**Ограничение**

ALTER TABLE Employee

ADD CONSTRAINT CHK\_Employee\_HireDate

CHECK (hire\_date >= '2000-01-01' AND hire\_date <= GETDATE());

ALTER TABLE Payroll

ADD CONSTRAINT CHK\_Payroll\_NetPayment

CHECK (net\_payment = total\_earnings - total\_deductions);

ALTER TABLE Payroll

ADD CONSTRAINT CHK\_Payroll\_Positive

CHECK (total\_earnings >= 0 AND total\_deductions >= 0 AND net\_payment >= 0);

**Обзор всех связей**

**1. Employee ↔ Department & Position**

*-- Сотрудник принадлежит к одному отделу*

department\_id INT NOT NULL,

CONSTRAINT fk\_department FOREIGN KEY (department\_id)

REFERENCES Department(department\_id)

*-- Сотрудник имеет одну должность*

position\_id INT NOT NULL,

CONSTRAINT fk\_position FOREIGN KEY (position\_id)

REFERENCES Position(position\_id)

**Тип связи:** One-to-Many (Один-ко-многим)

**Объяснение:**

* **Один** отдел может иметь **много** сотрудников
* **Одна** должность может быть у **многих** сотрудников
* **Один** сотрудник работает в **одном** отделе и имеет **одну** должность

**Пример:**

Отдел IT (1) → Иванов, Петров, Сидоров (N)

Должность Developer (1) → Иванов, Петров (N)

**2. Payroll ↔ Employee & PayPeriod**

*-- Ведомость принадлежит одному сотруднику*

employee\_id INT NOT NULL,

CONSTRAINT fk\_employee FOREIGN KEY (employee\_id)

REFERENCES Employee(employee\_id)

*-- Ведомость рассчитывается за один период*

period\_id INT NOT NULL,

CONSTRAINT fk\_period FOREIGN KEY (period\_id)

REFERENCES PayPeriod(period\_id)

**Тип связи:** One-to-Many (Один-ко-многим)

**Объяснение:**

* **Один** сотрудник может иметь **много** ведомостей (за разные месяцы)
* **Один** период может содержать **много** ведомостей (для разных сотрудников)
* **Одна** ведомость относится к **одному** сотруднику и **одному** периоду

**Пример:**

Сотрудник Иванов (1) → Ведомость за март, апрель, май (N)

Период Март 2024 (1) → Ведомости Иванова, Петрова, Сидоровой (N)

**3. Payroll\_Earning ↔ Payroll & Earning**

*-- Связь между ведомостью и начислениями*

PRIMARY KEY (payroll\_id, earning\_id),

CONSTRAINT fk\_pe\_payroll FOREIGN KEY (payroll\_id)

REFERENCES Payroll(payroll\_id) ON DELETE CASCADE,

CONSTRAINT fk\_pe\_earning FOREIGN KEY (earning\_id)

REFERENCES Earning(earning\_id)

**Тип связи:** Many-to-Many (Многие-ко-многим) через промежуточную таблицу

**Объяснение:**

* **Одна** ведомость может содержать **много** начислений
* **Один** тип начисления может использоваться в **многих** ведомостях
* **Промежуточная таблица** Payroll\_Earning связывает их

**Пример:**

Ведомость №1 (1) → Оклад + Премия + Надбавка (N)

Тип начисления "Премия" (1) → Ведомость №1, Ведомость №2, Ведомость №3 (N)

**4. Payroll\_Deduction ↔ Payroll & Deduction**

*-- Связь между ведомостью и удержаниями*

PRIMARY KEY (payroll\_id, deduction\_id),

CONSTRAINT fk\_pd\_payroll FOREIGN KEY (payroll\_id)

REFERENCES Payroll(payroll\_id) ON DELETE CASCADE,

CONSTRAINT fk\_pd\_deduction FOREIGN KEY (deduction\_id)

REFERENCES Deduction(deduction\_id)

