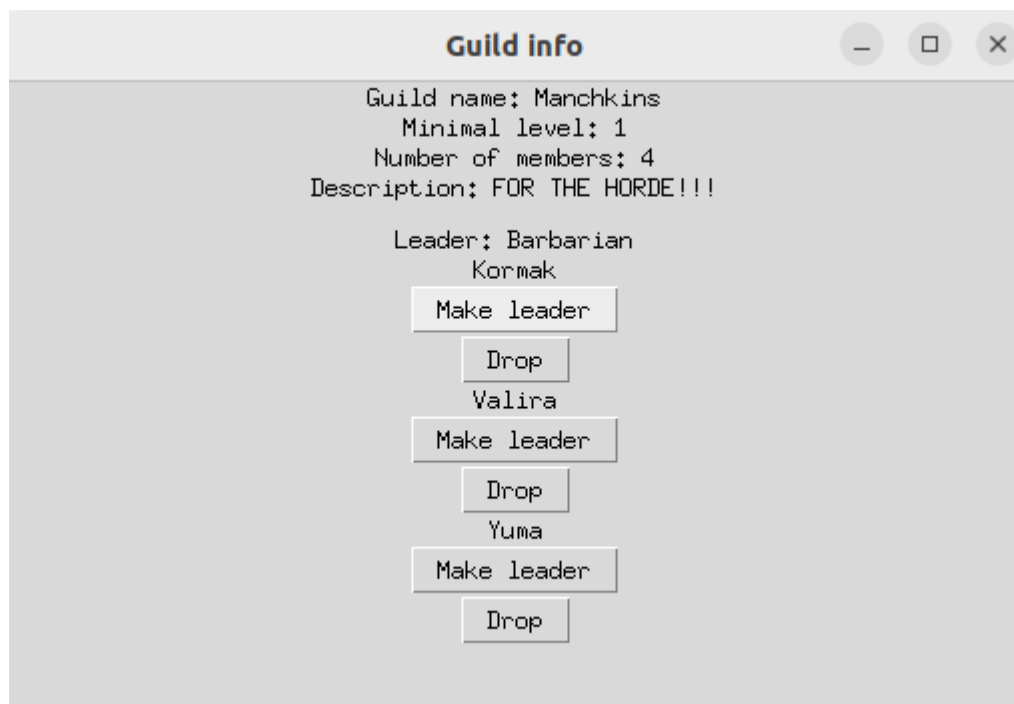


Рисунок 16 - Окно гильдии (взгляд со стороны лидера).

5. Тестирование

Во время разработки клиент-серверного приложения производилось тестирование на проверку работоспособности всех его частей. Множество раз проверялись и модифицировались запросы на получение необходимой информации, а также на обновление этой информации.

Особый упор был на то, чтобы в рабочих условиях была успешно протестирована работа всех триггеров.

Также была проверена работа приложения при одновременной деятельности нескольких персонажей (Рисунок 17).

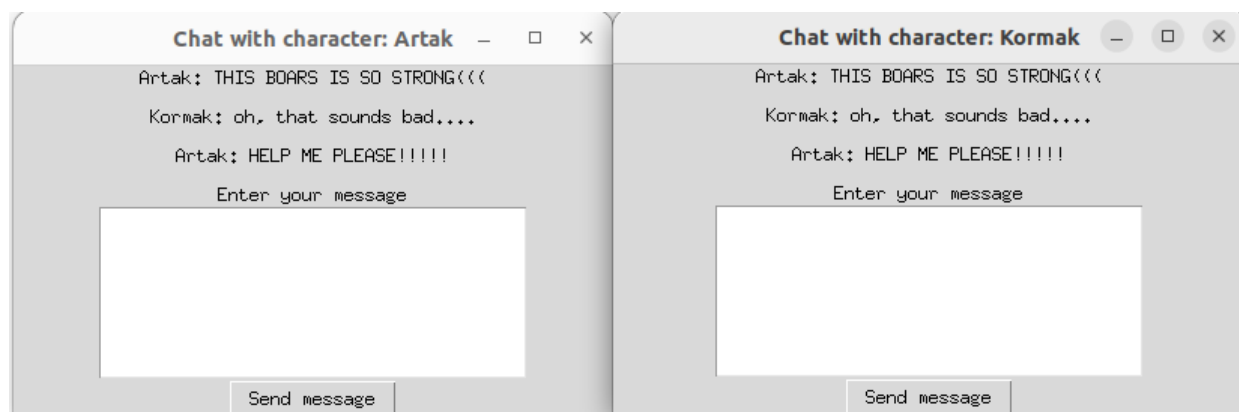


Рисунок 17 - Одновременная переписка двух персонажей.

На этапе финальной версии приложения был успешно реализован и проверен весь необходимый функционал.

