Содержание

[**Введение** 2](#_Toc184217400)

[**1 Аналитическая часть** 3](#_Toc184217401)

[**1.1 Анализ предметной области** 3](#_Toc184217402)

[**1.2 Постановка задачи** 4](#_Toc184217403)

[2. **Разработка логической модели базы данных** 6](#_Toc184217404)

[**2.1. Построение диаграммы «сущность-связь» в нотации П.Чена.** 6](#_Toc184217405)

[**2.2. Построение модели основанной на ключах (Key Based model, KB)** 8](#_Toc184217406)

[**2.3. Построение полной атрибутивной модели в нотации IDEF1X. Нормализация отношений до третьей нормальной формы** 10](#_Toc184217407)

[**3** **Разработка физической модели базы данных** 21](#_Toc184217408)

[**3.1 Выбор аппаратной и программной платформы для реализации БД** 21](#_Toc184217409)

[**3.2 Реализация базы данных** 21](#_Toc184217410)

[**3.3 Тестирование базы данных** 30](#_Toc184217411)

[**3.4 Разграничение прав доступа** 36](#_Toc184217412)

[**4. Разработка клиентского приложения** 37](#_Toc184217413)

[**4.1. Обоснование выбора языка программирования** 37](#_Toc184217414)

[**4.2. Разработка интерфейса пользователя** 37](#_Toc184217415)

[**4.3. Алгоритм работы каждого из модулей** 37](#_Toc184217416)

[**4.4. Тестирование работы приложения** 37](#_Toc184217417)

[**Приложения** 38](#_Toc184217418)

**Введение**

Документ представляет анализ обработки табличной информации, внесённой в базу данных. Информация сфокусирована на добыче ресурсов с астероидов. Роботы-шахтёры, которые добывают полезные ископаемые на астероиде, складывают ресурсы в хранилище и поручают роботам-доставщикам доставку на Землю или космическую базу. В работе освещаются технологии обработки больших объемов данных, рассматриваются современные подходы к анализу данных в базе данных, моделированию диаграмм перед созданием физической модели базы данных. Кроме того, документ исследует вопросы, связанные с эффективностью хранения и обработки данных, а также их интеграцией в существующие системы управления ресурсами. Особое внимание уделяется нормализации данных, что позволяет минимизировать дублирование информации и повысить целостность базы данных. Анализируются различные методы оптимизации запросов к базе данных, что способствует ускорению обработки информации и улучшению производительности системы в целом.

Таким образом, данный анализ предоставляет комплексный подход к разработке и внедрению системы управления данными, необходимыми для успешной добычи и транспортировки ресурсов с астероидов, что открывает новые горизонты в области космической экономики и технологий.

**1 Аналитическая часть**

**1.1 Анализ предметной области**

База данных роботов-шахтёров это 2д симуляция экспедиций на различные астероиды с целью добычи полезных ископаемых.

Пользователю выделяется робот, здание “крафтер” и кредит – сумма, затраченная на запуск экспедиции, которую нужно погасить за определённый срок, чтобы окупить проект.

В здание “крафтер” можно поместить различные ресурсы для автоматического создания других зданий/компонентов по рецепту. Основные здания: маркет, up, склад, робот-доставщик.

В здании “маркет” пользователь имеет возможность отправить экспедицию на Землю с определённым шансом успешной доставки, который зависит от качества робота-доставщика, либо запросить доставку на астероид. Любой запуск доставки требует определённое количество полезных ископаемых или валюты, однако экспедиции прибыльны из-за недостатка ресурсов на Земле.

Здание “up” может увеличить добычу робота или другие его характеристики путём улучшения или добавления отдельных модулей. Улучшение также происходит автоматически и требует только ресурсы.

Все здания хранятся у робота в инвентаре и занимают определённую грузоподъёмность.

Роботов-шахтёров может быть несколько, они могут быть созданы в здании “крафтер”, для работы со множеством из них можно оснастить робота модулем “программатор”, который позволяет прописать несложную логику автоматической добычи.

Существуют кристаллы и породы. Кристаллы – полезные ископаемые, с целью добычи которых и проводится экспедиция.

Исходя из аналитической части можно определить следующие сущности в бд:

Робот-шахтёр

Робот-доставщик

Здания (Крафтер, маркет, Up)

Кристаллы и породы

Валюта (в т.ч и кристаллы)

Программатор

**1.2 Постановка задачи**

Существует несколько основных сущностей: робот-шахтёр (robot), магазин (market), здание для улучшений (UP), кристалл (crystal). Робот копает кристаллы, продаёт, улучшает навыки (добыча, хп, хилл и др.), может ставить блоки (если умение разблокировано)

Робот-шахтёр имеет несколько полей: id, name, slotsId, currentCrystalsId, xpos, ypos. Id пк, отвечающий за номер робота, name имя робота, statsId и storableCrystalsId – 2 отдельные таблицы, связанные с роботом-шахтёром. В statsId отображается прокачка робота, в storableCrystalsId – количество определённых кристаллов, которые хранятся у робота в данный момент.

slotsId – имеет 2 поля Id и slotID, 2е поле принимает значение 0, если в слоте нету умения и slotID сущности slot, если умение установлено.

slot - имеет поля id, skillId, lvl, value

skillTable – таблица со всеми умениями. Поля: id, name, lvl, value. Значения в бд вносятся с помощью цикла через клиентскую часть

currentCrystalsId – сущность с текущим грузом всех кристаллов. Имеет поля: id, gcount, bcount, rcount….

Магазин даёт возможность купить или продать кристаллы, в разных магазинах разные цены. Имеет поля: id, buyId, sellId, xpos, ypos

buyTable таблица с полями: id, gprice, bprice, rprice…

sellTable таблица с полями: id, gprice, bprice, rprice…

Кристалл – сущность добываемого кристалла. Имеет поля: id, durability, crystalType, isX, xpos, ypos.

2. **Разработка логической модели базы данных**

**2.1. Построение диаграммы «сущность-связь» в нотации П.Чена.**

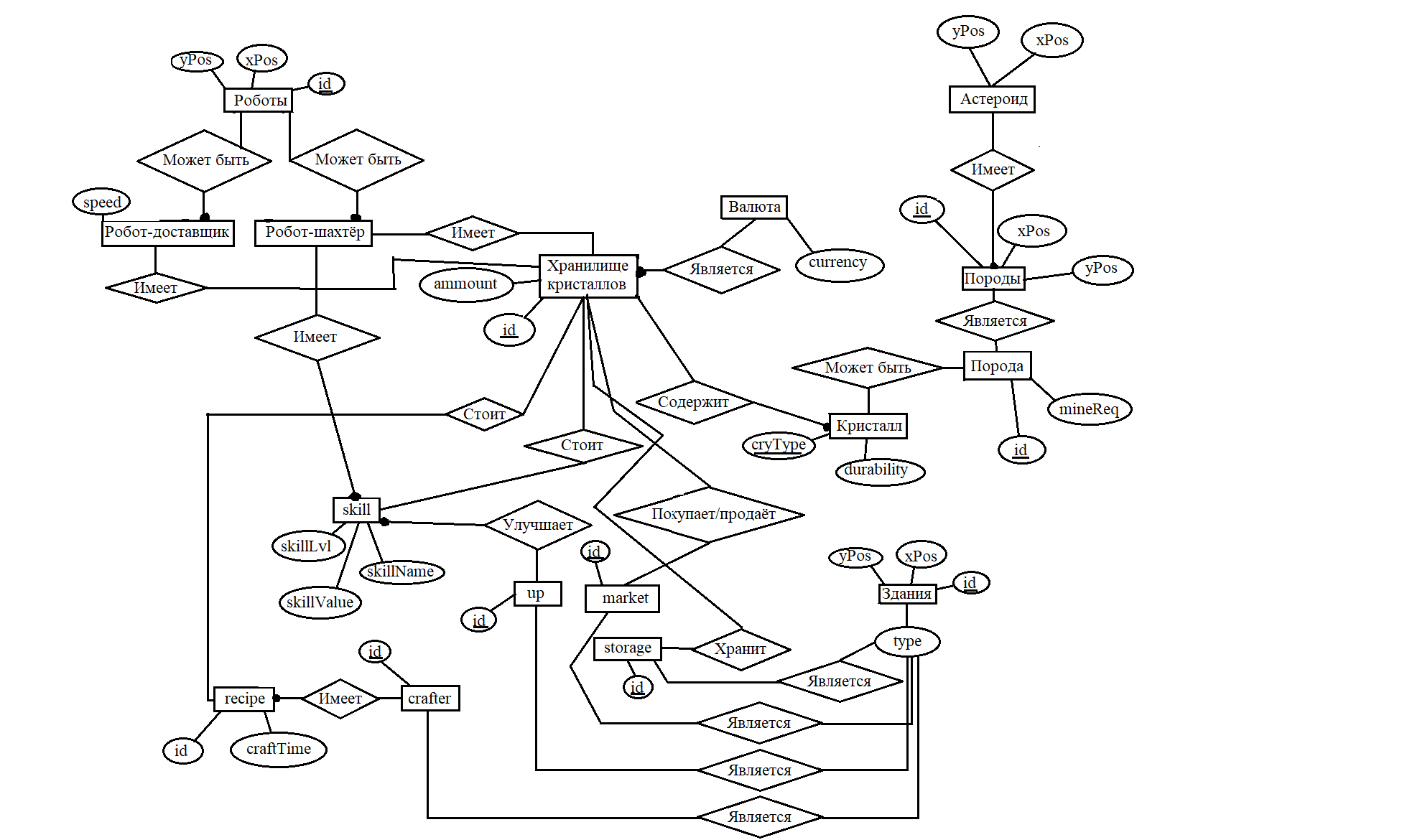


Рисунок 1 – ERD диаграмма по нотации П.Чена

Первая ERD модель имела следующий вид (рис. 1). Позже в неё были внесены изменения и убраны такие сущности, как склад, крафтер, маркет и up, т.к весь логический функционал этих зданий можно реализовать и без потребности в дополнительных сущностях в базе данных, достаточно лишь атрибута type в сущности здания.