**Тип связи:** Many-to-Many (Многие-ко-многим) через промежуточную таблицу

**Объяснение:**

* **Одна** ведомость может содержать **много** удержаний
* **Один** тип удержания может использоваться в **многих** ведомостях

**Пример:**

Ведомость №1 (1) → НДФЛ + Алименты + Профсоюз (N)

Тип удержания "НДФЛ" (1) → Все ведомости всех сотрудников (N)

**Описание таблиц**

**🗂️ 1. Таблица Department (Отделы)**

sql

CREATE TABLE Department (

department\_id SERIAL PRIMARY KEY, *-- PK: Уникальный идентификатор отдела*

name VARCHAR(100) NOT NULL, *-- UNIQUE: Название отдела (должно быть уникальным)*

phone VARCHAR(20) *-- Обычное поле*

);

**Ключи:**

* **PK:** department\_id (автоинкремент)
* **UNIQUE:** name (название отдела должно быть уникальным)

**🗂️ 2. Таблица Position (Должности)**

sql

CREATE TABLE Position (

position\_id SERIAL PRIMARY KEY, *-- PK: Уникальный идентификатор должности*

title VARCHAR(100) NOT NULL, *-- UNIQUE: Название должности (уникальное)*

base\_salary DECIMAL(10,2) NOT NULL *-- Обычное поле: базовая зарплата*

);

**Ключи:**

* **PK:** position\_id (автоинкремент)
* **UNIQUE:** title (название должности должно быть уникальным)

**🗂️ 3. Таблица Employee (Сотрудники)**

sql

CREATE TABLE Employee (

employee\_id SERIAL PRIMARY KEY, *-- PK: Уникальный идентификатор сотрудника*

full\_name VARCHAR(150) NOT NULL, *-- Обычное поле: ФИО*

hire\_date DATE NOT NULL, *-- Обычное поле: дата приема*

bank\_account VARCHAR(50) NOT NULL, *-- UNIQUE: номер счета (уникальный)*

department\_id INT NOT NULL, *-- FK: ссылка на отдел*

position\_id INT NOT NULL, *-- FK: ссылка на должность*

CONSTRAINT fk\_department FOREIGN KEY (department\_id)

REFERENCES Department(department\_id), *-- FK к Department*

CONSTRAINT fk\_position FOREIGN KEY (position\_id)

REFERENCES Position(position\_id) *-- FK к Position*

);

**Ключи:**

* **PK:** employee\_id (автоинкремент)
* **FK:** department\_id → Department(department\_id)
* **FK:** position\_id → Position(position\_id)
* **UNIQUE:** bank\_account (банковский счет должен быть уникальным)

**🗂️ 4. Таблица PayPeriod (Расчетные периоды)**

sql

CREATE TABLE PayPeriod (

period\_id SERIAL PRIMARY KEY, *-- PK: Уникальный идентификатор периода*

month INT NOT NULL CHECK (month BETWEEN 1 AND 12),

year INT NOT NULL,

status VARCHAR(20) NOT NULL *-- Обычное поле: статус периода*

*-- UNIQUE: month+year (логически уникальная комбинация)*

);

**Ключи:**

* **PK:** period\_id (автоинкремент)
* **UNIQUE:** комбинация month + year (логическое ограничение)

**🗂️ 5. Таблица Payroll (Расчеты зарплаты)**

sql

CREATE TABLE Payroll (

payroll\_id SERIAL PRIMARY KEY, *-- PK: Уникальный идентификатор ведомости*

employee\_id INT NOT NULL, *-- FK: ссылка на сотрудника*

period\_id INT NOT NULL, *-- FK: ссылка на период*

calculation\_date DATE NOT NULL,

total\_earnings DECIMAL(10,2) NOT NULL DEFAULT 0,

total\_deductions DECIMAL(10,2) NOT NULL DEFAULT 0,

net\_payment DECIMAL(10,2) NOT NULL DEFAULT 0,

CONSTRAINT fk\_employee FOREIGN KEY (employee\_id)

