Экзаменационный проект по дисциплине Базы данных студента гр. **M3436** **Кочеткова Никиты Олеговича** по теме **Рыболовная база**

Описание проекта

В данном проекте рассматривается Рыболовная база. Есть рыбаки, которые рыбачат, и они могут иметь разный возраст. Чтобы стать рыбаком в сети этой базы, необходимо иметь индивидуальный адрес электронной почты, а не чей-то чужой. Также есть некоторые дополнительные параметры, но не суть важно. Эти рыбаки хотят арендовать лодки. Арендует один человек лодку на несколько человек до определённой даты. Оплата происходит сразу при аренде. Если человек возвращает лодку после последнего срока, он доплачивает за пропущенные дни. С этим разбираются физически специальные люди на рыболовной базе. Потом статус лодке ставиться как “c” (eng) - “сданная обратно на базу” в табличке аренды (при аренде ставится “a” - “арендована”). Пока происходит техническое обслуживание лодки она будет оставаться в этой таблице. Как только она будет готова к дальнейшей эксплуатации её удалят и она снова будет доступна для аренды. Но лодки также различаются по моделям, для опытных рыбаков и любителей поплавать на вёслах. После того, как рыбаки арендовали лодку они задумываются у всех ли есть лицензия на ловлю рыбы. Тот у кого лицензии нет, тот не может ловить определённую рыбу. На каждую рыбу нужна своя лицензия, которая истекает по окончанию определённого времени и выдаётся на человека. Теперь у всех есть по лицензии и все отправляются искать место для разбития лагеря. Мы предоставляем на выбор некоторое число островов с их координатами и вместительностью, чтобы плотность людей по группам не была слишком большой. После того, как группа высаживается на остров их заносят в отдельную таблицу, чтобы можно было всегда администратору сказать, насколько загружены разные места для стоянки и есть ли вообще куда плыть. Также у каждого острова есть дополнительная информация по поводу плаванья рыбы. Не на каждом острове есть всякая рыба. Администратор может сказать с какой вероятностью вы можете вытянуть некоторую рыбу вблизи

этого острова. Чтобы не путаться острова помечаются специальными флагами, на которые и указывают координаты. Так рыбаки не запутаются с островами и один остров не будут как новый кучу раз помечать. Во время пребывания на базе рыбаки могут переместиться на другой остров, чтобы половить другую рыбу. Администратор всё это также отмечает в базе. После всех мероприятий рыбаки возвращаются с острова на вход базы. Их удаляются с последнего острова пребывания. А также завершают аренду лодки с поправкой на дату (если аренда была завершена раньше, то деньги не возвращаются).

Построение отношений

В результате предварительного проектирования были выделены следующие отношения:

- **Fisherman** - рыбак, который может рыбачить на воде, окружённый островами. У него есть имя (ФИО) и возраст (>= 0). Возраст для ограничения на аренду лодок, для рыбаков с возрастом меньше 18 лет. Возраст фактический на момент последнего обращения. Обновляется при просьбе (если был достигнут более взрослый). Также у него есть email, который позволяет его однозначно идентифицировать от его тёзок.
- **Boat** - лодка. Имеет вместительно (>= 1) и стоимость на 1 день аренды (>= 0 рублей, копейки не рассматриваются). Также содержит некоторое описание, которое характеризует её внешнее и качественное состояние, но оно не обязательно (можно посмотреть вживую). Также у лодки есть тип - это может быть как вёсельная лодка W2020 от Karma (некоторая компания) , так и моторная C78-R от Kontraw.
- **BoatRent** - аренда лодки. Содержит того, кто арендует (18+ по возрасту) и id лодки, которую арендуют, а также срок до которого арендуют.
- **BoatKind** - тип лодки. Есть название это типа (напр. модель). А также небольшое описание этого самого типа (не обязательно). Есть также состояние аренды (“a” - в аренде, “c” - лодка сдана обратно).
- **Island** - остров, на который могут высаживаться рыбаки. Есть название, чтобы было удобно ориентироваться. Есть широта с долготой, которые помогают отделять разные острова. Они указывают на единственный специальный физический флажок на острове, который его идентифицирует. Также есть вместимость (>= 1) на острове для комфортного проживания. Больше человек он вмещает на нём не может быть. Узнать сколько мест осталось, можно запросом у администратора рыболовной базы.
- **FishermanIsland** - показывает, кто где находится. Имеет id рыбака и id острова, на котором он расположился.

