Содержание

[**Введение** 2](#_Toc182951649)

**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.

**Аналитическая часть**

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

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.

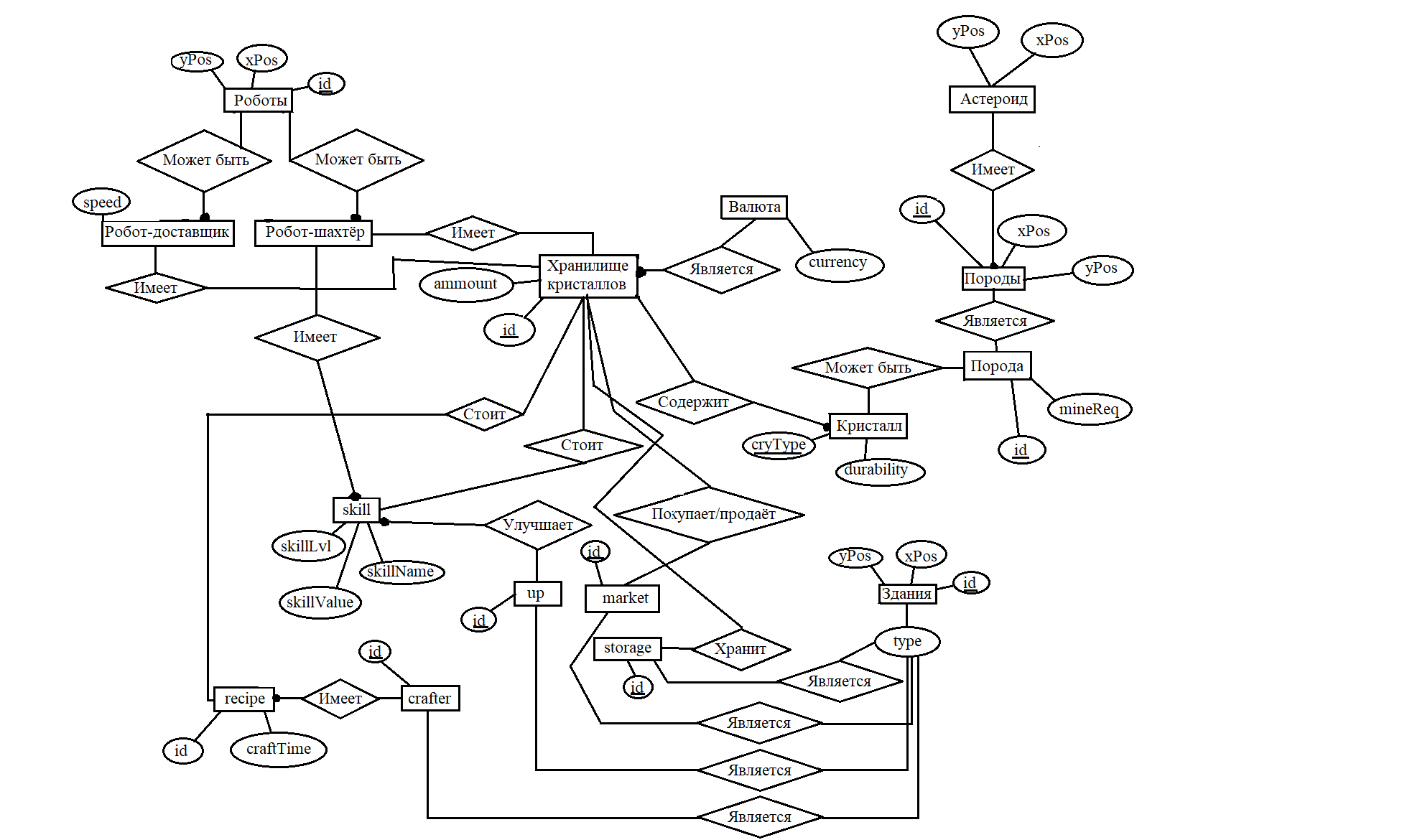


Рисунок 1 – ERD диаграмма по нотации П.Чена

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.

