Экзаменационный проект   
по дисциплине  
Проектирование баз данных

студента гр. **M34351**

**Попова Владимира Вячеславовича**

по теме

**Система управления персоналом на складах**

# Описание проекта

В рамках проекта реализуем работу Системы управления персоналом на складах.

Есть различные здания, которые используются в качестве складов и/или иных объектов для хранения товаров (например: склады, склады для крупногабаритных товаров, сортировочные центры или фуллфиллмент центры, которые совмещают в себе как функции склада, так и функции сортировочного центра). Для их описания и учета нам необходимо знать их регион, адрес и часовой пояс.

На каждом складе будем иметь различные зоны, каждая из которых имеет свое название и отличается своим уровнем доступа, а также типом: рабочая или нет.

На каждом складе работают сотрудники: аутстафф (внештатные) – нанятые через компании-посредники, и штатные сотрудники. Для учета персонала нам важно хранить следующую информацию:

- ФИО,

- пол,

- дата рождения,

- ссылка на фото сотрудника (предполагается, что фото загружено в корпоративное хранилище на момент найма),

- серия и номер паспорта,

- номер телефона,

- должность,

- дата найма,

- дата увольнения (при наличии) и причина (при наличии)

- дата заключения и планового окончания контракта (для внештатных сотрудников).

Каждый сотрудник занимает определенную должность. Поэтому нам необходимо хранить название должности, оклад на ней и уровень доступа этой должности на складе:

- internal – допуск к зонам погрузки/разгрузки и упаковки/распаковки товаров,

- external – internal + допуск к залам хранения товаров,

- full – external + допуск к административным зонам (кабинеты штатных сотрудников, отдела кадров, руководства и др.) и зонам управления объектом (диспетчерская, электрощитовая, и т. д.).

Имеется ряд компаний-посредников, с которыми заключается договор о найме внештатных сотрудников на объекты. Для учета компаний будем хранить их контактную информацию: название, адрес, номер телефона и email, а также информацию о заключенном договоре: дата заключения и дата планового завершения.

Для контроля перемещения сотрудников на территории объекта будем логировать перемещение. Каждая запись лога будет содержать в себе следующую информацию:

- время совершения действия,

- тип сотрудника, который перемещается,

- идентификатор склада,

- идентификатор зоны внутри склада,

- тип действия (вход или выход из зоны).

# Построение отношений

В результате предварительного проектирования были выделены следующие отношения:

● **Warehouse** – объект, на котором хранятся товары. Он имеет определенный тип (WH – склад, SC – сортировочный центр, FFC – фуллфиллмент центр, LS – склад для крупногабаритных товаров). У него есть название, адрес, код региона (указывается числом как на автомобильных номерах: от 1 до 999) и часовой пояс (указывается в формате UTC: от -12 до +12).

● **WarehouseZone** – зона склада. По ней могут перемещаться сотрудники, имеющие достаточный уровень доступа (“internal”, “external” или “full”). Она относится к конкретному объекту и может быть рабочей, либо нет.

● **Position** – должность сотрудника. Имеет название, размер оклада в рублях и уровень доступа внутри склада (“internal”, “external”, “full”).

● **StaffWorker** – штатный сотрудник определенного объекта, к которому он прикреплен. Нанимается на неопределенный срок. У него есть ФИО, пол (“M” – male, “F” - female), дата рождения (работают только совершеннолетние), серия и номер паспорта (записываются строкой подряд без пробелов), должность (хранится идентификатором соответствующей **Position**), личное фото (хранится в виде ссылки на файл), номер телефона (хранится в формате “+<код страны><код сети или города><номер телефона>”). Также хранятся данные о дате приема на работу (создание записи сотрудника), увольнения (хранятся в формате DD-MM-YYYY) и причине деактивации (“incident” – увольнение, “own\_request” – по собственному желанию, “by\_agreement” – по соглашению сторон, “reduction” - сокращение).