- **FishingPermit** - лицензия на ловлю определённой рыбы. Выдаётся рыбаку на определённую рыбу по какой-то срок. Если срок кончился, рыбак должен выпустить такую пойманную рыбу сразу.
- **Habitability** - обитаемость. Можно узнать насколько велик шанс поймать определённую рыбу на определённом острове (0 <= p <= 1).
- **Fish** - рыба, которая водится на рыболовной базе. У рыбы есть название.

Отношение Fisherman

Атрибуты:
- FishermanId
- FishermanName
- Age
- Email

Функциональные зависимости:
- FishermanId -> FishermanName
- FishermanId -> Age
- FishermanId -> Email
- Email -> FishermanId

Ключи: {FishermanId}, {Email}
Так как замыкание этого множества, даёт все элементы и оно минимально по включению.
Сверху функциональные зависимости уже образуют неприводимое множество.

Нормализация

1НФ:
Отношение в 1НФ, так как нет повторяющихся групп, все атрибуты атомарны, у отношения есть ключ.

2НФ:
Отношение в 2НФ, так как нет составных ключей, а значит неключевые атрибуты зависят от ключа в целом (не от части ключа).
А также уже в 1НФ.

3НФ:
Отношение в 3НФ, так как неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.
А также уже в 2НФ.

НФБК:
Отношение находится в НФБК, так как в 3НФ и в каждой нетривиальной функциональной зависимости X→Y, X является надключом.

4НФ:
Отношение в 4НФ, так как по теореме Дейта-Фейгина 2: “Если отношение находится в НФБК и существует простой ключ → отношение находится в 4НФ”, а у нас есть простой ключ.
А также уже в НФБК.

5НФ:
Отношение в 5НФ, так как по теореме Дейта-Фейгина 1: “Если отношение находится в 3НФ и все ключи простые → отношение находится в 5НФ”, а у нас все ключи простые и отношение в 3НФ.

Отношение Boat

Атрибуты:
- BoatId
- BoatKindId
- BoatCapacity
- Price
- BoatDescription

Функциональные зависимости:
- BoatId -> BoatKindId
- BoatId -> BoatCapacity
- BoatId -> Price
- BoatId -> BoatDescription

Ключи: {BoatId}
Так как замыкание этого множества, даёт все элементы и оно минимально по включению.
Сверху функциональные зависимости уже образуют неприводимое множество.

Нормализация
1НФ:
Отношение в 1НФ, так как нет повторяющихся групп, все атрибуты атомарны, у отношения есть ключ.

2НФ:
Отношение в 2НФ, так как ключ не составной, а значит неключевые атрибуты зависят от ключа в целом (не от части ключа).
А также уже в 1НФ.

3НФ:
Отношение в 3НФ, так как неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.

А также уже в 2НФ.

НФБК:
Отношение находится в НФБК, так как в 3НФ и в каждой нетривиальной функциональной зависимости X→Y, X является надключом.

4НФ:
Отношение в 4НФ, так как по теореме Дейта-Фейгина 2: “Если отношение находится в НФБК и существует простой ключ → отношение находится в 4НФ”, а у нас есть простой ключ.
А также уже в НФБК.

5НФ:
Отношение в 5НФ, так как по теореме Дейта-Фейгина 1: “Если отношение находится в 3НФ и все ключи простые → отношение находится в 5НФ”, а у нас все ключи простые и отношение в 3НФ.

Отношение BoatRent

Атрибуты:
- FishermanId
- BoatId
- BoatRentDeadline
- State

Функциональные зависимости:
- FishermanId, BoatId -> BoatRentDeadline
- FishermanId, BoatId -> State

Ключи: {FishermanId, BoatId}
Так как замыкание этого множества, даёт все элементы и оно минимально по включению.

Нормализация
1НФ:
Отношение в 1НФ, так как нет повторяющихся групп, все атрибуты атомарны, у отношения есть ключ.

2НФ:
Отношение в 2НФ, так как неключевые атрибуты зависят от ключа в целом (не от части ключа).
А также уже в 1НФ.

3НФ:
Отношение в 3НФ, так как неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.
А также уже в 2НФ.