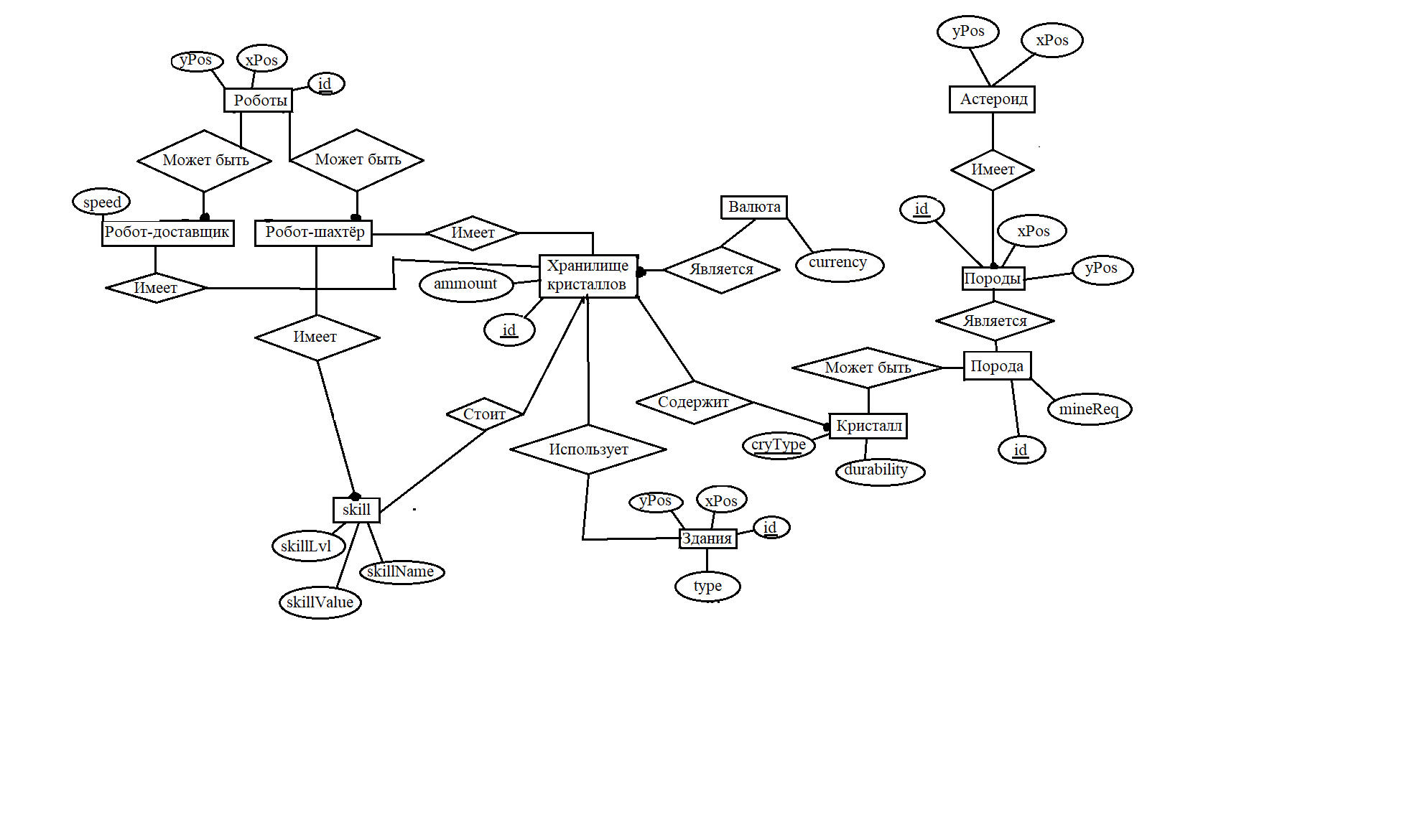


Рисунок 2 – ERD диаграмма (итоговая версия)

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.