● **OutstaffWorker** - внештатный сотрудник, которого нанимали через компанию-посредника. Нанимается на заранее определенный в контракте срок. У него есть ФИО, пол (“M” – male, “F” - female), дата рождения (работают только совершеннолетние), серия и номер паспорта (записываются строкой подряд без пробелов), должность (хранится идентификатором соответствующей **Position**), личное фото (хранится в виде ссылки на файл), номер телефона (хранится в формате “+<код страны><код сети или города><номер телефона>”). Также хранятся идентификатор компании, предоставившей сотрудника, дата начала действия контракта с сотрудником и дата его завершения. Также хранится информация об увольнении сотрудника (“incident” – увольнение нашей компанией, “fired\_by\_comp” – увольнение компанией-посредником, “end\_of\_contract” – истечение заключенного договора).

● **OutstaffCompany** – компания-посредник, которая предоставляет внештатных сотрудников для работы на объектах. Имеет свое название, адрес, контакты для связи – телефон (хранится в формате “+<код страны><код сети или города><номер телефона>”) и email, а также дату заключения и расторжения договора о совместной работе.

● **LogEntity** – запись о перемещении сотрудника по объекту. Имеет время и дату совершения действия, а также кто совершал это действие (идентификатор штатного или внештатного сотрудника), идентификатор зоны, на которой совершено действие и его тип (“enter” – вход и “exit” - выход).

## **Отношение Warehouse**

Атрибуты:

● WarehouseId

● WarehouseType

● WarehouseName

● WarehouseAddress

● WarehouseTimezone

● WarehouseRegionCode

Функциональные зависимости:

● WarehouseId -> WarehouseName

● WarehouseId -> WarehouseType

● WarehouseId -> WarehouseAddress

● WarehouseId -> WarehouseTimezone

● WarehouseId -> WarehouseRegionCode

● WarehouseAddress, WarehouseRegionCode -> WarehouseId

Ключи:

● {WarehouseId}

## **Отношение WarehouseZone**

Атрибуты:

● WarehouseZoneId

● WarehouseZoneType

● WarehouseZoneName

● WarehouseId

● WarehouseZoneAccessLevel

Функциональные зависимости:

● WarehouseId, WarehouseZoneId -> WarehouseZoneName

● WarehouseId, WarehouseZoneId -> WarehouseZoneType

● WarehouseId, WarehouseZoneId -> WarehouseZoneAccessLevel

Ключи:

● {WarehouseId, WarehouseZoneId}

Нормализация:

1НФ:

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

2НФ:

Отношение в 2НФ, т.к. нет составных ключей, а значит неключевые атрибуты зависят от ключа в целом (не от части ключа).

А также уже в 1НФ.

3НФ:

Отношение 3НФ, т.к. неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.

А также уже в 2НФ.

НФБК:

Отношение находится в НФБК, т.к. в 3НФ и в каждой нетривиальной функциональной зависимости X -> Y, X является надключом.

4НФ:

Отношение в 4НФ, т.к. 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.

Найдем все нетривиальные зависимости соединений. Попытаемся разделить на разные по размеру части (на 2 нет смысла, т.к. это 4НФ).

Придется много перебрать, но в итоге получим, что итоговые таблицы не будут совпадать с изначальной, т.е. (pi(r1) njoin … njoin pi(rn)) != R, т.к. будут лишние строки. Значит отношение в 5НФ.

## **Отношение Position**

Атрибуты:

● PositionId

● PositionName

● PositionAccessLevel

● Salary

Функциональные зависимости:

● PositionId -> PositionName

● PositionId -> PositionAccessLevel

● PositionId -> Salary

● PositionName -> PositionId

Ключи:

● {PositionId}

● {PositionName}

Нормализация:

1НФ:

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

2НФ:

Отношение в 2НФ, т.к. нет составных ключей, а значит неключевые атрибуты зависят от ключа в целом (не от части ключа).

А также уже в 1НФ.

3НФ:

Отношение 3НФ, т.к. неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.

А также уже в 2НФ.