ЗАКЛЮЧЕНИЕ

В итоге выполнения данной курсовой работы была изучена система управления базами данных PostgreSQL, а также были получены навыки написания клиент-серверного приложения на языке Python, которое подключается к базе данных и взаимодействует с ней.

Разработанное клиент-серверное приложение позволяет в полной мере протестировать всю работу базы данных и проверить работоспособность сложных запросов к этой базе.

В результате, улучшением на будущее будет являться подключение этой базы данных к настоящему серверу ролевой многопользовательской онлайн игры и выполнение протестированных запросов в реальных условиях.

СПИСОК ЛИТЕРАТУРЫ

1. PostgreSQL: Documentation [Электронный ресурс] - URL: <https://www.postgresql.org/docs> (Дата обращения 10.06.2023)
2. Python: Documentation [Электронный ресурс] - URL: <https://docs.python.org/3/> (Дата обращения 10.06.2023)
3. Tkinter - Python interface to Tcl/Tk [Электронный ресурс] - URL: <https://docs.python.org/3/library/tkinter.html> (Дата обращения 10.06.2023)
4. Psycopg – PostgreSQL database adapter for Python [Электронный ресурс] - URL: <https://www.psycopg.org/docs/> (Дата обращения 10.06.2023)
5. Socket — Low-level networking interface [Электронный ресурс] - URL: <https://docs.python.org/3/library/socket.html> (Дата обращения 10.06.2023)
6. Introducing JSON [Электронный ресурс] - URL: <https://www.json.org/json-en.html> (Дата обращения 10.06.2023)

Приложение А

Листинги серверной части

```
1.  player_id = None
2.  character_id = None
3.  db_conn = psycopg2.connect(user='postgres',
4.                               host="localhost",
5.                               password='2128506Vc',
6.                               port="5432",
7.                               dbname="allods")
8.  cursor = db_conn.cursor()
```

Листинг А.1 - Подключение к базе данных.

```
1.  def try_change_query(query, cursor, db_conn, player_socket):
2.      try:
3.          cursor.execute(query)
4.          db_conn.commit()
5.          message = {
6.              'status': True
7.          }
8.          send_message(message, player_socket)
9.      except (Exception, Error) as error:
10.         db_conn.rollback()
11.         message = {
12.             "status": False,
13.             "reason": str(error)
14.         }
15.         send_message(message, player_socket)
```

Листинг А.2 - функция, которая исполняет запрос на изменение таблицы.

```
1.  query_learned = f"select skills.skill_id, skill_name, description,
2.                      cooldown, class, manacost, character_id, skill_level " \
3.  f"from skills left join skill_learned sl on skills.skill_id =
4.  sl.skill_id " \
5.  f"and sl.character_id = {character_id} where skills.class = {cl}"
```

Листинг А.3 - Подготовка запроса, который вернет все способности для класса игрока и информацию о статусе их изучения.

```
1. query_weapon = f"SELECT i.Item_ID, i.Item_name, i.Item_level,  
    i.Rarity, i.Description, i.Item_type, " \  
2. f"w.Damage, w.Attack_speed, w.Attack_range, w.Weapon_type,  
    iii.is_equipped, iii.amount " \  
3. f"FROM item_in_inventory iii " \  
4. f"JOIN items i ON iii.Item_ID = i.Item_ID " \  
5. f"JOIN weapon w ON i.Item_ID = w.Item_ID " \  
6. f"WHERE iii.Character_ID = {character_id}"
```

Листинг А.4 - Подготовка запроса, который вернет все оружия в инвентаре персонажа.

```
1. query = f"SELECT m.Sending_time, sender.Character_Name AS  
    Sender_Name, m.Text, receiver.Character_Name AS Receiver_Name " \  
2. f"FROM messages m " \  
3. f"JOIN characters AS sender ON m.Sender_ID = sender.Character_ID "  
    \  
4. f"JOIN to_other AS t1 ON m.Message_ID = t1.Message_ID " \  
5. f"JOIN characters AS receiver ON t1.Receiver_ID =  
    receiver.Character_ID " \  
6. f"WHERE (" \  
7. f"(m.Sender_ID = {id} AND t1.Receiver_ID = {character_id}) " \  
8. f"OR" \  
9. f"(m.Sender_ID = {character_id} AND t1.Receiver_ID = {id})) " \  
10.f"ORDER BY m.Sending_time DESC " \  
11.f"LIMIT {k}"
```

Листинг А.5 - Подготовка запроса, который вернет самые поздние k сообщений между двумя игроками.

```
1. query = f"SELECT CASE WHEN Character_ID = {character_id} THEN  
    Friend_ID ELSE Character_ID END " \  
2. f"FROM characters_friends WHERE Character_ID = {character_id} OR  
    Friend_ID = {character_id}"
```

Листинг А.6 - Подготовка запроса, который вернет список друзей персонажа.