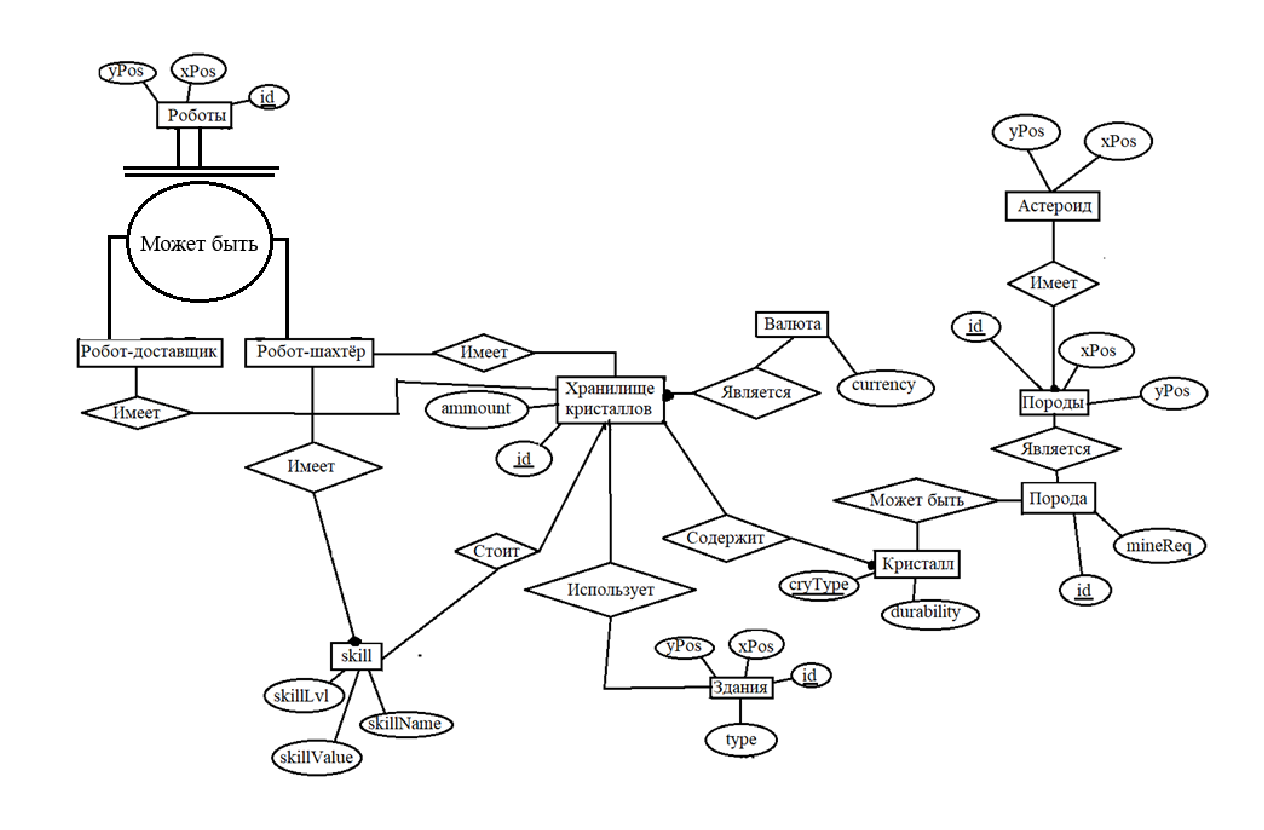


Рисунок 2 – ERD диаграмма (итоговая версия)

ERD диаграмма сущность-связь по нотации п.Чена имеет таблицы, которые отображаются в виде прямоугольника, атрибуты, отображаемые, как ромб и саму связь, отображаемую в виде глагола внутри ромба. Атрибут является первичным ключом, если имеет подчёркивание снизу. Диаграмма помогает понять взаимосвязи в будущей проектируемой таблице. Исходя из аналитической части, можно определить сущности, а также глаголы, используемые для связи между сущностями. Таблицы в диаграмме не имеют связи многие ко многим, что позволяет пропустить этап приведения таблицы с использованием 1 ко многим.

**2.2. Построение модели основанной на ключах (Key Based model, KB)**

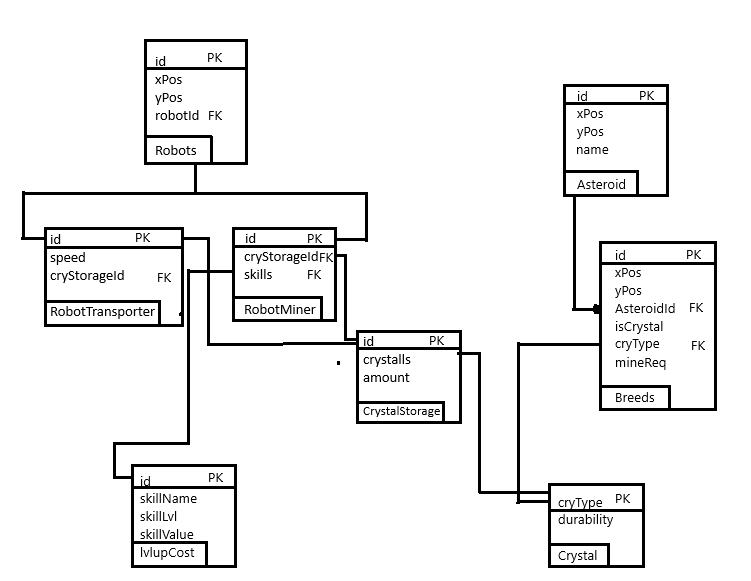


Рисунок 3 – KeyBasedModel – A0

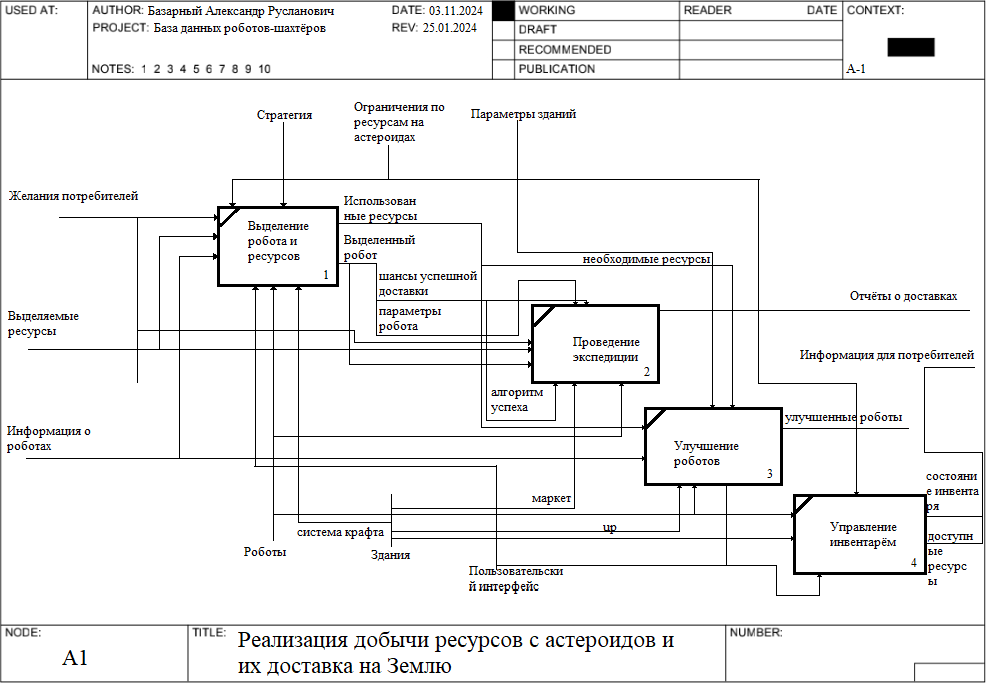


Рисунок 4 – KeyBasedModel – A1

KeyBasedModel – модель основанная на ключах. Обычно имеет 2 и более страницы. У каждого блока есть 4 входа: стрелочки снизу, сверху и слева – входные данные, справа – данные на выходе.

Механизмы (стрелочка снизу вверх) – это то, что используется для проведения необходимой работы

Контроль (стрелочка сверху вниз) – это механизмы управления (положения, правила, инструкции…)

Входящие – ставят определённую задачу

Исходящие – результат деятельности

**2.3. Построение полной атрибутивной модели в нотации IDEF1X. Нормализация отношений до третьей нормальной формы**

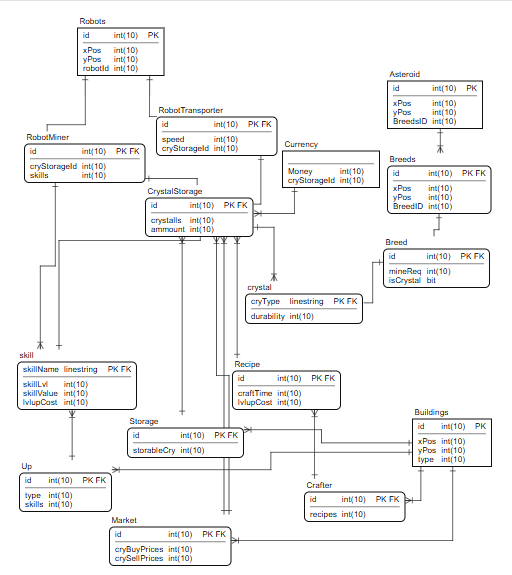


Рисунок 5 – диаграмма IDEF1X (до нормализации )

В IDEF1 также имелись сущности зданий, которые позже были удалены.

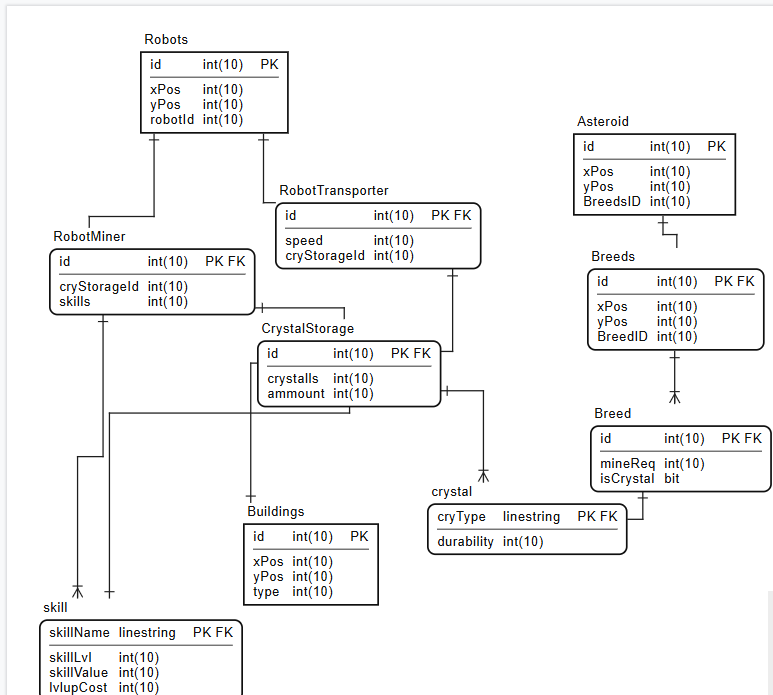


Рисунок 6 – Итоговая диаграмма IDEF1X (до нормализации)

1НФ: Диаграмма IDEF1X на рис.6 уже является атомарной. Атрибуты являются простыми и принимают только скалярные значения. Разбиение атрибута на составные части не требуется.