НФБК:

Отношение находится в НФБК, т.к. в 3НФ и в каждой нетривиальной функциональной зависимости X -> Y, X является надключом.

4НФ:

Отношение в 4НФ, т.к. по теореме Дейта-Фейгина 2: “Если отношение находится в НФБК существует простой ключ -> отношение находится в 4НФ”, а у нас есть простой ключ.

А также уже в НФБК.

5НФ:

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

А также уже в 3НФ.

## **Отношение StaffWorker**

Атрибуты:

● StaffWorkerId

● StaffWorkerName

● StaffWorkerGender

● StaffWorkerBirthDate

● StaffWorkerPassport

● StaffWorkerPositionId

● StaffWorkerPhoto

● StaffWorkerPhone

● StaffWorkerCreatedAt

● StaffWorkerFiredAt

● StaffWorkerWarehouseId

● StaffWorkerDeactivationReason

Функциональные зависимости:

● StaffWorkerId -> StaffWorkerName

● StaffWorkerId -> StaffWorkerGender

● StaffWorkerId -> StaffWorkerBirthDate

● StaffWorkerId -> StaffWorkerPassport

● StaffWorkerId -> StaffWorkerPositionId

● StaffWorkerId -> StaffWorkerPhoto

● StaffWorkerId -> StaffWorkerPhone

● StaffWorkerId -> StaffWorkerCreatedAt

● StaffWorkerId -> StaffWorkerFiredAt

● StaffWorkerId -> StaffWorkerWarehouseId

● StaffWorkerId -> StaffWorkerDeactivationReason

● StaffWorkerPassport -> StaffWorkerId

● StaffWorkerPhone -> StaffWorkerId

● StaffWorkerPhoto -> StaffWorkerId

Ключи:

● {StaffWorkerId}

● {StaffWorkerPassport}

● {StaffWorkerPhone}

● {StaffWorkerPhoto}

## **Отношение OutstaffWorker**

Атрибуты:

● OutstaffWorkerId

● OutstaffWorkerName

● OutstaffWorkerGender

● OutstaffWorkerBirthDate

● OutstaffWorkerPassport

● OutstaffWorkerPositionId

● OutstaffWorkerPhoto

● OutstaffWorkerPhone

● OutstaffWorkerCreatedAt

● OutstaffWorkerFiredAt

● OutstaffWorkerWarehouseId

● OutstaffWorkerDeactivationReason

● OutstaffWorkerCompanyId

● OutstaffWorkerContractStart

● OutstaffWorkerContractEnds

Функциональные зависимости:

● OutstaffWorkerId -> OutstaffWorkerName

● OutstaffWorkerId -> OutstaffWorkerGender

● OutstaffWorkerId -> OutstaffWorkerBirthDate

● OutstaffWorkerId -> OutstaffWorkerPassport

● OutstaffWorkerId -> OutstaffWorkerPositionId

● OutstaffWorkerId -> OutstaffWorkerPhoto

● OutstaffWorkerId -> OutstaffWorkerPhone

● OutstaffWorkerId -> OutstaffWorkerCreatedAt

● OutstaffWorkerId -> OutstaffWorkerFiredAt

● OutstaffWorkerId -> OutstaffWorkerWarehouseId

● OutstaffWorkerId -> OutstaffWorkerDeactivationReason

● OutstaffWorkerPassport -> OutstaffWorkerId

● OutstaffWorkerPhone -> OutstaffWorkerId

● OutstaffWorkerPhoto -> OutstaffWorkerId

Ключи:

● {OutstaffWorkerId}

● {OutstaffWorkerPassport}

● {OutstaffWorkerPhone}

● {OutstaffWorkerPhoto}

## **Отношение OutstaffCompany**

Атрибуты:

● OutstaffCompanyId

● OutstaffCompanyName

● OutstaffCompanyAddress

● OutstaffCompanyPhone

● OutstaffCompanyEmail

● OutstaffCompanyCreatedAt

● OutstaffCompanyContractStartDate

● OutstaffCompanyContractExpirationDate

Функциональные зависимости:

● OutstaffCompanyId -> OutstaffCompanyName

● OutstaffCompanyId -> OutstaffCompanyAddress

● OutstaffCompanyId -> OutstaffCompanyPhone

● OutstaffCompanyId -> OutstaffCompanyEmail

● OutstaffCompanyId -> OutstaffCompanyCreatedAt

● OutstaffCompanyId -> OutstaffCompanyContractStartDate

● OutstaffCompanyId -> OutstaffCompanyContractExpirationDate

● OutstaffCompanyPhone -> OutstaffCompanyId

● OutstaffCompanyEmail -> OutstaffCompanyId

● OutstaffCompanyName -> OutstaffCompanyId

Ключи:

● {OutstaffCompanyId}

● {OutstaffCompanyName}

● {OutstaffCompanyPhone}

● {OutstaffCompanyEmail}

Нормализация:

1НФ:

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

2НФ:

Отношение в 2НФ, т.к. нет составных ключей, а значит неключевые атрибуты зависят от ключа в целом (не от части ключа).

А также уже в 1НФ.

3НФ:

Отношение 3НФ, т.к. неключевые атрибуты непосредственно (не транзитивно) зависят от ключей.

А также уже в 2НФ.

НФБК:

Отношение находится в НФБК, т.к. в 3НФ и в каждой нетривиальной функциональной зависимости X -> Y, X является надключом.

4НФ:

Отношение в 4НФ, т.к. по теореме Дейта-Фейгина 2: “Если отношение находится в НФБК существует простой ключ -> отношение находится в 4НФ”, а у нас есть простой ключ.

А также уже в НФБК.

5НФ:

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

А также уже в 3НФ.

## **Отношение LogEntity**

Атрибуты:

● LogEntityId

● LogEntityStaffWorkerId

● LogEntityOutstaffWorkerId

● LogEntityCreatedAt

● LogEntityWarehouzeZoneId

● LogEntityWarehouseId

● LogEntityType

Функциональные зависимости:

● LogEntityId -> LogEntityStaffWorkerId

● LogEntityId -> LogEntityOutstaffWorkerId

● LogEntityId -> LogEntityCreatedAt

● LogEntityId -> LogEntityWarehouseZoneId

● LogEntityId -> LogEntityWarehouseId

● LogEntityId -> LogEntityType

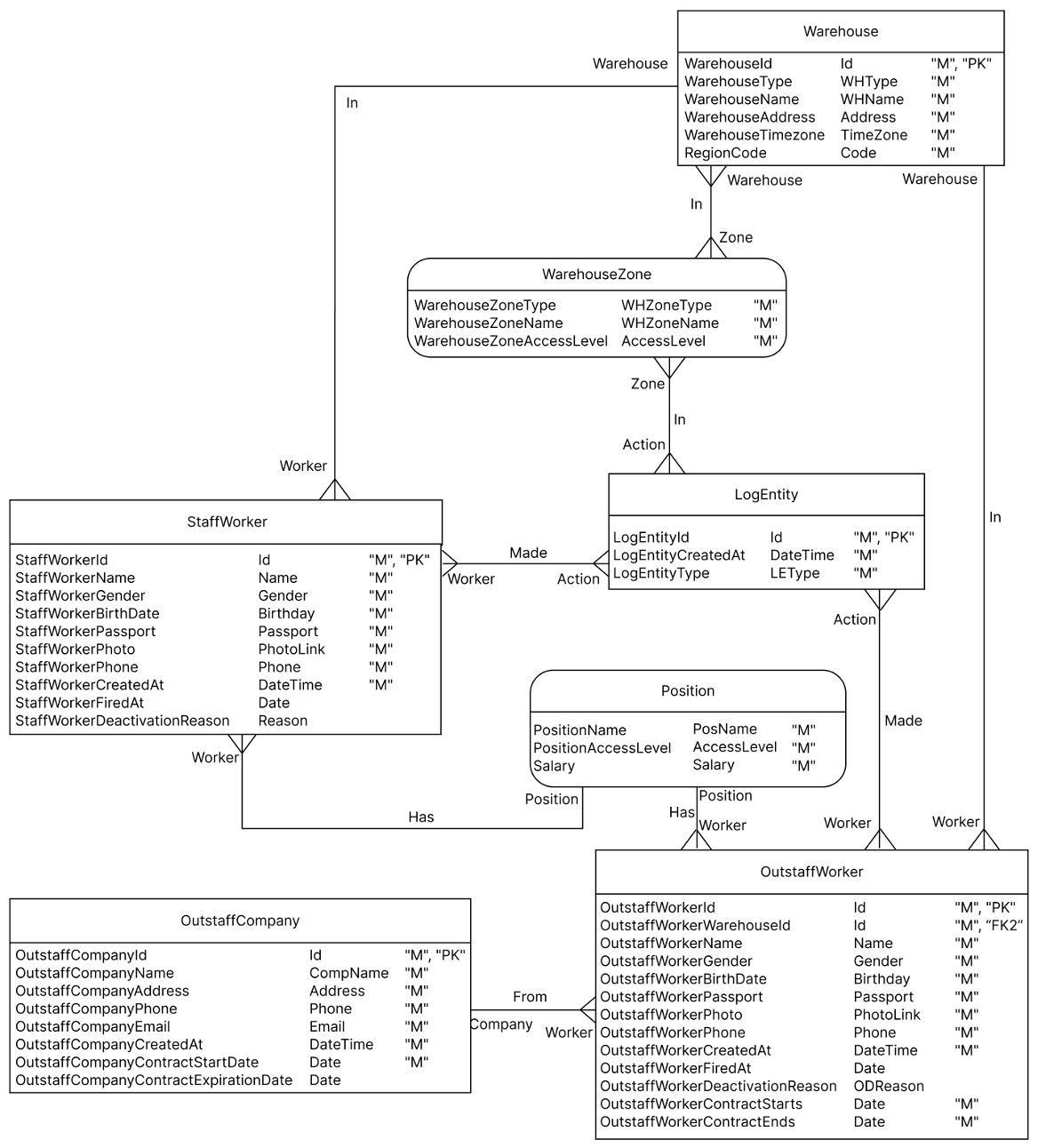
● LogEntityStaffWorkerId, LogEntityCreatedAt -> LogEntityId