REFERENCES Employee(employee\_id), *-- FK к Employee*

CONSTRAINT fk\_period FOREIGN KEY (period\_id)

REFERENCES PayPeriod(period\_id) *-- FK к PayPeriod*

*-- UNIQUE: employee\_id+period\_id (один расчет на сотрудника за период)*

);

**Ключи:**

* **PK:** payroll\_id (автоинкремент)
* **FK:** employee\_id → Employee(employee\_id)
* **FK:** period\_id → PayPeriod(period\_id)
* **UNIQUE:** комбинация employee\_id + period\_id (логическое ограничение)

**🗂️ 6. Таблица Earning (Справочник начислений)**

sql

CREATE TABLE Earning (

earning\_id SERIAL PRIMARY KEY, *-- PK: Уникальный идентификатор начисления*

type VARCHAR(100) NOT NULL, *-- UNIQUE: тип начисления (уникальный)*

amount\_or\_percent DECIMAL(10,2) NOT NULL *-- Обычное поле: сумма или процент*

);

**Ключи:**

* **PK:** earning\_id (автоинкремент)
* **UNIQUE:** type (тип начисления должен быть уникальным)

**🗂️ 7. Таблица Deduction (Справочник удержаний)**

sql

CREATE TABLE Deduction (

deduction\_id SERIAL PRIMARY KEY, *-- PK: Уникальный идентификатор удержания*

type VARCHAR(100) NOT NULL, *-- UNIQUE: тип удержания (уникальный)*

amount\_or\_percent DECIMAL(10,2) NOT NULL *-- Обычное поле: сумма или процент*

);

**Ключи:**

* **PK:** deduction\_id (автоинкремент)
* **UNIQUE:** type (тип удержания должен быть уникальным)

**🗂️ 8. Таблица Payroll\_Earning (Начисления в зарплате)**

sql

CREATE TABLE Payroll\_Earning (

payroll\_id INT NOT NULL, *-- PK часть + FK: ссылка на ведомость*

earning\_id INT NOT NULL, *-- PK часть + FK: ссылка на тип начисления*

actual\_amount DECIMAL(10,2) NOT NULL, *-- Обычное поле: фактическая сумма*

PRIMARY KEY (payroll\_id, earning\_id), *-- COMPOSITE PK: составной первичный ключ*

CONSTRAINT fk\_pe\_payroll FOREIGN KEY (payroll\_id)

REFERENCES Payroll(payroll\_id) ON DELETE CASCADE, *-- FK к Payroll*

CONSTRAINT fk\_pe\_earning FOREIGN KEY (earning\_id)

REFERENCES Earning(earning\_id) *-- FK к Earning*

);

**Ключи:**

* **PK:** (payroll\_id, earning\_id) (составной первичный ключ)
* **FK:** payroll\_id → Payroll(payroll\_id)
* **FK:** earning\_id → Earning(earning\_id)

**🗂️ 9. Таблица Payroll\_Deduction (Удержания в зарплате)**

sql

CREATE TABLE Payroll\_Deduction (

payroll\_id INT NOT NULL, *-- PK часть + FK: ссылка на ведомость*

deduction\_id INT NOT NULL, *-- PK часть + FK: ссылка на тип удержания*

actual\_amount DECIMAL(10,2) NOT NULL, *-- Обычное поле: фактическая сумма*

PRIMARY KEY (payroll\_id, deduction\_id), *-- COMPOSITE PK: составной первичный ключ*

CONSTRAINT fk\_pd\_payroll FOREIGN KEY (payroll\_id)

REFERENCES Payroll(payroll\_id) ON DELETE CASCADE, *-- FK к Payroll*

CONSTRAINT fk\_pd\_deduction FOREIGN KEY (deduction\_id)