2НФ: Для того, чтобы доказать, что диаграмма имеет 2 НФ необходимо просмотреть все связи у её атрибутов. Атрибут должен ссылаться только на первичный ключ.

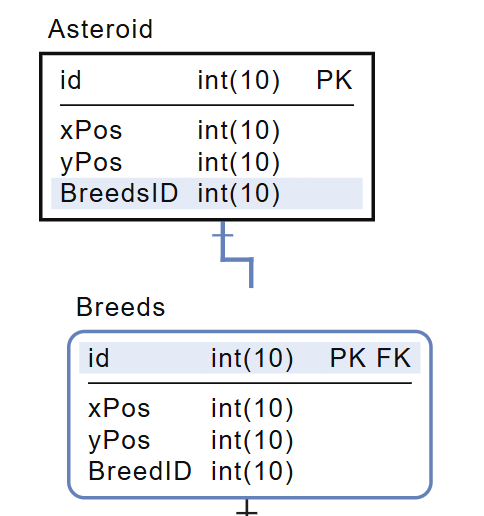


Рисунок 7 – Asteroid-Breeds (1к1)

На рис.7 атрибут BreedsId ссылается на id первичного ключа у сущности Breeds.

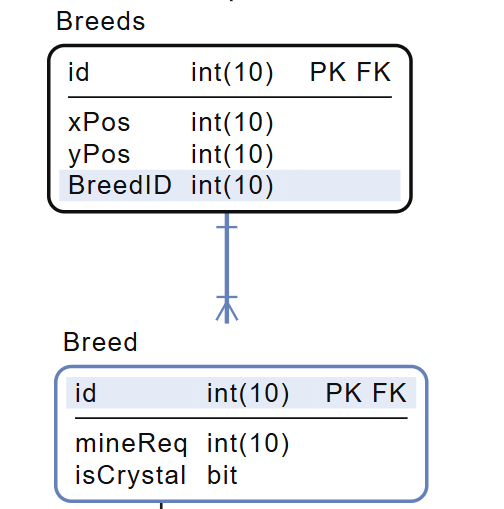


Рисунок 8 – Breeds-Breed (1 ко многим)

На рис 8 атрибут BreedID ссылается на id первичного ключа у сущности Breed. При проведении нормализации для 2нф было замечено, что можно произвести часть нормализации до 3нф: убрать сущность Breeds, изменить таблицу Breed и построить связь 1 ко многим у таблицы Breed и Asteroid.

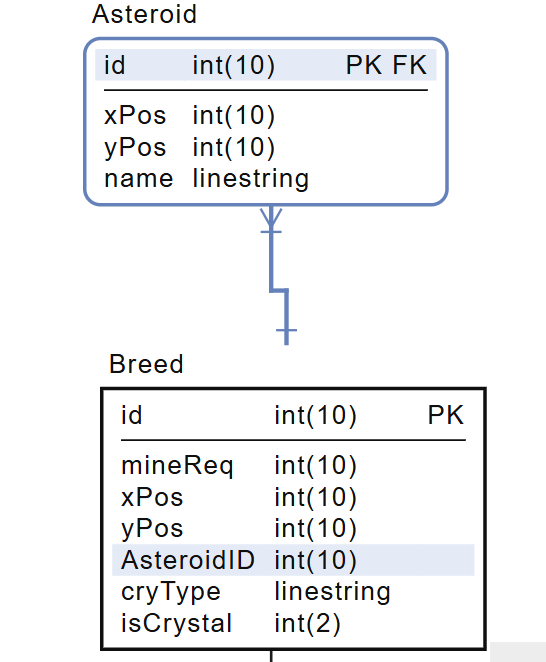


Рисунок 9 – упрощение таблицы. Asteroid-Breed (1 ко многим)

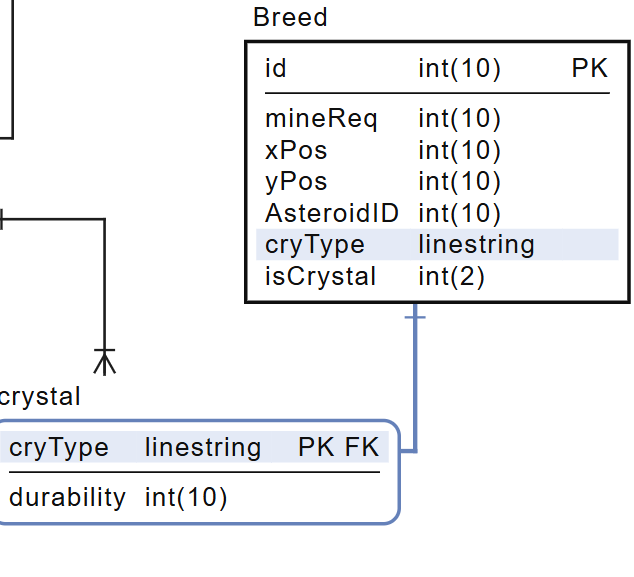


Рисунок 10 – Breed-Crystal (1к1)

На рис.10 атрибут cryType ссылается на cryType первичного ключа у сущности crystal (при условии, что isCrystal !=0, иначе null)

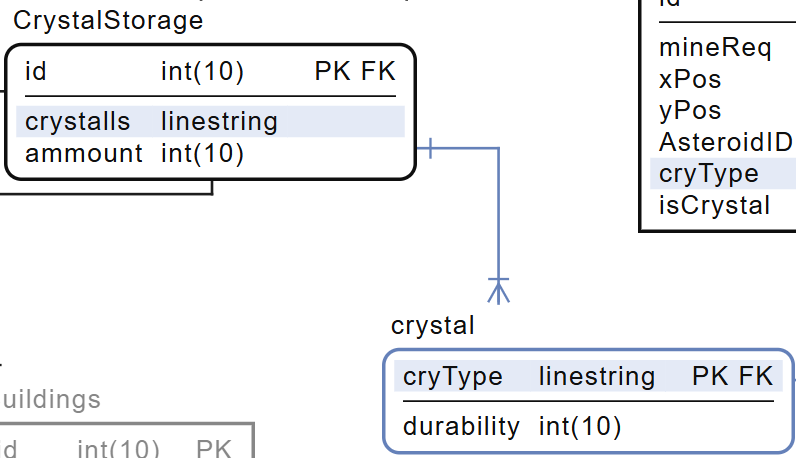


Рисунок 11 – Crystal-CrystalStorage

На рис.11 атрибут crystals ссылается на cryType первичного ключа у сущности crystal (при условии, что isCrystal !=0, иначе null).

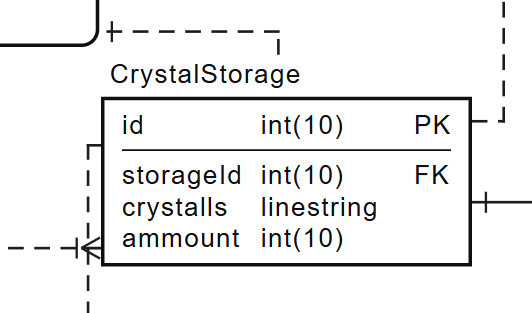


Рисунок 12 – проблема 2нф

Возникла проблема на рис. 12. Данные в CrystalStorage имеют следующий вид: создаются записи под id и storageId, однако таблицы ссылаются на storageId => идёт нарушение 2нф.

|  |  |  |  |
| --- | --- | --- | --- |
| id | storageId | cryType | ammount |
| 1 | 1 | A | 10 |
| 2 | 1 | B | 20 |
| 3 | 2 | A | 15 |

Таблица 1 – Пример записи данных

Решение: создание таблицы storage, storageId ссылается на id у таблицы storage.

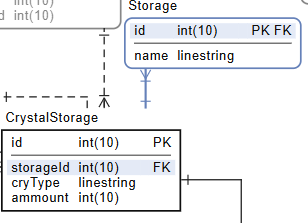


Рисунок 13 – Решение проблемы

Проблема решена. Пример данных:

|  |  |
| --- | --- |
| id | name |
| 1 | S1 |
| 2 | S2 |

Таблица 2 – Данные Storage

|  |  |  |  |
| --- | --- | --- | --- |
| id | storageId | cryType | ammount |
| 1 | 1 | A | 10 |
| 2 | 1 | B | 10 |
| 3 | 1 | C | 20 |
| 4 | 2 | A | 14 |
| 5 | 2 | B | 20 |

Таблица 3 – Данные CrystalStorage

Все предыдущие ссылки на crystalStorage следует заменить на таблицу Storage.

Дальнейшие таблицы также также будут связаны с пк по аналогии. => после решения проблемы таблица приняла 2 нф.

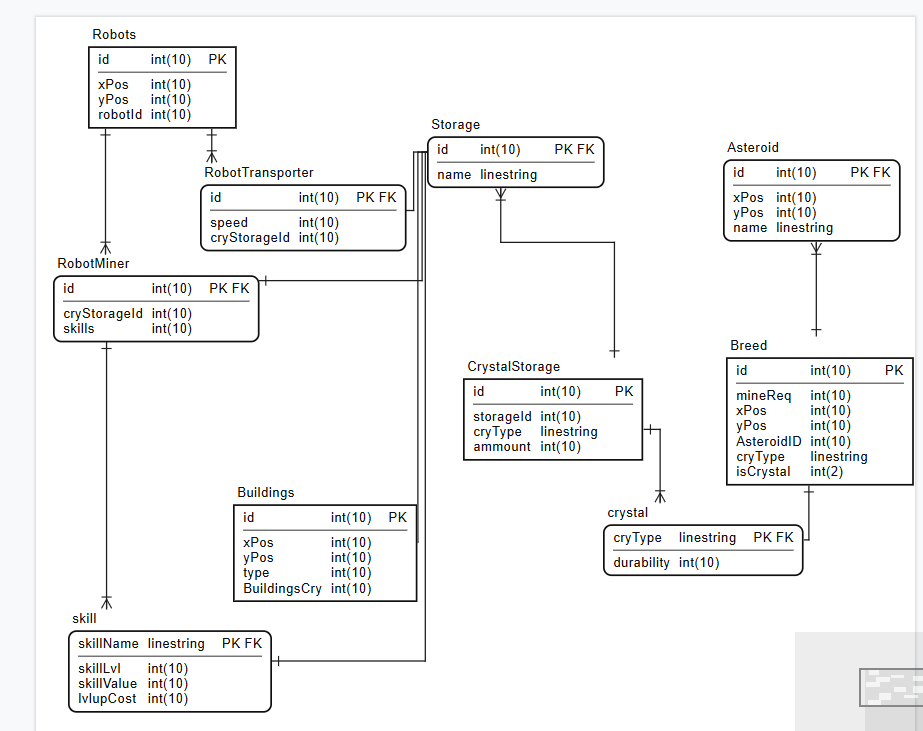


Рисунок 14 – Вид таблицы после нормализации до 2 нф

3НФ: все зависимости от пк нетранзитивны + 2НФ

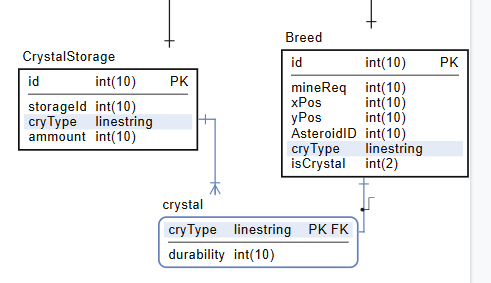


Рисунок 15 – Breed, Asteroid, Crystal

Пк cryType используется в CrystalStorage и Breed => атрибуты транзитивны.