● LogEntityOutstaffWorkerId, LogEntityCreatedAt -> LogEntityId

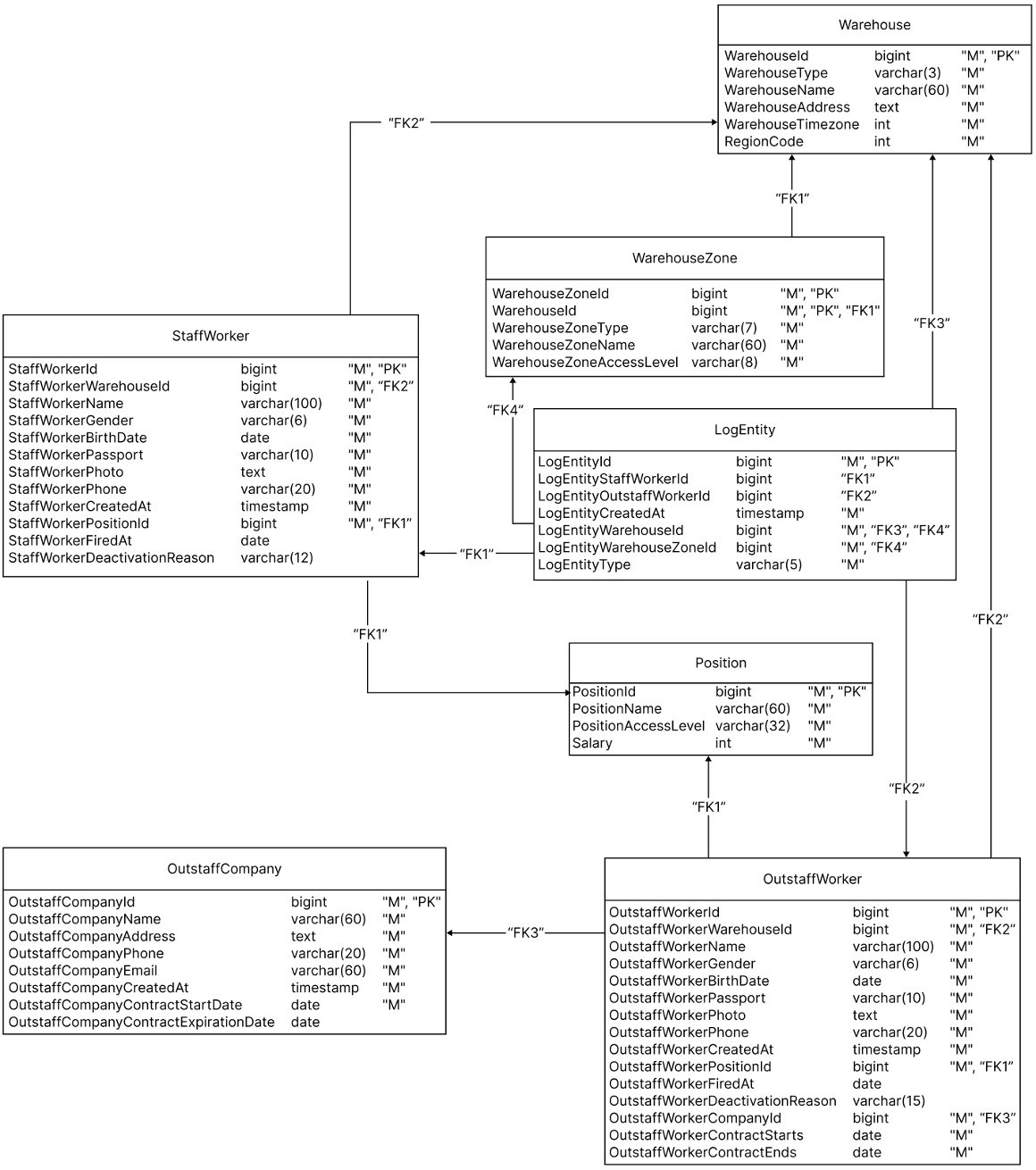
Ключи:

● {LogEntityId}

# Модель сущность-связь



# Физическая модель



При построении физической модели использовалось следующее отображение доменов в типы:

| Домен | Тип |
| --- | --- |
| Id | bigint |
| WHType | varchar(3) |
| WHName | varchar(60) |
| Address | text |
| TimeZone | int |
| Code | int |
| WHZoneType | varchar(7) |
| WHZoneName | varchar(60) |
| AccessLevel | varchar(8) |
| DateTime | timestamp |
| LEType | varchar(5) |
| PosName | varchar(60) |
| Salary | int |
| Name | varchar(100) |
| Gender | varchar(6) |
| Birthday | date |
| Passport | varchar(10) |
| PhotoLink | text |
| Phone | varchar(20) |
| Date | date |
| ODReason | varchar(15) |
| Reason | varchar(12) |
| CompName | varchar(60) |
| Email | varchar(60) |

# Определения таблиц

Для реализации проекта использовалась СУБД psql (PostgreSQL) 14. Определения таблиц и их индексов приведено в файле ddl.sql.

Все ограничения указаны внутри таблиц.

**Индексы:**

PostgreSQL автоматически создает индексы на primary key и unique. Дополнительные индексы объявлены в конце ddl.sql.

# Тестовые данные

Скрипт для добавления тестовых данных приведен в файле insert.sql.

# Запросы на получение данных

В рамках проекта были реализованы следующие запросы:

● **Найти штатных сотрудников на определенном объекте** — показывает список ФИО штатных сотрудников, на объекте, название которого указано в запросе.