НФБК:
Отношение находится в НФБК, так как в 3НФ и в каждой нетривиальной функциональной зависимости X→Y, X является надключом.

4НФ:
Отношение в 4НФ, так как каждая нетривиальная МЗ является ФЗ. Для каждой нетривиальной МЗ X ->> Y|Z и X– надключ. Для каждой нетривиальной МЗ X ->> Y|Z и атрибута A: X -> A.
А также уже в НФБК.

5НФ:
Для каждой нетривиальной ЗС *{X1,X2,...,Xn} каждое X_i – надключ.
Отношение уже в 4НФ.
Найдём все нетривиальные зависимости соединений. Попытаемся разделить на разные по размеру части (на 2 нет смысла, так как это 4НФ).
Придётся много перебрать, но мы в итоге получим, что итоговые таблицы не будут совпадать с изначальной, то есть (pi(r1) njoin … njoin pi(rn)) != R, так как будут лишние строки. Значит отношение в 5НФ.

Отношение BoatKind

Атрибуты:
- BoatKindId
- Name
- BoatKindDescription

Функциональные зависимости:
- BoatKindId -> Name
- BoatKindId -> BoatKindDescription

Ключи: {BoatKindId}
Так как замыкание этого множества, даёт все элементы и оно минимально по включению.
Сверху функциональные зависимости уже образуют неприводимое множество.

Нормализация
1НФ:
Отношение в 1НФ, так как нет повторяющихся групп, все атрибуты атомарны, у отношения есть ключ.

2НФ:
Отношение в 2НФ, так как ключ не составной, а значит неключевые атрибуты зависят от ключа в целом (не от части ключа).
А также уже в 1НФ.

3НФ:
Отношение в 3НФ, так как неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.
А также уже в 2НФ.

НФБК:
Отношение находится в НФБК, так как в 3НФ и в каждой нетривиальной функциональной зависимости X→Y, X является надключом.

4НФ:
Отношение в 4НФ, так как по теореме Дейта-Фейгина 2: “Если отношение находится в НФБК и существует простой ключ → отношение находится в 4НФ”, а у нас есть простой ключ.
А также уже в НФБК.

5НФ:
Отношение в 5НФ, так как по теореме Дейта-Фейгина 1: “Если отношение находится в 3НФ и все ключи простые → отношение находится в 5НФ”, а у нас все ключи простые и отношение в 3НФ.

Отношение Island

Атрибуты:
- IslandId
- IslandName
- Longitude
- Latitude
- IslandCapacity

Функциональные зависимости:
- IslandId -> IslandName
- IslandId -> Longitude
- IslandId -> Latitude
- IslandId -> IslandCapacity
- Longitude, Latitude -> IslandId

Ключи: {IslandId}, {Longitude, Latitude}
Так как замыкание каждого из этих множеств, даёт все элементы и они минимальные по включению.

Сверху функциональные зависимости уже образуют неприводимое множество.

Нормализация

1НФ:
Отношение в 1НФ, так как нет повторяющихся групп, все атрибуты атомарны, у отношения есть ключ.

2НФ:
Отношение в 2НФ, так как неключевые атрибуты зависят от ключа в целом (не от части ключа).
А также уже в 1НФ.

3НФ:
Отношение в 3НФ, так как неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.
А также уже в 2НФ.

НФБК:
Отношение находится в НФБК, так как в 3НФ и в каждой нетривиальной функциональной зависимости X→Y, X является надключом.

4НФ:
Отношение в 4НФ, так как по теореме Дейта-Фейгина 2: “Если отношение находится в НФБК и существует простой ключ → отношение находится в 4НФ”, а у нас есть простой ключ.
А также уже в НФБК.

5НФ:
Для каждой нетривиальной ЗС *{X1,X2,...,Xn} каждое X_i – надключ.
Отношение уже в 4НФ.
Найдём все нетривиальные зависимости соединений. Попытаемся разделить на разные по размеру части (на 2 нет смысла, так как это 4НФ).
Придётся много перебрать, но мы в итоге получим, что итоговые таблицы не будут совпадать с изначальной, то есть (pi(r1) njoin … njoin pi(rn)) != R, так как будут лишние строки. Значит отношение в 5НФ.