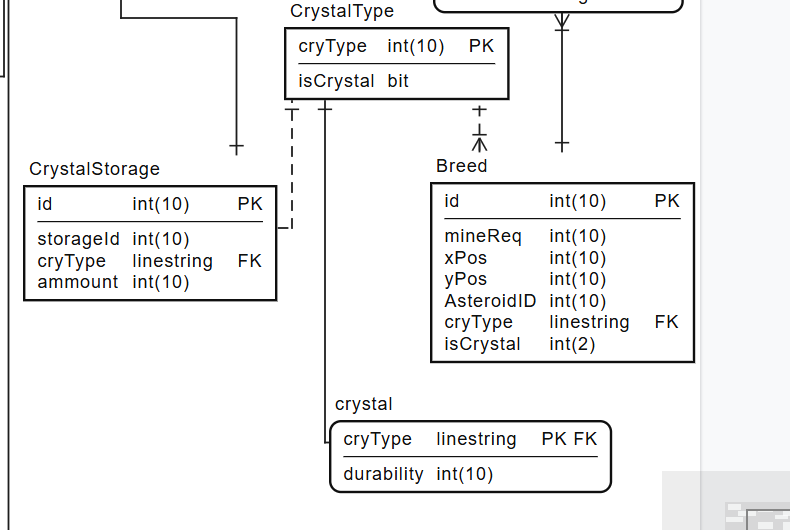


Рисунок 16 – Решение транзитивности

Атрибут cryType больше не транзитивен для Breed, CrystalStorage and Crystal

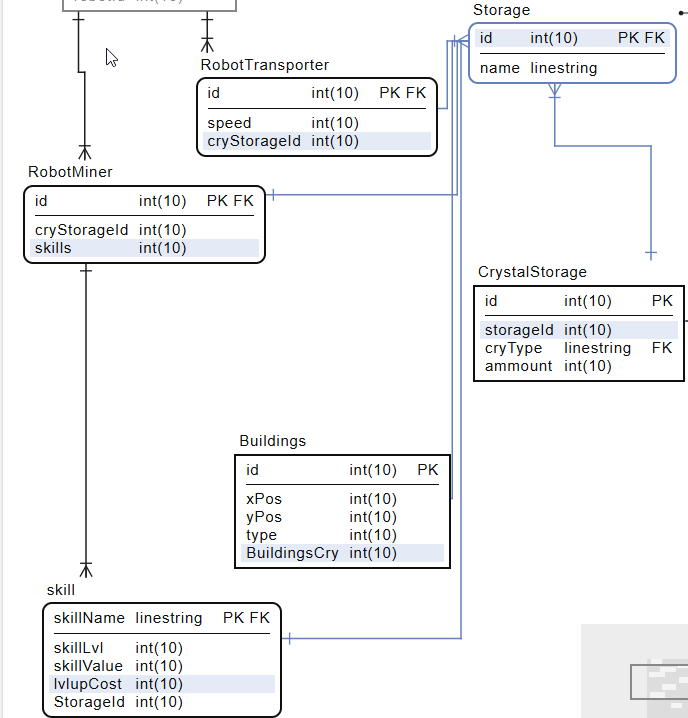


Рисунок 17 – Связи Storage

Атрибуты storageId и cryStorageId у сущностей RobotMiner и skill транзитивны.

Изначально планировалось решить проблему по аналогии с рис. 16. По итогу было принято решение удалить связь FK StorageId references Storage(id), т.к каждый робот имеет свой личный StorageId из которого и тратятся ресурсы для прокачки каждого отдельного робота.

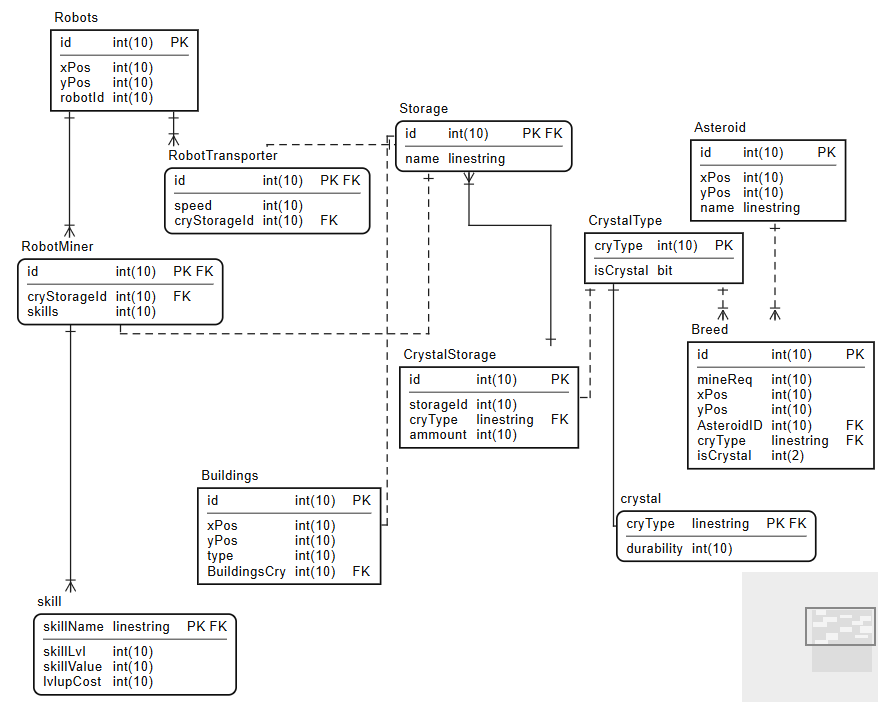


Рисунок 18 – IDEFX1 в 3НФ

Изначально на предыдущих рисунках возникла путаница с созданием FK на сайте для создания IDEFX1 диаграмм (создание происходило наоборот). Сейчас все FK изображены верно, таблица приведена к 3 нф, в дальнейшей нормализации не нуждается. Можно создавать базу данных

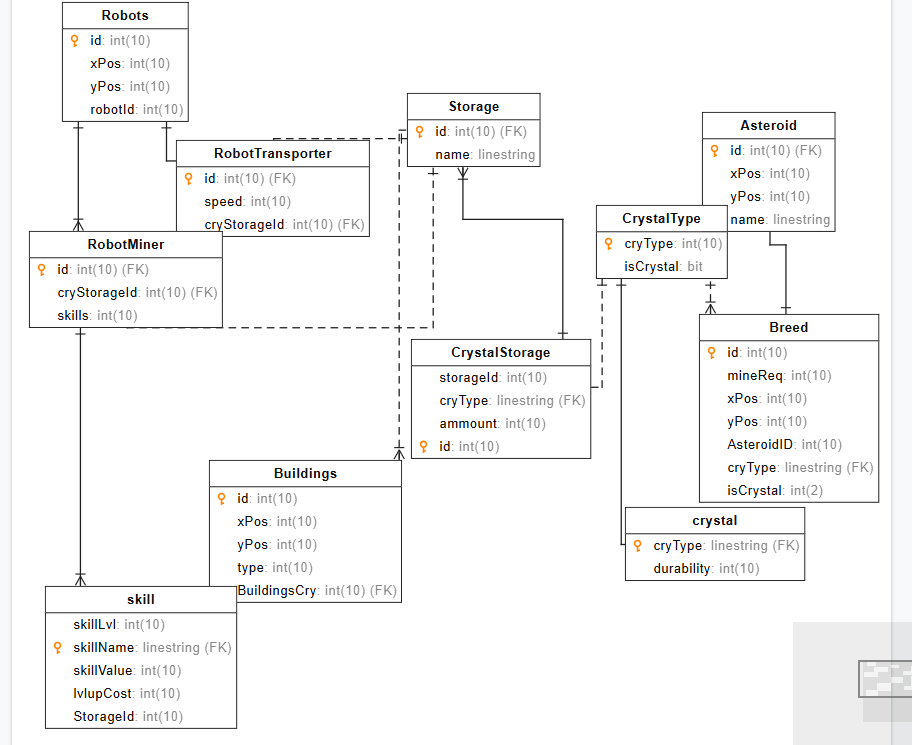


Рисунок 19 – Классическая нотация

**3** **Разработка физической модели базы данных**

**3.1 Выбор аппаратной и программной платформы для реализации БД**

Для разработки физической модели базы данных было использовано приложение SSMS (Sql Server Mangment Studio). Оно позволяет работать с реляционными базами данных, имеет простой интерфейс. Единственный минус – сложный этап подключения:

Этап 1: Панель управления – администрирование – ODBS data source – создание бд

Этап 2: Панель управления – администрирование – Службы – поиск ODBS и включение службы.

После перезахода на пк требуется повторно пройти этап 2. Иногда возникают сложности при создании в принципе.

Альтернатива – PostrgreSql, возможно Oracle, однако из-за более привычного интерфейса было выбрано приложение SSMS.

**3.2 Реализация базы данных**

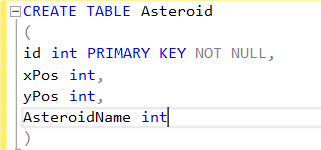


Рисунок 20 – создание таблиц

Во избежание путаницы было принято решение создать таблицы без связей, а после с помощью ALTER TABLE внести их. При создании таблиц использовался рис. 19 – модель в 3 нормальной форме, атрибуты name были изменены из-за конфликта со внутренними переменными.