Рисунок 3 – KeyBasedModel – A0

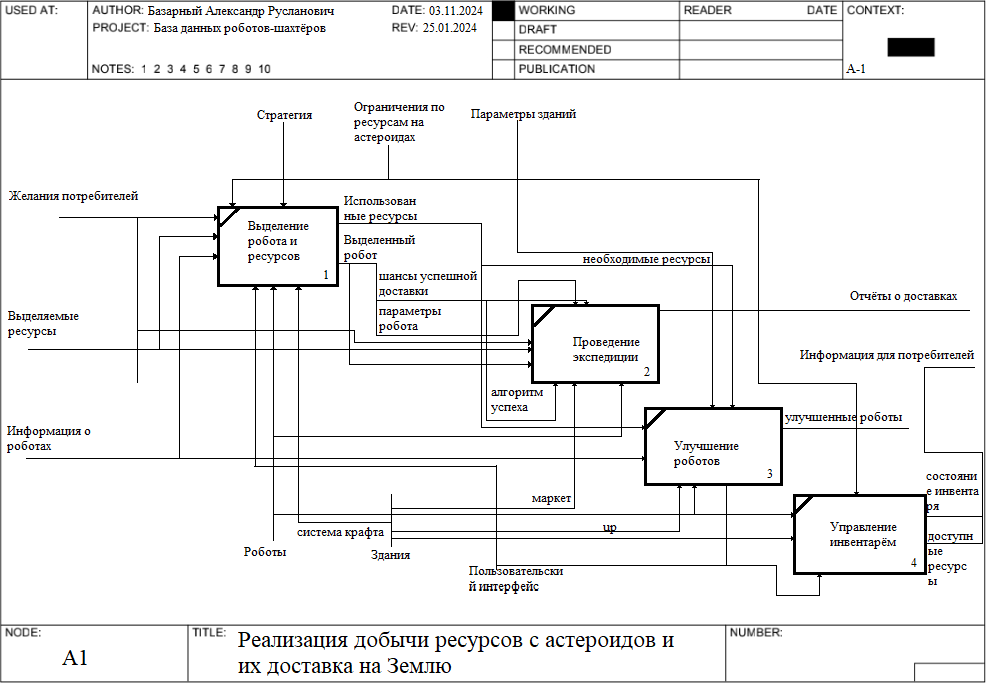


Рисунок 4 – KeyBasedModel – A1

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

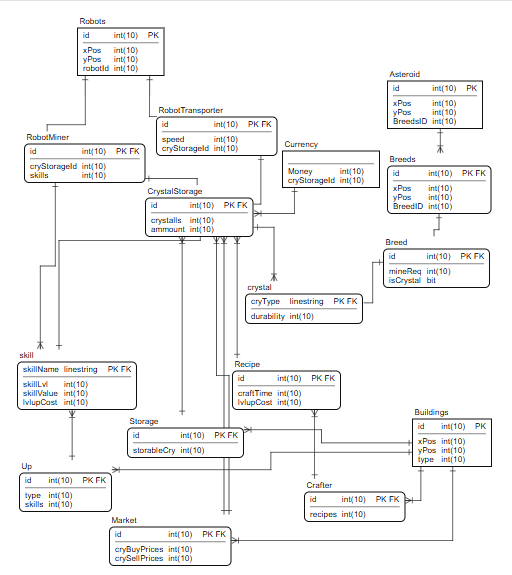


Рисунок 5 – диаграмма IDEF1X (до нормализации )