Отношение FishermanIsland

Атрибуты:
- FishermanId
- IslandId

Функциональные зависимости:
- FishermanId -> IslandId

Ключи: {FishermanId}
Так как замыкание этого множества, даёт все элементы и оно минимально по включению.

Нормализация

1НФ:
Отношение в 1НФ, так как нет повторяющихся групп, все атрибуты атомарны, у отношения есть ключ.

2НФ:
Отношение в 2НФ, так как ключ не составной, а значит неключевые атрибуты зависят от ключа в целом (не от части ключа).
А также уже в 1НФ.

3НФ:
Отношение в 3НФ, так как неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.
А также уже в 2НФ.

НФБК:
Отношение находится в НФБК, так как в 3НФ и в каждой нетривиальной функциональной зависимости X→Y, X является надключом.

4НФ:
Отношение в 4НФ, так как по теореме Дейта-Фейгина 2: “Если отношение находится в НФБК и существует простой ключ → отношение находится в 4НФ”, а у нас есть простой ключ.
А также уже в НФБК.

5НФ:
Отношение в 5НФ, так как по теореме Дейта-Фейгина 1: “Если отношение находится в 3НФ и все ключи простые → отношение находится в 5НФ”, а у нас все ключи простые и отношение в 3НФ.

Отношение FishingPermit

Атрибуты:

- FishermanId
- FishId
- FishingPermitDeadline

Функциональные зависимости:

- FishermanId, FishId -> FishingPermitDeadline

Ключи: {FishermanId, FishId}
Так как замыкание этого множества, даёт все элементы и оно минимально по включению.
Сверху функциональные зависимости уже образуют неприводимое множество.

Нормализация

1НФ:
Отношение в 1НФ, так как нет повторяющихся групп, все атрибуты атомарны, у отношения есть ключ.

2НФ:
Отношение в 2НФ, так как неключевые атрибуты зависят от ключа в целом (не от части ключа).
А также уже в 1НФ.

3НФ:
Отношение в 3НФ, так как неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.
А также уже в 2НФ.

НФБК:
Отношение находится в НФБК, так как в 3НФ и в каждой нетривиальной функциональной зависимости X→Y, X является надключом.

4НФ:
Отношение в 4НФ, так как каждая нетривиальная МЗ является ФЗ. Для каждой нетривиальной МЗ X ->> Y|Z и X– надключ. Для каждой нетривиальной МЗ X ->> Y|Z и атрибута A: X -> A.
А также уже в НФБК.

5НФ:
Для каждой нетривиальной ЗС *{X1,X2,...,Xn} каждое X_i – надключ.
Отношение в 4НФ.
Найдём все нетривиальные зависимости соединений. Попытаемся разделить на 3 части (на 2 нет смысла, так как это 4НФ).
Можем разбить только так на проекции.

- {FishermanId, FishId} - пусть r1
- {FishermanId, FishingPermitDeadline} - пусть r2
- {FishingPermitDeadline, FishId} - пусть r3

FishermanId	FishId	FishingPermitDeadline
1	1	date1

1	2	date2
2	1	date2
2	2	date1

Рассмотрим проекции каждого из этих подмножеств из множества R, а потом обратно соберём. Получим, что итоговая таблица не совпадает с изначальной, то есть (pi(r1) njoin pi(r2) njoin pi(r3)) != R, так как есть лишние строки. Значит отношение в 5НФ.

Отношение Habitability

Атрибуты:

- IslandId
- FishId
- Chance

Функциональные зависимости:

- IslandId, FishId -> Chance

Ключи: {IslandId, FishId}
Так как замыкание этого множества, даёт все элементы и оно минимально по включению.
Сверху функциональные зависимости уже образуют неприводимое множество.

Нормализация

1НФ:
Отношение в 1НФ, так как нет повторяющихся групп, все атрибуты атомарны, у отношения есть ключ.

2НФ:
Отношение в 2НФ, так как неключевые атрибуты зависят от ключа в целом (не от части ключа).
А также уже в 1НФ.

3НФ:
Отношение в 3НФ, так как неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.
А также уже в 2НФ.

НФБК:
Отношение находится в НФБК, так как в 3НФ и в каждой нетривиальной функциональной зависимости X→Y, X является надключом.