После завершения работы по воссозданию таблиц согласно модели, на рис.19 были написаны триггеры для ограничения доступа робота-транспортёра к скилам, так как он их не должен иметь.

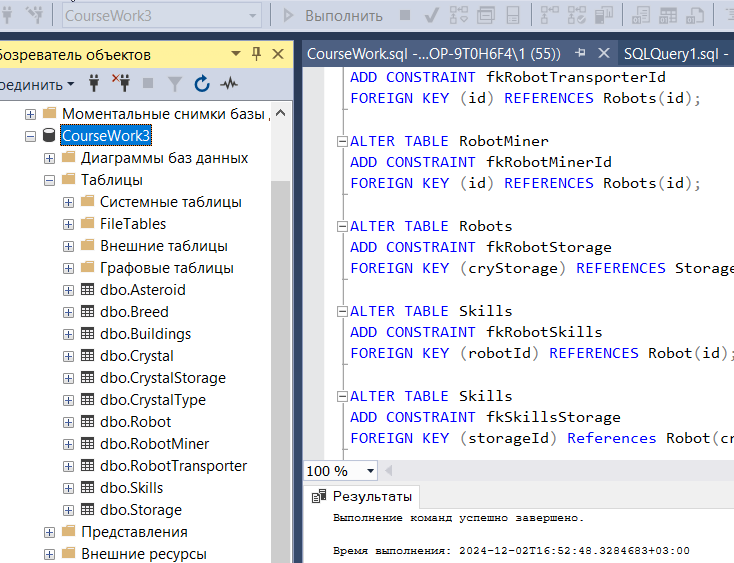


Рисунок 21 – Создание таблиц

На рис.21 показано успешное выполнение кода при создании таблиц, листинг приведён в приложении А

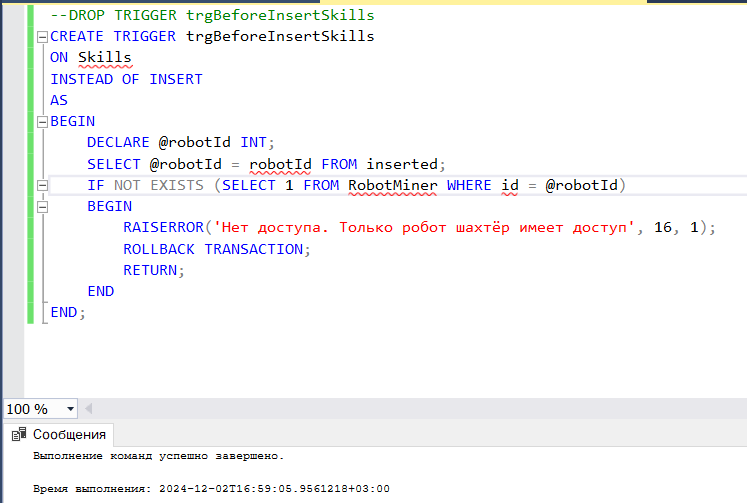


Рисунок 22 – Успешное создание триггера

На рисунке 22 создаётся триггер для ограничения внесения данных. Если тип робота равен 1 (что означает робота-транспортировщика), то выводится ошибка и транзакция откатывается.

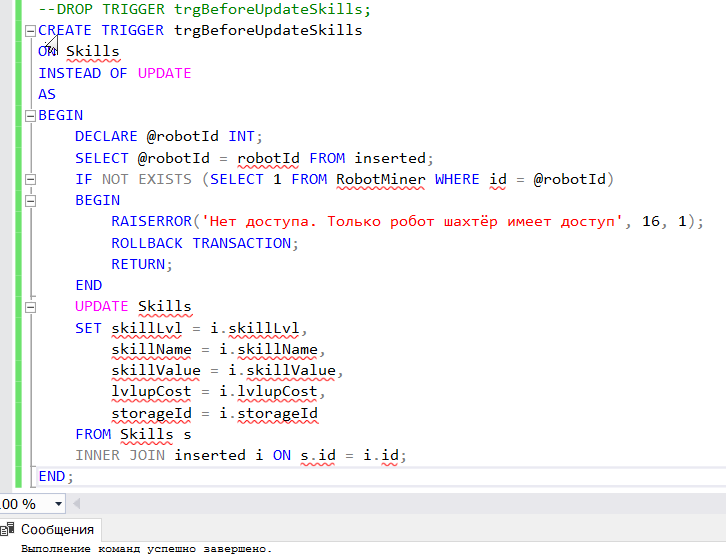


Рисунок 23 – Создание триггера обновления

На рис. 23 представлен триггер для обновления, чтобы данные не были изменены роботом-транспортировщиком.

Реализация генерация данных в Breeds + Asteroid при использовании SQL сложна, создать генератор непосредственно в SQL времязатратно, т.к существует несколько правил и обозначить их всех будет легче на языке высокоуровневого программирования, в данном случая на языке программирования C#. Остальные данные можно внести непосредственно через SQL для дальнейшего тестирования запросов.

Для работы с базой данных внутри Visual Studio Community был использован пакет NUGet с пространством имён System.Data.SqlClient. Данное расширение позволяет пользователю делать запросы внутри VisualStudio и менять подключённую базу данных таким образом.

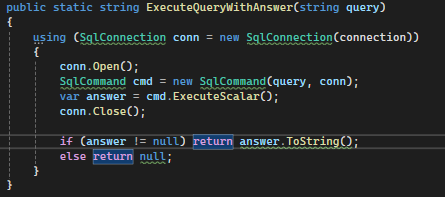


Рисунок 24 – Код для получения 1й строки

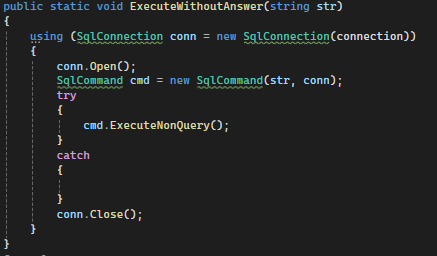


Рисунок 25 – Код для занесения запроса

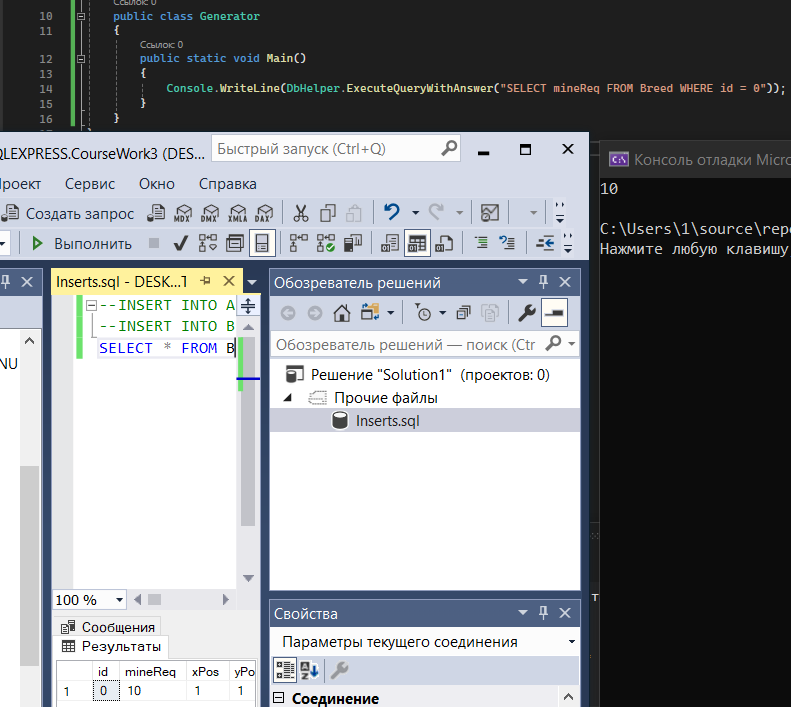


Рисунок 26 – Пример использования

Для проверки работоспособности кода были внесены некоторые данные вручную, в самом коде был получен столбец mineReq, где id был равен 0. С помощью взаимодействия VS + SQL можно создать удобный и простой в написании генератор.

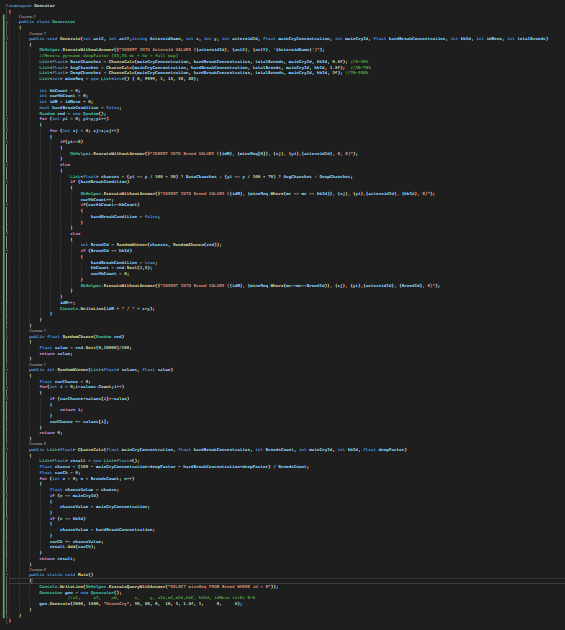


Рисунок 27 – Генератор

На рис. 27, а также в приложении Б показан код генератора, который генерирует астероид + породы для него. Породы генерируются следующим образом: при y = 0 – пустая порода, иначе происходит рандомизация.

Если координата yj меньше или равна 30%, то фактор глубины = 0.5f, меньше 70% 1.5f, иначе 3. При создании генератора указывается шанс встречи основного кристалла, который находится на астероиде + шанс встречи сложно разрушимой породы, которые умножаются на фактор глубины, а также их id для занесения в бд. Остальные породы распределяются равномерно. Если выпадает неразрушимая порода, то длина фигуры этой породы составляет от 1 до 5 ячеек (тайлов).