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

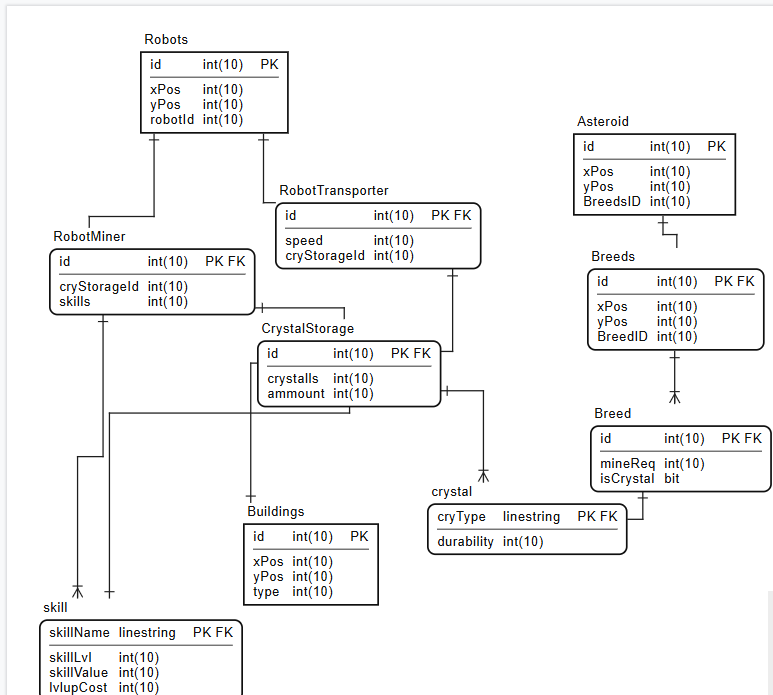


Рисунок 6 – Итоговая диаграмма IDEF1X (до нормализации)

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.

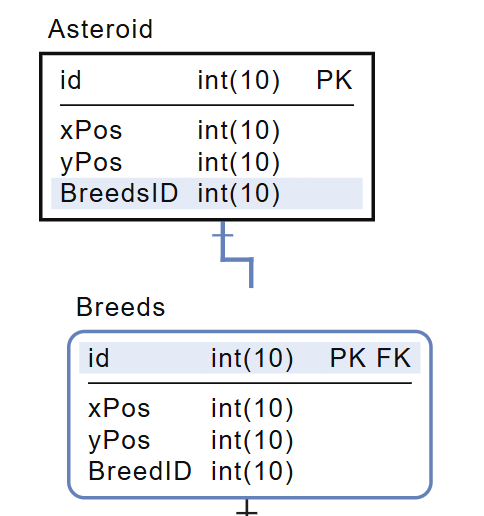


Рисунок 7 – Asteroid-Breeds (1к1)

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

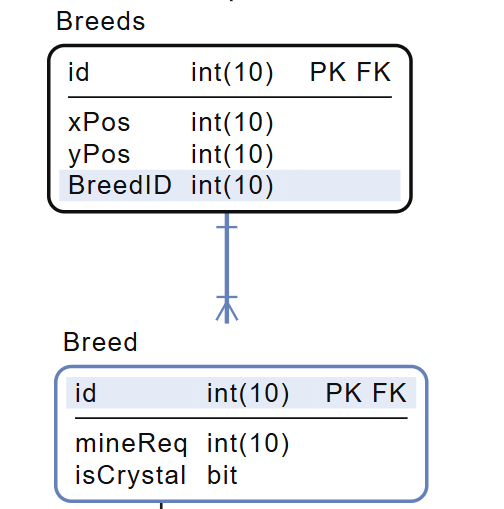


Рисунок 8 – Breeds-Breed (1 ко многим)

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.

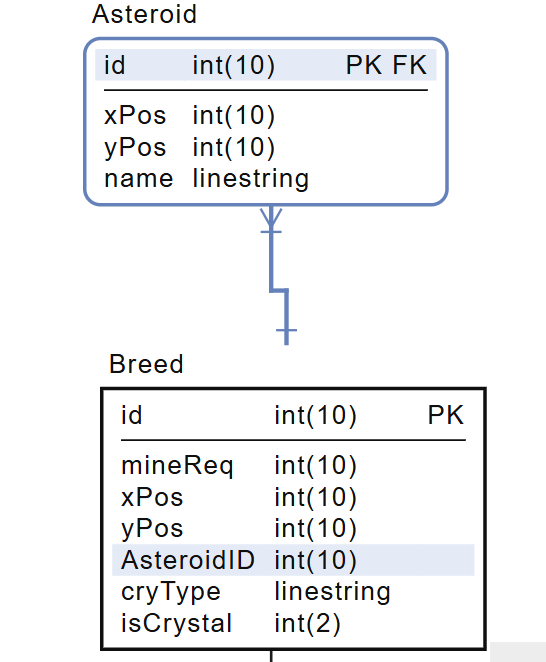


Рисунок 9 – упрощение таблицы. Asteroid-Breed (1 ко многим)