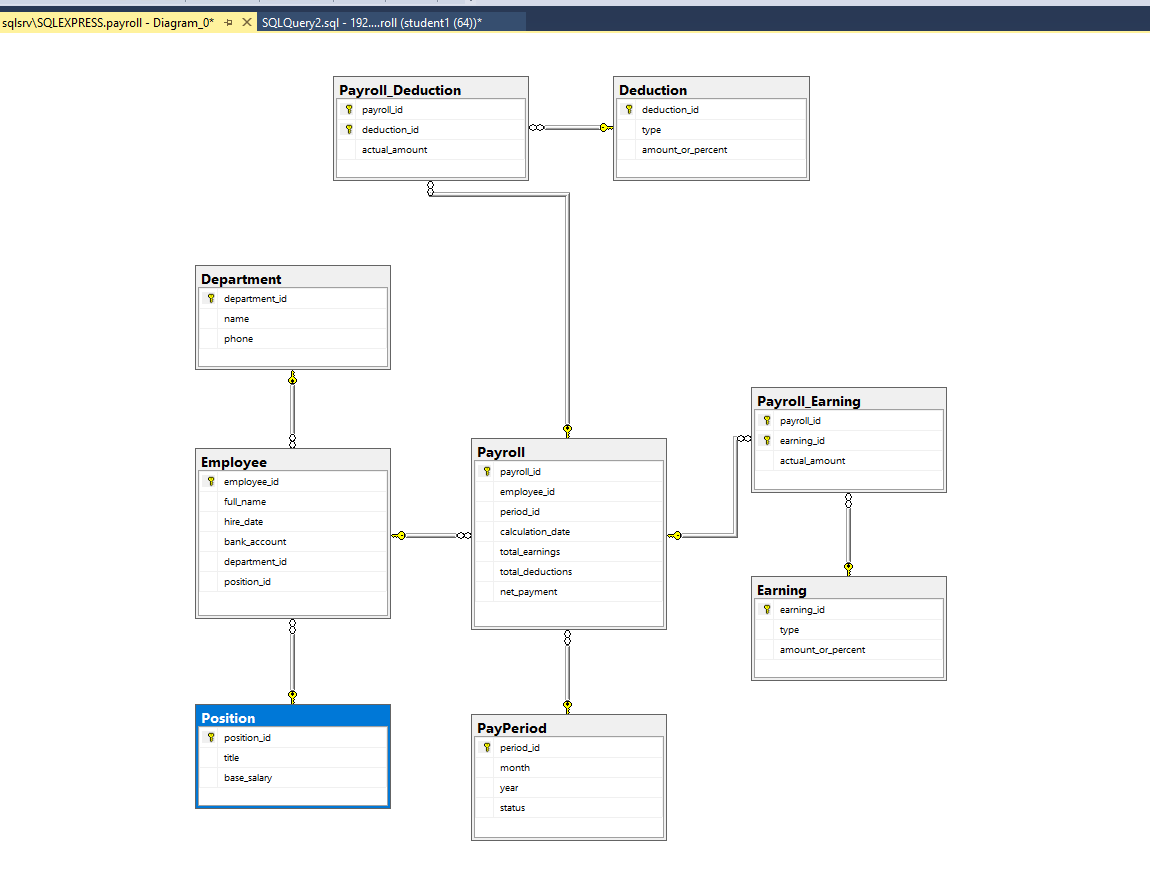
REFERENCES Deduction(deduction\_id) *-- FK к Deduction*

);

**Ключи:**

* **PK:** (payroll\_id, deduction\_id) (составной первичный ключ)
* **FK:** payroll\_id → Payroll(payroll\_id)
* **FK:** deduction\_id → Deduction(deduction\_id)

1. **Справочные таблицы** (Department, Position, Earning, Deduction) имеют только PK и UNIQUE
2. **Основные таблицы** (Employee, Payroll) имеют PK и FK
3. **Промежуточные таблицы** (Payroll\_Earning, Payroll\_Deduction) имеют составные PK и FK
4. **ON DELETE CASCADE** в промежуточных таблицах обеспечивает целостность данных

**Диограмма sql**