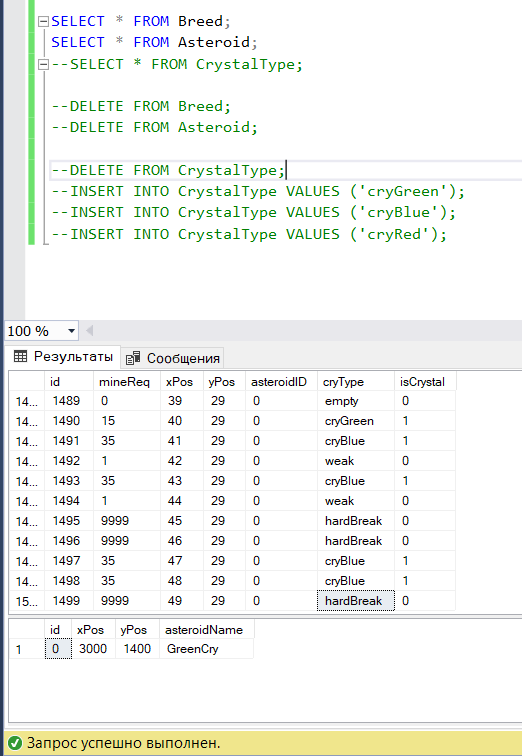


Рисунок 28 – Результат генерации

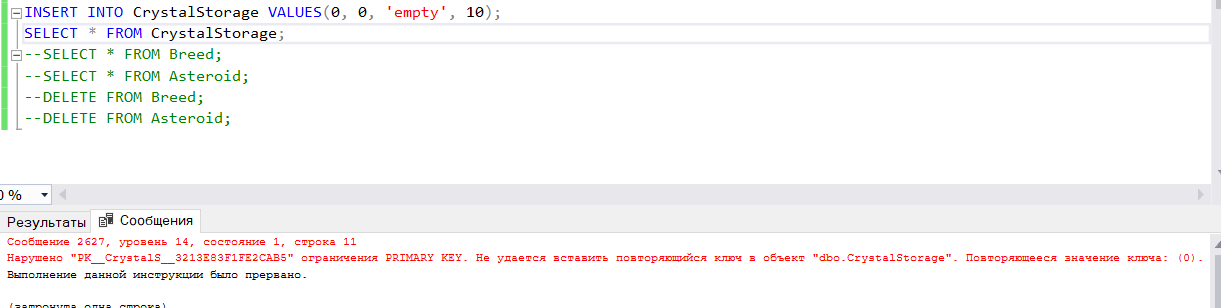


Рисунок 29 – Пример работы внешнего ключа

На рис. 29 продемонстрирован пример работы внешнего ключа, в склад кристаллов нельзя отнести нечто не являющееся кристалом

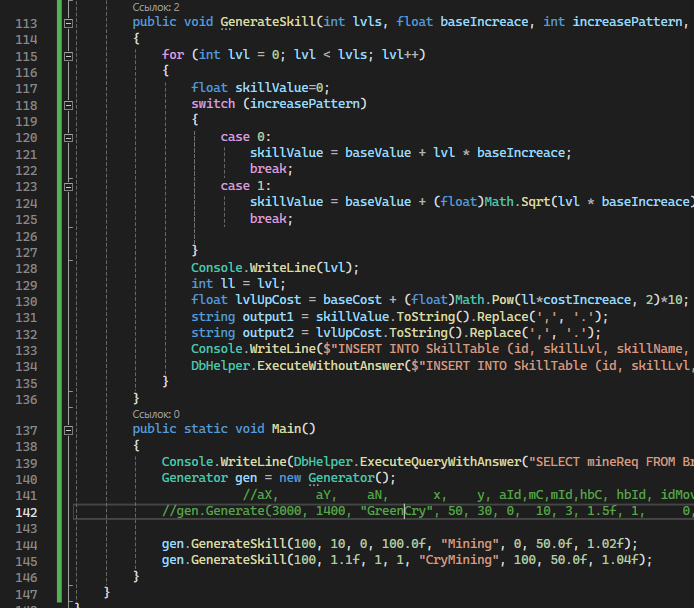


Рисунок 30 – Создание дополнительного генератора для генерации умений

На рис. 30 создаётся генератор для таблицы умений. Всего существует 2 способа генерации и в зависимости от паттерна выбирается нужный: экспоненциальный и инкрементный.

**3.3 Тестирование базы данных**

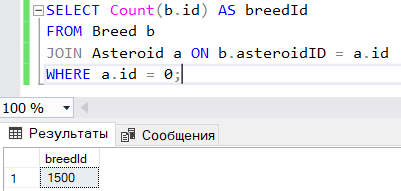


Рисунок 31 – Запрос 1

В 1 запросе просматривается количество пород на астероиде, ид которого равен 0.

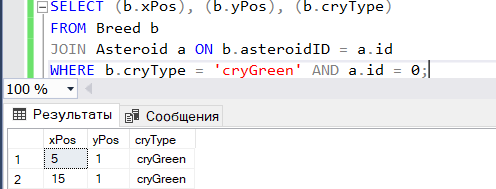


Рисунок 32 – Запрос 2

В запросе 2 идёт выборка координат + названия кристалла на астероиде с ид = 0, а также типом кристаллов равным cryGreen.

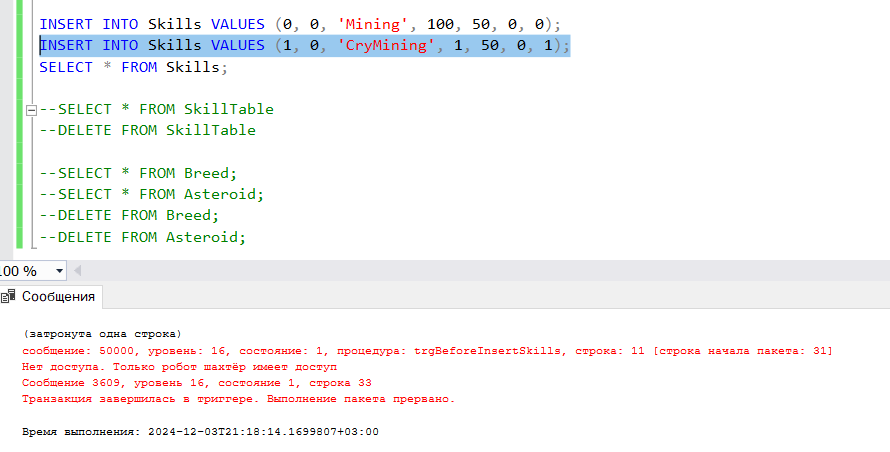


Рисунок 33 – Запрос 3

В запросе 3 проверяется работа триггера INSERT. Если id робота не является id у RobotMiner, то выводится сообщение и происходит откат транзакции.

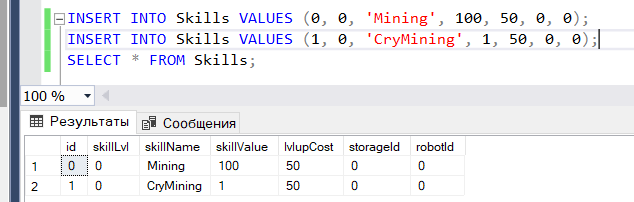


Рисунок 34 – Запрос 4

В запросе 4 продемонстрирован обычный запрос INSERT, когда триггер не конфликтует.

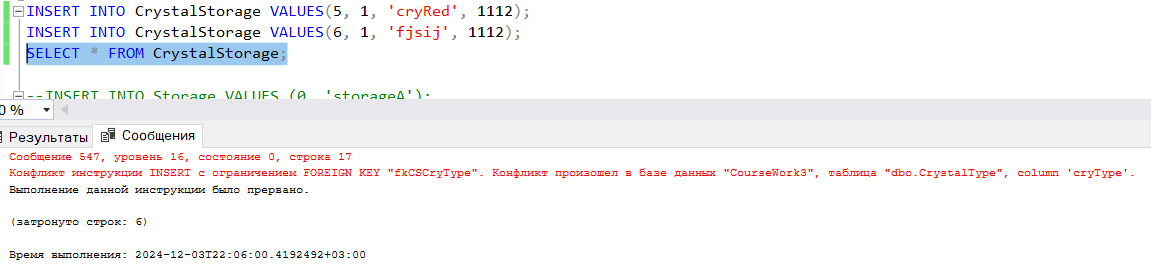


Рисунок 35 – Запрос 5

В запросе 5 проводится попытка внести не существующий в таблице CrystalTypes кристалл, а именно ‘fjsij’ в таблицу хранилища кристаллов, из-за чего происходит ошибка, т.к внешний ключ базы данных не позволяет это сделать.

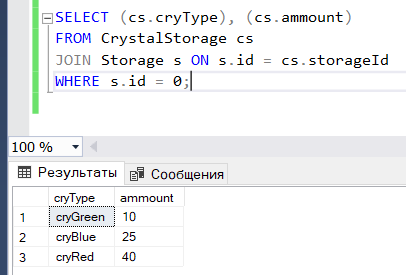


Рисунок 36 – Запрос 6

В запросе 6 происходит выборка кристаллов и их количества, в складе, id которого равен 0.

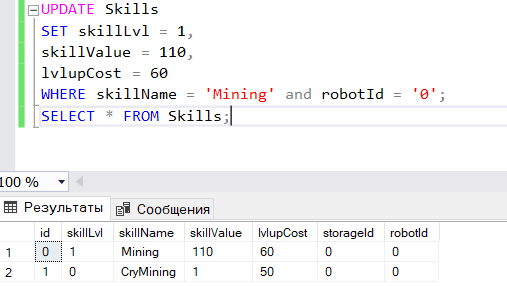


Рисунок 37 – Запрос 7

В запросе 7 происходит улучшение скила Mining у робота с id 0 при помощи метода UPDATE.

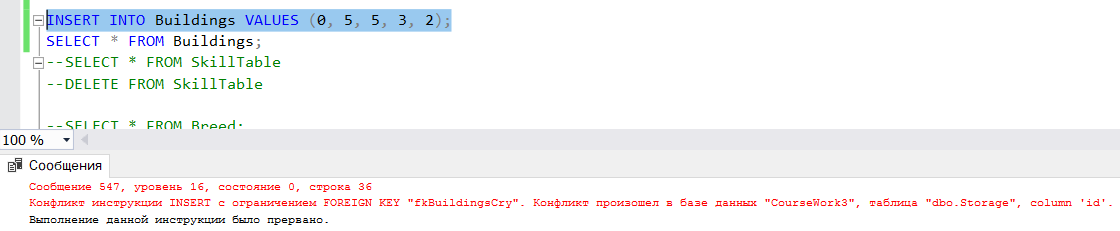


Рисунок 38 – Запрос 8

В запросе 8 идёт попытка внесения не существующего склада в здание Buildings, из-за чего происходит ошибка, т.к внешний ключ базы данных не позволяет это сделать.



Рисунок 39 – Запрос 9

Запрос 9 – циклический запрос внутри генератора, который генерирует породы.