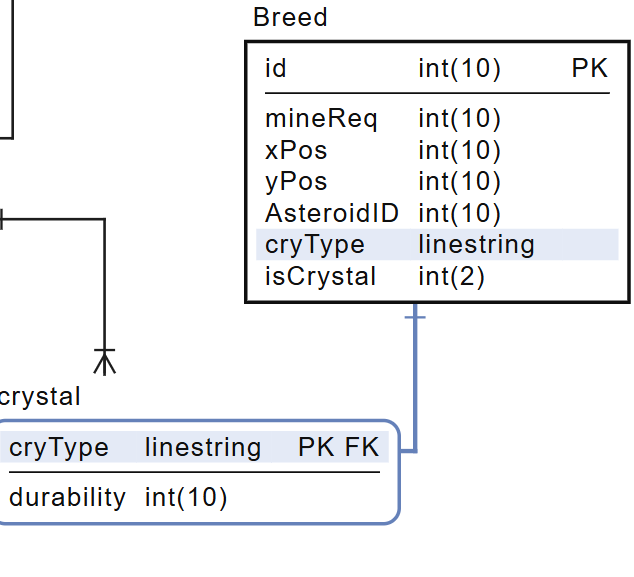


Рисунок 10 – Breed-Crystal (1к1)

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

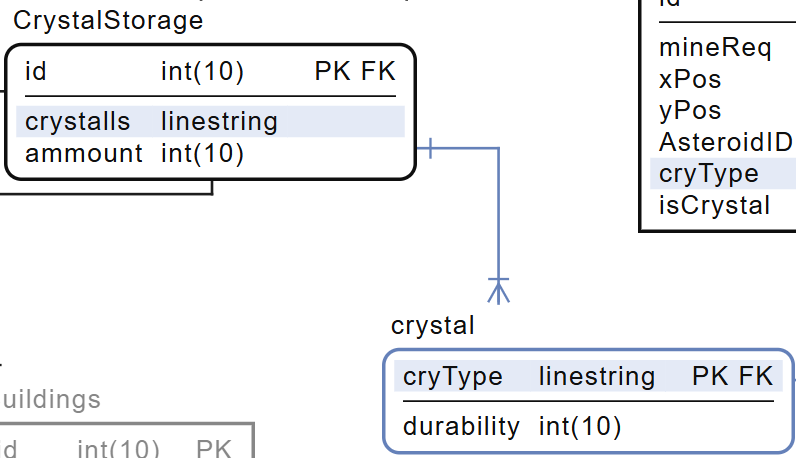


Рисунок 11 – Crystal-CrystalStorage

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

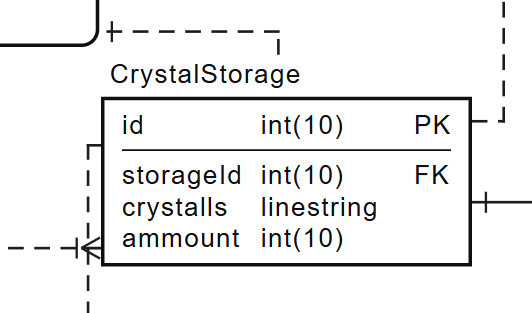


Рисунок 12 – проблема 2нф

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.

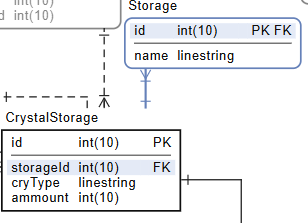


Рисунок 13 – Решение проблемы

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.