4НФ:
Отношение в 4НФ, так как каждая нетривиальная МЗ является ФЗ. Для каждой нетривиальной МЗ X ->> Y|Z и X– надключ. Для каждой нетривиальной МЗ X ->> Y|Z и атрибута A: X -> A.
А также уже в НФБК.

5НФ:
Для каждой нетривиальной ЗС *{X1,X2,...,Xn} каждое X_i – надключ.
Отношение в 4НФ.
Найдём все нетривиальные зависимости соединений. Попытаемся разделить на 3 части (на 2 нет смысла, так как это 4НФ).
Можем разбить только так на проекции.

- {IslandId, FishId} - пусть r1
- {IslandId, Chance} - пусть r2
- {Chance, FishId} - пусть r3

FishermanId	FishId	Chance
1	1	1
1	2	2
2	1	2
2	2	1

Рассмотрим проекции каждого из этих подмножеств из множества R, а потом обратно соберём. Получим, что итоговая таблица не совпадает с изначальной, то есть (pi(r1) njoin pi(r2) njoin pi(r3)) != R, так как есть лишние строки. Значит отношение в 5НФ.

Отношение Fish

Атрибуты:

- FishId
- FishName

Функциональные зависимости:

- FishId -> FishName

Ключи: {FishId}
Так как замыкание этого множества, даёт все элементы и оно минимально по включению.
Сверху функциональные зависимости уже образуют неприводимое множество.

Нормализация

1НФ:

Отношение в 1НФ, так как нет повторяющихся групп, все атрибуты атомарны, у отношения есть ключ.

2НФ:

Отношение в 2НФ, так как ключ не составной, а значит неключевые атрибуты зависят от ключа в целом (не от части ключа).
А также уже в 1НФ.

3НФ:

Отношение в 3НФ, так как неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.
А также уже в 2НФ.

НФБК:

Отношение находится в НФБК, так как в 3НФ и в каждой нетривиальной функциональной зависимости X→Y, X является надключом.

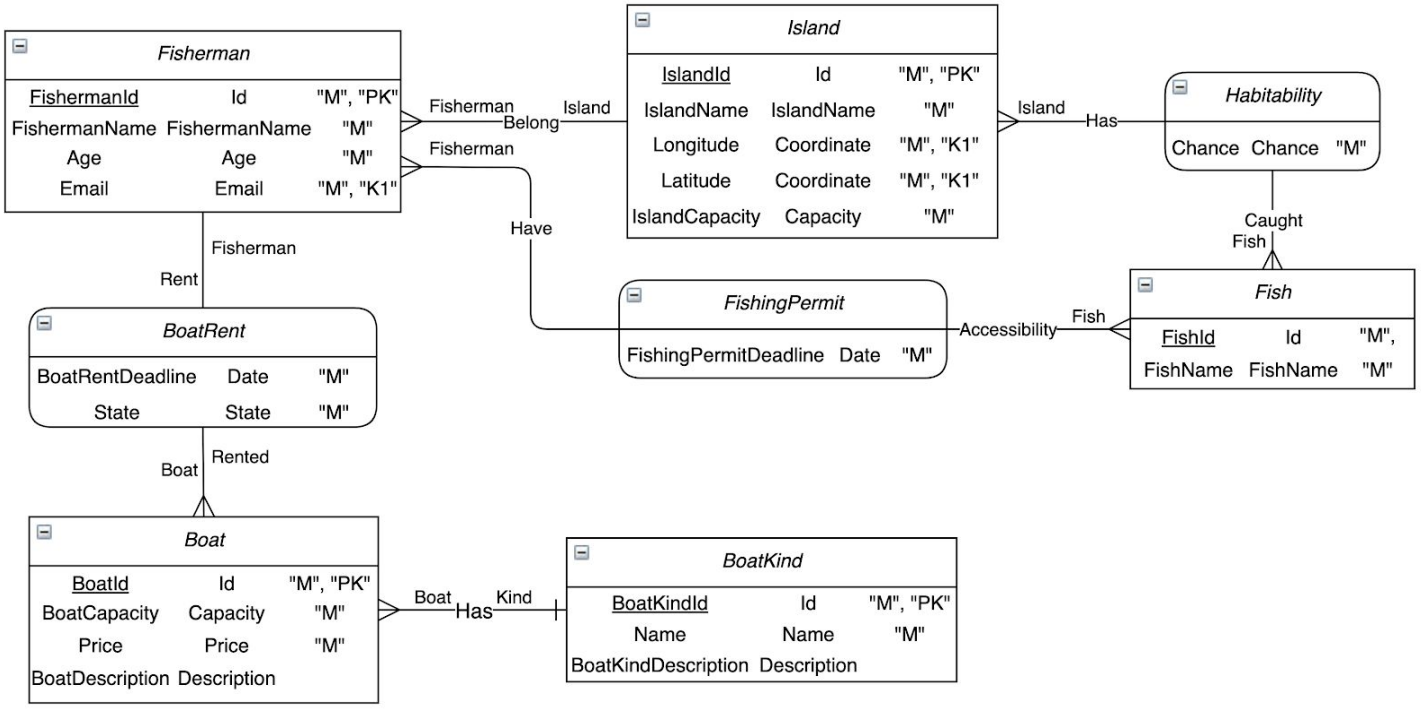
4НФ:

Отношение в 4НФ, так как по теореме Дейта-Фейгина 2: “Если отношение находится в НФБК и существует простой ключ → отношение находится в 4НФ”, а у нас есть простой ключ.
А также уже в НФБК.

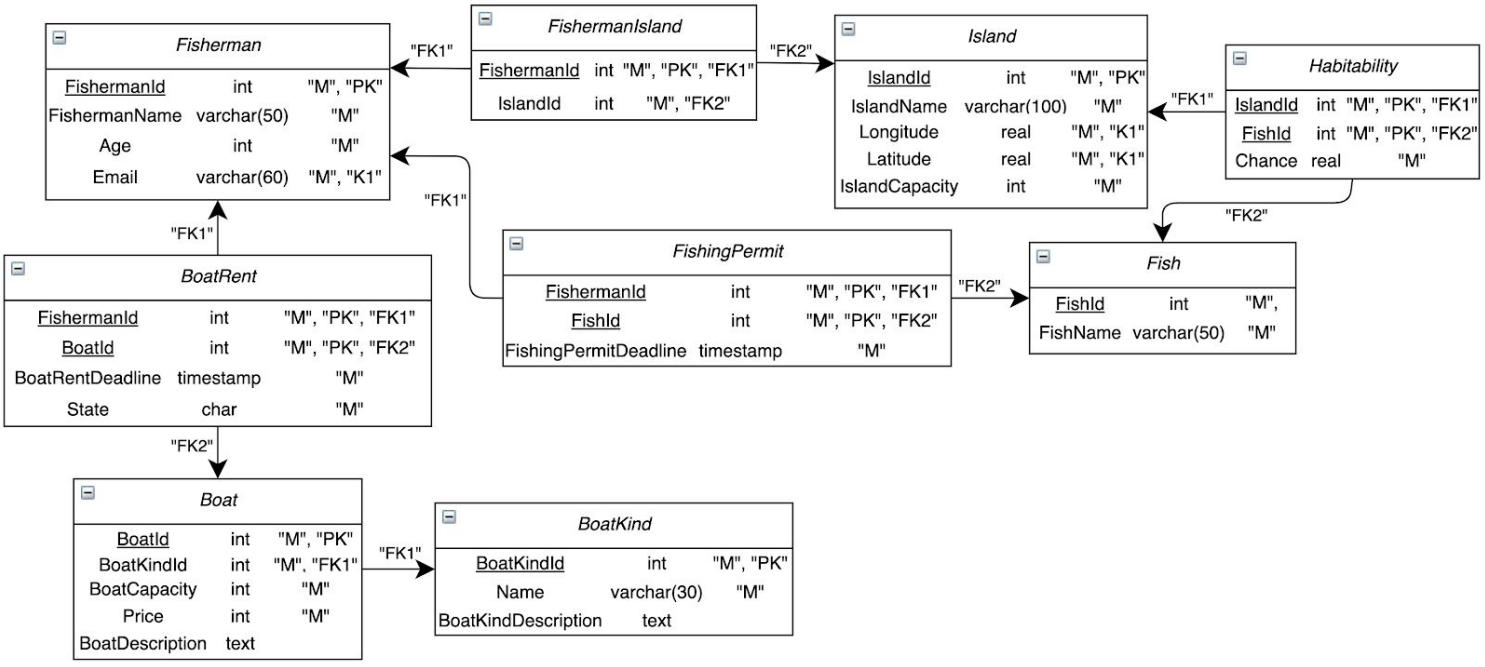
5НФ:

Отношение в 5НФ, так как по теореме Дейта-Фейгина 1: “Если отношение находится в 3НФ и все ключи простые → отношение находится в 5НФ”, а у нас все ключи простые и отношение в 3НФ.