Рисунок 40 – Запрос 10

Запрос 10 – циклический запрос внутри, генератора, который генерирует умения. Причём значения float конкретно Visual Studio выводит через плавающую запятую, а SQL принимает такую запятую, как следующий аргумент, поэтому необходимо преобразовать float в строку и заменить символ “,” на “.”

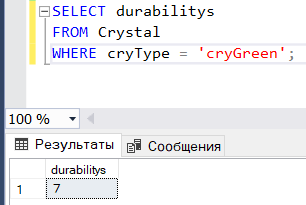


Рисунок 41 – Запрос 11

В запросе 11 происходит получение прочности зелёного кристалла

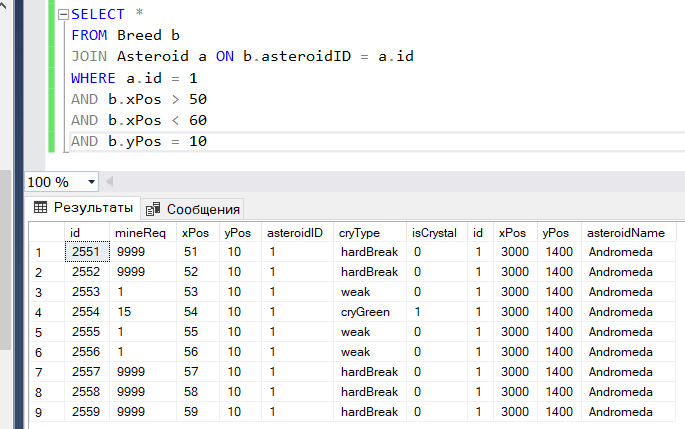


Рисунок 42 – Запрос 12

В запросе 12 происходит выборка всего в пределах координат x от 50 до 60 и y = 10 на астероиде с ид 1 (Andromeda)

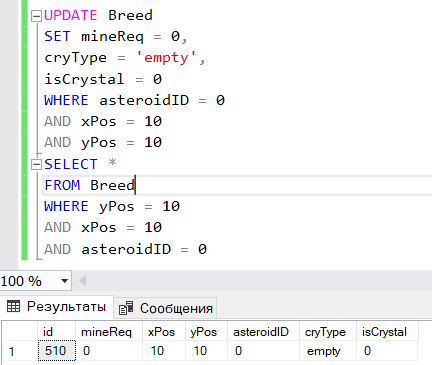


Рисунок 43 – Запрос 13

В запросе 13 происходит замена породы на пустую (при учёте, что робот её выкопал)

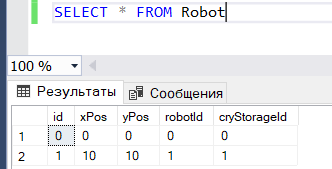


Рисунок 44 – Запрос 14

В запросе 14 происходит вывод всех существующих роботов

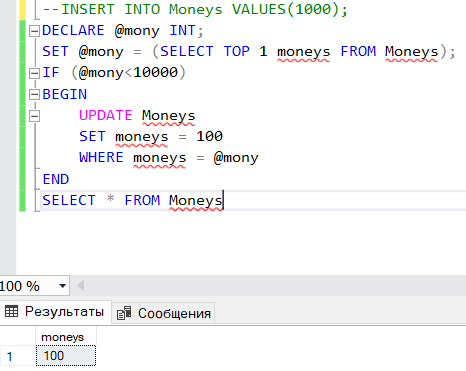


Рисунок 45 – Запрос 15

В запросе 15 при условии, что денег мало их становится ещё меньше. Если переменная @mony меньше 10000, то количество изменяется на 100.

**3.4 Разграничение прав доступа**

Придумаем

**4. Разработка клиентского приложения**

**4.1. Обоснование выбора языка программирования**

Для разработки приложения был выбран язык программирования C# + платформа Unity.

У разных платформ разный язык программирования, например, Unity использует C#, UnrealEngine C++, Godot и Construct Engine 2 свой встроенный + блочный.

C# превосходит C++ наличием встроенных библиотек, что ускоряет разработку программ, взамен жертвуя памятью. Основные такие библиотеки: List, Linq, Plinq цикл foreach. Благодаря этому и был выбран язык программирования C#, а вместе с ним и платформа разработка Unity.

Если говорить конкретно о платформах, то Godot год назад запретил российским разработчикам использовать платформу, Construct имеет свой язык, остаются Unreal Engine и Unity – 2 стабильные платформы. Из минусов Unity – не поддерживается в Крыму, требует fee за количество скачиваний приложения.

**4.2. Разработка интерфейса пользователя**

**4.3. Алгоритм работы каждого из модулей**

**4.4. Тестирование работы приложения**

**Приложения**

Приложение A – Листинг кода SQL

CREATE TABLE Asteroid

(

id int PRIMARY KEY NOT NULL,

xPos int,

yPos int,

asteroidName VARCHAR(64)

);

CREATE TABLE Breed

(

id int PRIMARY KEY NOT NULL,

mineReq int,

xPos int,

yPos int,

asteroidID int,

cryType VARCHAR(64),

isCrystal bit

);

CREATE TABLE Crystal

(

cryType VARCHAR (64) PRIMARY KEY NOT NULL,

durabilitys int

);

CREATE TABLE CrystalType

(

cryType VARCHAR (64) PRIMARY KEY

);

CREATE TABLE CrystalStorage

(

id int PRIMARY KEY NOT NULL,

storageId int,

cryType VARCHAR(64),

ammount int

);

CREATE TABLE Storage

(

id int PRIMARY KEY NOT NULL,

storageName VARCHAR(64)

);

CREATE TABLE Robot

(

id int PRIMARY KEY NOT NULL,

xPos int,

yPos int,

robotId int,

cryStorageId int UNIQUE,

--speed

);

CREATE TABLE RobotTransporter

(

id int PRIMARY KEY NOT NULL

);

CREATE TABLE RobotMiner

(

id int PRIMARY KEY NOT NULL

);

CREATE TABLE Buildings

(

id int PRIMARY KEY NOT NULL,

xPos int,

yPos int,

buildingType int,

buildingCry int UNIQUE

);

CREATE TABLE Skills

(

id int PRIMARY KEY NOT NULL,

skillLvl int,

skillName VARCHAR(64),

skillValue float,

lvlupCost float,

storageId int,

robotId int,

);

ALTER TABLE Breed

ADD CONSTRAINT fkAsteroid

FOREIGN KEY (asteroidID) REFERENCES Asteroid(id);

------------ALTER TABLE Breed

------------ADD CONSTRAINT fkCryType

------------FOREIGN KEY (cryType) REFERENCES CrystalType(cryType);

ALTER TABLE Crystal

ADD CONSTRAINT fkCrystalCryType

FOREIGN KEY (cryType) REFERENCES CrystalType(cryType);

ALTER TABLE CrystalStorage

ADD CONSTRAINT fkCSCryType

FOREIGN KEY (cryType) REFERENCES CrystalType(cryType);

ALTER TABLE CrystalStorage

ADD CONSTRAINT fkStorageId

FOREIGN KEY (storageId) REFERENCES Storage(id);

ALTER TABLE Buildings

ADD CONSTRAINT fkBuildingsCry

FOREIGN KEY (buildingCry) REFERENCES Storage(id);

ALTER TABLE RobotTransporter

ADD CONSTRAINT fkRobotTransporterId

FOREIGN KEY (id) REFERENCES Robot(id);

ALTER TABLE RobotMiner

ADD CONSTRAINT fkRobotMinerId

FOREIGN KEY (id) REFERENCES Robot(id);

ALTER TABLE Robot

ADD CONSTRAINT fkRobotStorage

FOREIGN KEY (cryStorageId) REFERENCES Storage(id);

ALTER TABLE Skills

ADD CONSTRAINT fkRobotSkills

FOREIGN KEY (robotId) REFERENCES Robot(id);

ALTER TABLE Skills

ADD CONSTRAINT fkSkillsStorage

FOREIGN KEY (storageId) References Robot(cryStorageId);

CREATE TABLE SkillTable

(

id int PRIMARY KEY NOT NULL,

skillLvl int,

skillName VARCHAR(64),

skillValue float,

lvlupCost float,

);

CREATE TABLE Moneys

(

moneys float

)

--CREATE TRIGGER trgBeforeInsertSkills

--ON Skills

--INSTEAD OF INSERT

--AS

--BEGIN

-- DECLARE @robotId INT;

-- SELECT @robotId = robotId FROM inserted;

-- IF NOT EXISTS (SELECT 1 FROM RobotMiner WHERE id = @robotId)

-- BEGIN

-- RAISERROR('Нет доступа. Только робот шахтёр имеет доступ', 16, 1);

-- ROLLBACK TRANSACTION;

-- RETURN;

-- END

-- ELSE

-- BEGIN

-- INSERT INTO Skills

-- SELECT \* FROM inserted;

-- END

--END;

--CREATE TRIGGER trgBeforeUpdateSkills

--ON Skills

--INSTEAD OF UPDATE

--AS

--BEGIN

-- DECLARE @robotId INT;

-- SELECT @robotId = robotId FROM inserted;

-- IF NOT EXISTS (SELECT 1 FROM RobotMiner WHERE id = @robotId)

-- BEGIN

-- RAISERROR('Нет доступа. Только робот шахтёр имеет доступ', 16, 1);

-- ROLLBACK TRANSACTION;

-- RETURN;

-- END

-- UPDATE Skills

-- SET skillLvl = i.skillLvl,

-- skillName = i.skillName,

-- skillValue = i.skillValue,

-- lvlupCost = i.lvlupCost,

-- storageId = i.storageId

-- FROM Skills s

-- INNER JOIN inserted i ON s.id = i.id;

--END;

--INSERT INTO CrystalType VALUES ('cryGreen');

--INSERT INTO CrystalType VALUES ('cryBlue');

--INSERT INTO CrystalType VALUES ('cryRed');

--SELECT \* FROM CrystalType;

--INSERT INTO Crystal VALUES ('cryBlue', 6);

--INSERT INTO Crystal VALUES ('cryGreen', 7);

--INSERT INTO Crystal VALUES ('cryRed', 5);

--SELECT \* FROM Crystal;

--INSERT INTO CrystalStorage VALUES(0, 0, 'cryGreen', 10);

--INSERT INTO CrystalStorage VALUES(1, 0, 'cryBlue', 25);

--INSERT INTO CrystalStorage VALUES(2, 0, 'cryRed', 40);

--INSERT INTO CrystalStorage VALUES(3, 1, 'cryGreen', 568);

--INSERT INTO CrystalStorage VALUES(4, 1, 'cryBlue', 426);