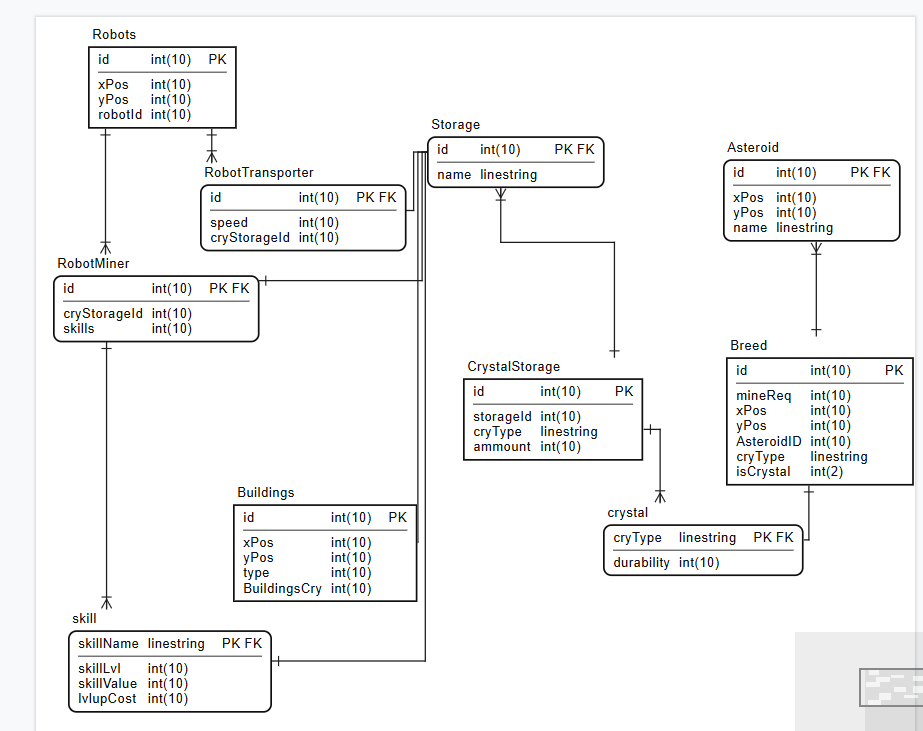


Рисунок 14 – Вид таблицы после нормализации до 2 нф

3НФ: все зависимости от пк нетранзитивны + 2НФ

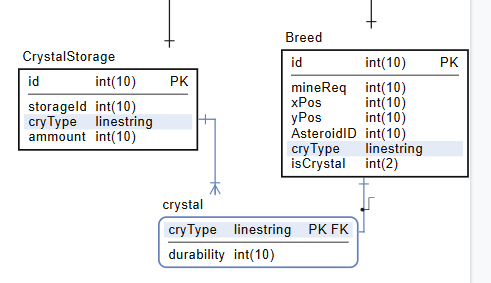


Рисунок 15 – Breed, Asteroid, Crystal

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

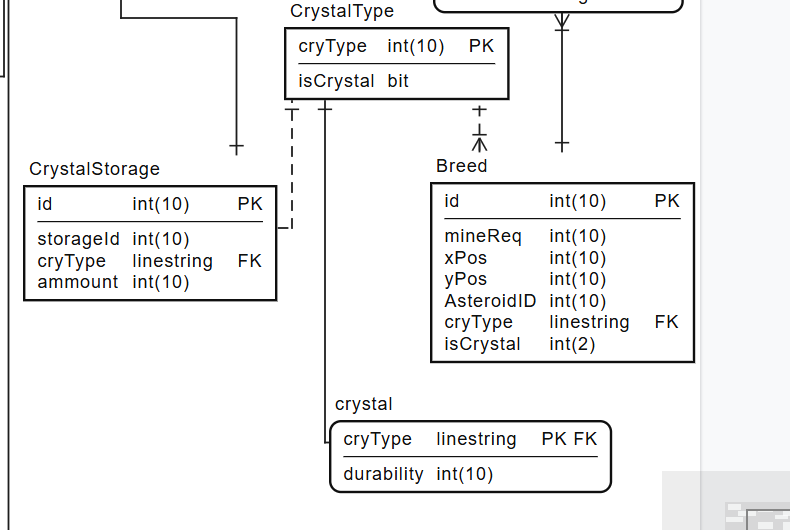


Рисунок 16 – Решение транзитивности