Модель сущность-связь



Физическая модель



При построении физической модели использовалось следующее отображение доменов в типы:

Домен	Тип

Id	int
FishermanName	varchar(50)
Age	int
Email	varchar(60)
Date	timestamp
State	char
Capacity	int
Price	int
Description	text
IslandName	varchar(100)
Coordinate	real
Name	varchar(30)
Chance	real
FishName	varchar(50)

Определения таблиц

Для реализации проекта использовалась СУБД psql (PostgreSQL) 12.4. Определения таблиц и их индексов приведено в файле `ddl.sql`.

Добавляются отдельные ограничения:

- **checkAgeForRent** - ограничение на добавление данных в **BoatRent**. Если у рыбака возраст меньше < 18, то он не сможет арендовать лодку. Реализовать это при задании таблицы сразу нельзя.
- **checkFreeBoatForRent** - ограничение на добавление данных в **BoatRent**. Если лодка, которую хотят добавить, как арендуемую уже есть в таблице, то возвращается сообщение, что нужно подождать пока она освободится. Реализовать это при задании таблицы сразу нельзя.
- **checkBoatInsertOnlyAForRen** - ограничение на добавление данных в **BoatRent**. Если лодка, которую хотят добавить как уже сданную не была в таблице, то будет выдано сообщение, что необходимо сначала её арендовать.
  Реализовать это при задании таблицы сразу нельзя.
- **checkFreeSpaceInIslandForStay** - проверяет, что остров на который хочет переселиться рыбак может его вместить. Если не вмещает, то

говорит поискать пока другой остров. Если хватает места, то добавляет привязывает его к новому острову, а от старого отвязывает.
Реализовать это при задании таблицы сразу нельзя.

- **checkLicenseForPermit** - проверяет, что лицензия, которую хочет купить рыболов будет заменена в случае, если осталась ещё старая, даже если её срок ещё не истёк.
Реализовать это при задании таблицы сразу нельзя.

Индексы:
PostgreSQL автоматически создаёт индексы на Primary key и Unique.
Дополнительные индексы объявлены в конце `ddl.sql`.

Тестовые данные

Скрипт для добавления тестовых данных приведен в файле `data.sql`.

Запросы на получение данных

В рамках проекта были реализованы следующие запросы:

- **Свободные лодки для аренды** — показывает список из свободных лодок с информацией о id, вместимости, цене и описании её состояния.
- **Описание модели лодки** — показывает имя и описание модели лодки по её id.
- **Список свободных островов с хотя бы 1 человеком и количество мест** — показывает список свободных островов на которых есть уже хотя бы 1 человек (id и имя) и количество свободных мест на нём.
- **Список рыб на острове** — показывает все виды рыб, которые могут быть на острове с определённым id.
- **Вероятность поймать рыбу на всех островах, где она есть** — покажет id острова, название и вероятность на нём поймать рыбу
- **Лицензии у рыбаков** — покажет дату окончания (или уже закончившуюся) лицензий у всех рыбаков.
- **Остаток аренды лодок у рыбаков** — покажет список из лодок, которые арендованы рыбаками и дату окончания аренды. Сданные лодки не учитываются.

Запросы на получение данных и вспомогательные представления приведены в файле `selects.sql`. Сначала идут вспомогательные представления **freeBoats**, **freeIslandsPlaces**, **allPermits**, **allRentsFisherBoat**.