--INSERT INTO CrystalStorage VALUES(5, 1, 'cryRed', 1112);

--SELECT \* FROM CrystalStorage;

--INSERT INTO Storage VALUES (0, 'storageA');

--INSERT INTO Storage VALUES (1, 'storageB');

--SELECT \* FROM Storage;

--INSERT INTO Robot VALUES(0, 0, 0, 0, 0);

--INSERT INTO Robot VALUES(1, 10, 10, 1, 1);

--SELECT \* FROM Robot;

--INSERT INTO RobotMiner VALUES(0);

--INSERT INTO RobotTransporter VALUES(1);

--SELECT \* FROM RobotMiner;

--SELECT \* FROM RobotTransporter;

--INSERT INTO Skills VALUES (0, 0, 'Mining', 100, 50, 0, 0);

--INSERT INTO Skills VALUES (1, 0, 'CryMining', 1, 50, 0, 0);

--SELECT \* FROM Skills;

--INSERT INTO Buildings VALUES (0, 5, 5, 3, 1);

--SELECT \* FROM Buildings;

--SELECT \* FROM SkillTable

--DELETE FROM SkillTable

--SELECT \* FROM Breed;

--SELECT \* FROM Asteroid;

--DELETE FROM Breed;

--DELETE FROM Asteroid;

--SELECT Count(b.id) AS breedId

--FROM Breed b

--JOIN Asteroid a ON b.asteroidID = a.id

--WHERE a.id = 0;

--SELECT (b.xPos), (b.yPos), (b.cryType)

--FROM Breed b

--JOIN Asteroid a ON b.asteroidID = a.id

--WHERE b.cryType = 'cryGreen' AND a.id = 0;

--INSERT INTO Skills VALUES (1, 0, 'CryMining', 1, 50, 0, 0);

--INSERT INTO Skills VALUES (1, 0, 'CryMining', 1, 50, 0, 1);

--SELECT \* FROM Skills;

--INSERT INTO CrystalStorage VALUES(6, 1, 'asfw', 1112);

--SELECT (cs.cryType), (cs.ammount)

--FROM CrystalStorage cs

--JOIN Storage s ON s.id = cs.storageId

--WHERE s.id = 0;

--UPDATE Skills

--SET skillLvl = 1,

--skillValue = 110,

--lvlupCost = 60

--WHERE skillName = 'Mining' and robotId = '0';

--SELECT \* FROM Skills;

--INSERT INTO Buildings VALUES (0, 5, 5, 3, 3);

--SELECT \* FROM Buildings;

--SELECT durabilitys

--FROM Crystal

--WHERE cryType = 'cryGreen';

--SELECT \*

--FROM Breed b

--JOIN Asteroid a ON b.asteroidID = a.id

--WHERE a.id = 1

--AND b.xPos > 50

--AND b.xPos < 60

--AND b.yPos = 10

--UPDATE Breed

--SET mineReq = 0,

--cryType = 'empty',

--isCrystal = 0

--WHERE asteroidID = 0

--AND xPos = 10

--AND yPos = 10

--SELECT \*

--FROM Breed

--WHERE yPos = 10

--AND xPos = 10

--AND asteroidID = 0

--SELECT \* FROM Robot

--INSERT INTO Moneys VALUES(1000);

--DECLARE @mony INT;

--SET @mony = (SELECT TOP 1 moneys FROM Moneys);

--IF (@mony<10000)

--BEGIN

-- UPDATE Moneys

-- SET moneys = 100

-- WHERE moneys = @mony

--END

--SELECT \* FROM Moneys

Приложение B – Листинг кода генераторов VisualStudio

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Channels;

using System.Threading.Tasks;

namespace Genrator

{

public class Generator

{

public void Generate(int astX, int astY,string AsteroidName, int x, int y, int asteroidId, float mainCryConcentration, int mainCryId, float hardBreakConcentration, int hbId, int idMove, int totalBreeds)

{

DbHelper.ExecuteWithoutAnswer($"INSERT INTO Asteroid VALUES ({asteroidId}, {astX}, {astY}, '{AsteroidName}')");

//Менять ручками deepFactor (33,3% mc + hb = full map)

List<float> BaseChanches = ChanceCalc(mainCryConcentration, hardBreakConcentration, totalBreeds, mainCryId, hbId, 0.5f); //0-30%

List<float> AvgChanches = ChanceCalc(mainCryConcentration, hardBreakConcentration, totalBreeds, mainCryId, hbId, 1.5f); //30-70%

List<float> DeepChanches = ChanceCalc(mainCryConcentration, hardBreakConcentration, totalBreeds, mainCryId, hbId, 3f); //70-100%

List<int> mineReq = new List<int>() { 0, 9999, 1, 15, 35, 80};

List<int> isCrystal = new List<int>() {3, 4, 5};

List<string> names = new List<string>() {"empty", "hardBreak", "weak", "cryGreen", "cryBlue", "cryRed"};

int hbCount = 0;

int curHbCount = 0;

int idM = idMove + 0;

bool hardBreakCondition = false;

Random rnd = new Random();

foreach (string crystalName in names)

{

DbHelper.ExecuteWithoutAnswer($"IF NOT EXISTS (SELECT 1 FROM CrystalTypes WHERE cryType = '{crystalName}') BEGIN INSERT INTO CrystalTypes (cryType) VALUES ('{crystalName}') END");

}

for (int yi = 0; yi<y;yi++)

{

for (int xj = 0; xj<x;xj++)

{

if(yi==0)

{

DbHelper.ExecuteWithoutAnswer($"INSERT INTO Breed VALUES ({idM}, {mineReq[0]}, {xj}, {yi},{asteroidId}, '{names[0]}', 0)");

}

else

{

List<float> chances = (yi <= y / 100 \* 30) ? BaseChanches : (yi <= y / 100 \* 70) ? AvgChanches : DeepChanches;

if (hardBreakCondition)

{

Console.WriteLine($"INSERT INTO Breed VALUES ({idM}, {mineReq.Where(mr => mineReq.IndexOf(mr) == hbId).First()}, {xj}, {yi},{asteroidId}, '{names.Where(n => names.IndexOf(n) == hbId).First()}', 0)");

DbHelper.ExecuteWithoutAnswer($"INSERT INTO Breed VALUES ({idM}, {mineReq.Where(mr => mineReq.IndexOf(mr) == hbId).First()}, {xj}, {yi},{asteroidId}, '{names.Where(n=>names.IndexOf(n)==hbId).First()}', 0)");

curHbCount++;

if(curHbCount>=hbCount)

{

hardBreakCondition = false;

}

}

else

{

int BreedId = RandomWinner(chances, RandomChance(rnd));

if (BreedId == hbId)

{

hardBreakCondition = true;

hbCount = rnd.Next(1,5);

curHbCount = 0;

}

int ic = (isCrystal.FirstOrDefault(ic => ic == BreedId) > 0) ? 1 :0;

Console.WriteLine($"INSERT INTO Breed VALUES ({idM}, {mineReq.Where(mr => mineReq.IndexOf(mr) == BreedId).First()}, {xj}, {yi},{asteroidId}, {names.Where(n => names.IndexOf(n) == BreedId).First()}, 0)");

DbHelper.ExecuteWithoutAnswer($"INSERT INTO Breed VALUES ({idM}, {mineReq.Where(mr => mineReq.IndexOf(mr) == BreedId).First()}, {xj}, {yi}, {asteroidId}, '{names.Where(n => names.IndexOf(n) == BreedId).First()}', {ic})");

}

}

idM++;

Console.WriteLine(idM + " / " + x\*y+idMove);

}

}

}

public float RandomChance(Random rnd)

{

float value = rnd.Next(0,10000)/100;

return value;

}

public int RandomWinner(List<float> values, float value)

{

float curChance = 0;

for(int i = 0;i<values.Count;i++)

{

if (curChance+values[i]>=value)

{

return i;

}

curChance += values[i];

}

return 0;

}

public List<float> ChanceCalc(float mainCryConcentration, float hardBreakConcentration, int BreedsCount, int mainCryId, int hbId, float deepFactor)

{

List<float> result = new List<float>();

float chance = (100 - mainCryConcentration\*deepFactor - hardBreakConcentration\*deepFactor) / BreedsCount;

float curCh = 0;

for (int n = 0; n < BreedsCount; n++)

{

float chanceValue = chance;

if (n == mainCryId)

{

chanceValue = mainCryConcentration;

}

if (n == hbId)

{

chanceValue = hardBreakConcentration;

}

curCh += chanceValue;

result.Add(curCh);

}

return result;

}

public void GenerateSkill(int lvls, float baseIncreace, int increasePattern, float baseValue, string skillName, int idMove, float baseCost, float costIncreace)

{

for (int lvl = 0; lvl < lvls; lvl++)

{

float skillValue=0;

switch (increasePattern)

{

case 0:

skillValue = baseValue + lvl \* baseIncreace;

break;

case 1:

skillValue = baseValue + (float)Math.Sqrt(lvl \* baseIncreace);

break;

}

Console.WriteLine(lvl);

int ll = lvl;

float lvlUpCost = baseCost + (float)Math.Pow(ll\*costIncreace, 2)\*10;

Console.WriteLine($"INSERT INTO SkillTable (id, skillLvl, skillName, skillValue, lvlUpCost) VALUES ({idMove + lvl},{lvl}, '{skillName}',{skillValue.ToString().Replace(',', '.')}, {lvlUpCost.ToString().Replace(',', '.')})");

DbHelper.ExecuteWithoutAnswer($"INSERT INTO SkillTable (id, skillLvl, skillName, skillValue, lvlUpCost) VALUES ({idMove+lvl},{lvl}, '{skillName}', " +

$"{skillValue.ToString().Replace(',', '.')}, {lvlUpCost.ToString().Replace(',', '.')})");

}

}

public static void Main()

{

Console.WriteLine(DbHelper.ExecuteQueryWithAnswer("SELECT mineReq FROM Breed WHERE id = 0"));

Generator gen = new Generator();

//aX, aY, aN, x, y, aId,mC,mId,hbC, hbId, idMove totBr 0-5

gen.Generate(3000, 1400, "GreenCry", 50, 30, 0, 10, 3, 1.5f, 1, 0, 6);

gen.Generate(3000, 1400, "Andromeda", 100, 200, 1, 15, 4, 3f, 1, 1500, 6);

//gen.GenerateSkill(100, 10, 0, 100.0f, "Mining", 0, 50.0f, 1.02f);

//gen.GenerateSkill(100, 1.1f, 1, 1, "CryMining", 100, 50.0f, 1.04f);

}

}

}