Атрибут cryType больше не транзитивен для Breed, CrystalStorage and Crystal

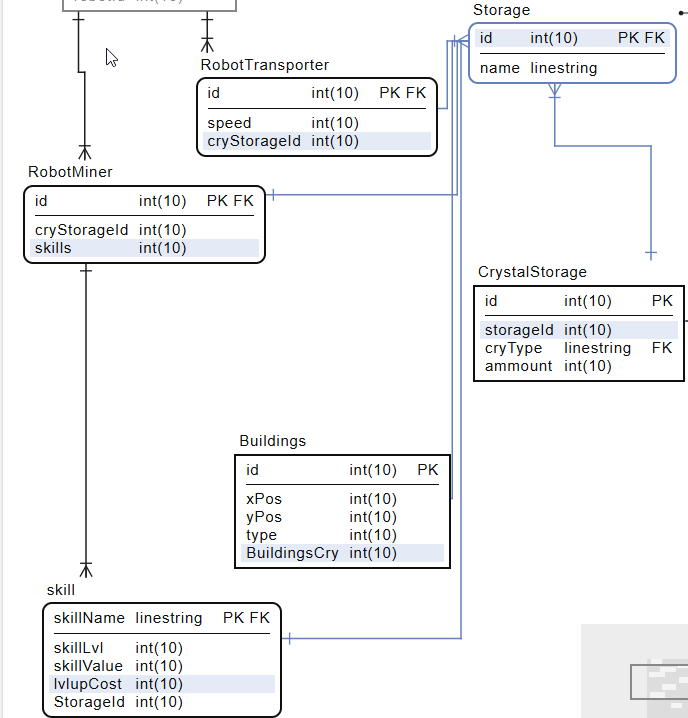


Рисунок 17 – Связи Storage

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.

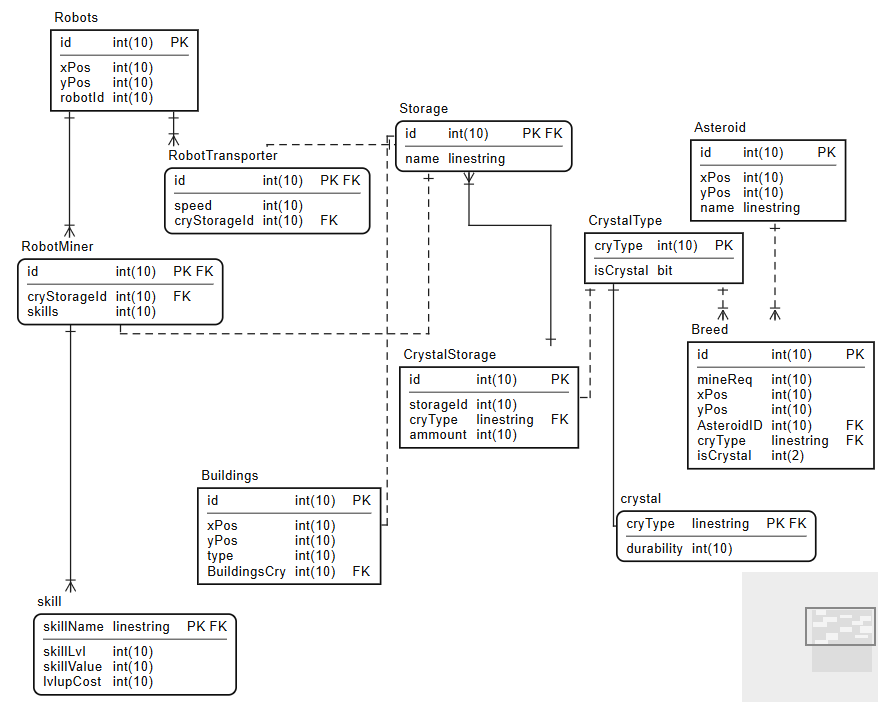


Рисунок 18 – IDEFX1 в 3НФ

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