Запросы на изменение данных

В рамках проекта были реализованы следующие запросы:

- **Зарегистрировать нового рыбака** — при вызове **addFisherman**(...) в рыболовную БД будет добавлен новый рыбак, если у него email, который раньше не встречался у других людей, а также указаны все обязательные поля.’
  Пример для запуска на тестовых данных: *select **addFisherman**(15, 'Alexand Nevsky', 39, 'bestofbest@yandex.ru');*
- **Забронировать лодку** — в таблицу по броням лодок будет добавлена ещё одна. Если такая возможность есть, иначе будет написана причина, по которой нельзя сейчас забронировать желаемую лодку.
  Пример для запуска на тестовых данных: select **rentBoat**(1, 5, '2024-03-21 11:32:54');
- **Сдать лодку** — в таблице по броням статус лодки будет сменён на ‘c’, если лодка была арендована тем же человеком, что хочет его сменить.
  Пример для запуска на тестовых данных: select **returnTheBoat**(2, 2);
- **Остановиться жить на острове** — добавит рыбака в таблицу с лагерями людей по островам. В случае нехватки места, попросит подождать, пока оно появится.
  Пример для запуска на тестовых данных: select **stayIsland**(1, 4);
- **Получить лицензию** — добавит или обновит лицензию на ловлю рыбы в таблице с лицензиями.
  Пример для запуска на тестовых данных: select **buyPermit**(1, 3, '2022-11-12 06:12:33');

Запросы на изменение данных, хранимые процедуры и триггеры приведены в файле `updates.sql`.

Необходимые триггеры на проверки разных случаев уже определены в `ddl.sql`.

Уровни изоляции для запросов

- **Свободные лодки для аренды**
  Уровень изоляции: **read committed**.
  Так как в данном запросе (read only) мы бегаем в пару таблиц быстро, но не в одну, то другая транзакция во время своего выполнения может что-то в поменять, но не сильно страшно, поэтому read committed.
- **Описание модели лодки**
  Уровень изоляции: **read uncommitted**.
  Так как ходим один раз за описаниями лодок, то никаких проблем быть не может. Поэтому read uncommitted.
- **Список свободных островов с хотя бы 1 человеком и количество мест**
  Уровень изоляции: **repeatable read**.

Несколько раз ходит в память за разными таблицами, которые могли быть изменены другими операциями во время выполнения. Выполняет не быстро. Но аномалии сериализации нет. Поэтому repeatable read.

- **Список рыб на острове**
  Уровень изоляции: **read committed**.
  Аналогично с ситуацией “Свободные лодки для аренды”, так как в данном запросе мы бегаем в 1 таблицу быстро, но пару раз, то другая транзакция во время своего выполнения может что-то в поменять, но не сильно страшно, поэтому read committed.
- **Вероятность поймать рыбу на всех островах, где она есть**
  Уровень изоляции: **read uncommitted**.
  Так как ходим один раз сразу за островами и вероятностью поймать рыбу и больше не обращаемся, то никаких проблем быть не может. Поэтому read uncommitted.
- **Лицензии у рыбаков**
  Уровень изоляции: **read uncommitted**.
  Аналогично ситуации с “Вероятность поймать рыбу на всех островах, где она есть”. Так как ходим один раз сразу за лицензиями и рыбаками и больше не обращаемся, то никаких проблем быть не может. Поэтому read uncommitted.
- **Остаток аренды лодок у рыбаков**
  Уровень изоляции: **read uncommitted**.
  Аналогично ситуации с “Вероятность поймать рыбу на всех островах, где она есть”. Так как ходим один раз сразу за арендой лодок и рыбаками и больше не обращаемся, то никаких проблем быть не может. Поэтому read uncommitted.
- **Зарегистрировать нового рыбака**
  Уровень изоляции: **read uncommitted**.
  Так как мы сразу добавляем человека в таблицу без лишних действий, то read uncommitted.
- **Забронировать лодку**
  Уровень изоляции: **read uncommitted**.
  Аналогично с ситуацией у “зарегистрировать нового рыбака”, поэтому тоже read uncommitted.
- **Сдать лодку**
  Уровень изоляции: **repeatable read**.
  Так как мы делаем обновление данных, а потом ещё дополнительную проверку на корректность, то другая транзакция могла поменять нам запись и попортить данные, поэтому repeatable read.
- **Остановиться жить на острове**
  Уровень изоляции: **read uncommitted**.
  Аналогично с ситуацией у “зарегистрировать нового рыбака”. Мы просто добавляем новое место для жилья, поэтому тоже read uncommitted.
- **Получить лицензию**

Уровень изоляции: **read uncommitted**.
Аналогично с ситуацией у “остановиться жить на острове”. Мы просто добавляем новую лицензию рыбаку, поэтому тоже read uncommitted.