● **Найти штатных сотрудников с определенным ФИО** — показывает список штатных сотрудников, чье ФИО содержит запрос, а также объект, на котором человек работает.

● **Найти количество внештатных сотрудников, нанятых каждой из компаний-партнеров** — показывает список из количества человек, нанятых каждой-партнером за все время.

● **Найти общее количество работающих сотрудников** — показывает суммарное количество работающих штатных и внештатных сотрудников по всем объектам.

● **Найти историю действий штатного сотрудника по ФИО** — показывает все перемещения штатного сотрудника, чье ФИО соответствует поисковому параметру. Также указывается название объекта, зоны объекта, задействован вход или выход и время создания записи лога.

● **Найти историю действий сотрудников в определенной зоне определенного объекта** — показывает записи логов всех сотрудников, кто совершал действия в указанной зоне объекта из поисковых параметров.

● **Найти количество заходивших людей для каждой зоны определенного объекта** — показывает для объекта, название которого передается в запросе, все зоны и количество посещений каждой из них за определенную дату.

● **Найти информацию о количестве входов и выходов каждой зоны на определенном объекте за определенный день** — показывает список из зон с указанием количества входов и выходов сотрудников для каждой зоны за определенную дату для определенного объекта соответственно.

Для реализации запросов были созданы вспомогательные представления:

● **warehousesStaff** — показывает список, содержащий в себе информацию об идентификаторе каждого штатного сотрудника, его ФИО, идентификатор объекта, на котором сотрудник работает, и его (объекта) название, а также информацию о дате деактивации записи сотрудника.

● **warehousesOutstaff** — показывает список, содержащий в себе информацию об идентификаторе каждого внештатного сотрудника, его ФИО, идентификатор объекта, на котором сотрудник работает, и его (объекта) название, а также информацию о дате окончания действия контракта с сотрудником о работе.

● **outstaffWorkersByCompanies** — показывает список, содержащий в себе информацию об идентификаторе каждого внештатного сотрудника, его ФИО, идентификатор компании-партнера, нанявшей его, а также название самой компании-партнера.

● **logsByStaffWorker** — показывает список, содержащий в себе записи логов о всех штатных сотрудниках и их перемещениях: идентификатор сотрудника, объекта и зоны объекта, время создания записи лога, тип записи лога (вход или выход), название объекта и зоны.

Запросы на получение данных и вспомогательные представления приведены в файле select.sql.

# Запросы на изменение данных

В рамках проекта были реализованы следующие запросы:

● **Добавить новую компанию-партнера** — при вызове **addOutstaffCompany**(...) в нашу БД будет добавлена новая компания для найма внештатных сотрудников, если она имеет уникальные юридическое название, телефон для связи и почтовый адрес, а также указаны все обязательные поля. (Пример для запуска на тестовых данных: select addOutstaffCompany(4, 'Тестовая компания', 'г. Тест, ул. Тестовая, д. 404', '+701234567890', 'noreply@testcomp.ru', '2023-05-21', '2025-05-21');)

● **Уволить штатного или внештатного сотрудника** — при вызове **fireWorker**(...) в нашу БД будет добавлена информация о деактивации записи сотрудника сегодняшним числом с указанной причиной увольнения. (Пример для запуска на тестовых данных: select fireworker(true, 1, 'incident');)

● **Создать запись под нового штатного или внештатного сотрудника** — при вызове **createWorker**(...) в нашу БД будет добавлена запись о новом сотруднике, при условии наличия всех обязательных полей и соблюдения уникальности полей для номера паспорта, ссылки на фото сотрудника и номера телефона для связи. (Пример для запуска на тестовых данных: select createWorker(false, 54, 'Вася Петров', 'male', '1990-01-01', '7898643543', 7, 'https://ourstorage.org/7898643543/image.jpeg', '+79637135889', 1, 1, '2024-01-01', '2025-01-01');)

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