**Предметная область:**

«Разработка БД городского краеведческого музея»

**Описание предметной области:**

Целью проекта является создание базы данных для хранения информации о коллекциях музея, сотрудниках, посетителях, событиях и других релевантных данных для эффективного управления музеем и предоставления удобного доступа к информации организации.

Краеведческий музей – это музей, который фокусируется на истории, культуре, природе и искусстве конкретного региона или местности.

**База данных:**

Описание базы данных:

• Экспонат:

1. ID экспоната (PK)
2. Наименование экспоната
3. Описание экспоната
4. Дата создания/нахождения
5. Место создания/происхождения
6. Фотография
7. Документация
8. ID типа экспоната (FK)

• Коллекция:

1. ID коллекции (PK)
2. Наименование коллекции
3. Описание коллекции
4. Дата создания
5. ID тематики коллекции (FK)

• Сотрудник:

1. ID сотрудника (PK)
2. Фамилия
3. Имя
4. Отчество
5. Дата рождения
6. Телефон
7. Электронная почта
8. ID должности (FK)

• Посетитель:

1. ID посетителя (PK)
2. Фамилия
3. Имя
4. Отчество
5. Возраст
6. Адрес
7. Телефон
8. Электронная почта
9. Дата визита

• Выставка:

1. ID выставки (PK)
2. Наименование выставки
3. Описание выставки
4. Дата открытия
5. Дата закрытия
6. ID пространства (FK)
7. ID сотрудника (FK)

• Билет:

1. ID билета (PK)
2. Стоимость
3. Дата продажи
4. ID пространства (FK)
5. ID посетителя (FK)
6. ID типа билета (FK)

• Пространство:

1. ID пространства (PK)
2. Наименование пространства
3. Площадь
4. Местоположение

• Событие:

1. ID события (PK)
2. Наименование события
3. Дата проведения
4. Время проведения
5. ID пространства (FK)
6. ID сотрудника (FK)
7. ID типа события (FK)

• Товар:

1. ID товара (PK)
2. Наименование товара
3. Описание товара
4. Цена
5. Количество на складе
6. ID посетителя (FK)

• Тип экспоната:

1. ID типа экспоната (PK)
2. Название типа экспоната

• Тематика коллекции:

1. ID тематики коллекции (PK)
2. Название тематики коллекции

• Тип события:

1. ID типа события (PK)
2. Название типа события

• Тип билета:

1. ID типа билета (PK)
2. Название типа билета

• Должность:

1. ID должности (PK)
2. Название должности

Ассоциативные сущности:

• Экспозиция (Экспонат; Выставка):

1. ID выставки (FK)
2. ID экспоната (FK)
3. Статус

• Состав коллекции (Экспонат; Коллекция)

1. ID экспоната (FK)
2. ID коллекции (FK)
3. Комментарий

Основные сущности:

* Экспонат
* Коллекция
* Сотрудник
* Посетитель
* Выставка
* Билет
* Пространство
* Событие
* Товар
* Тип экспоната
* Тематика коллекции
* Тип события
* Тип билета
* Должность

Основные связи между сущностями:

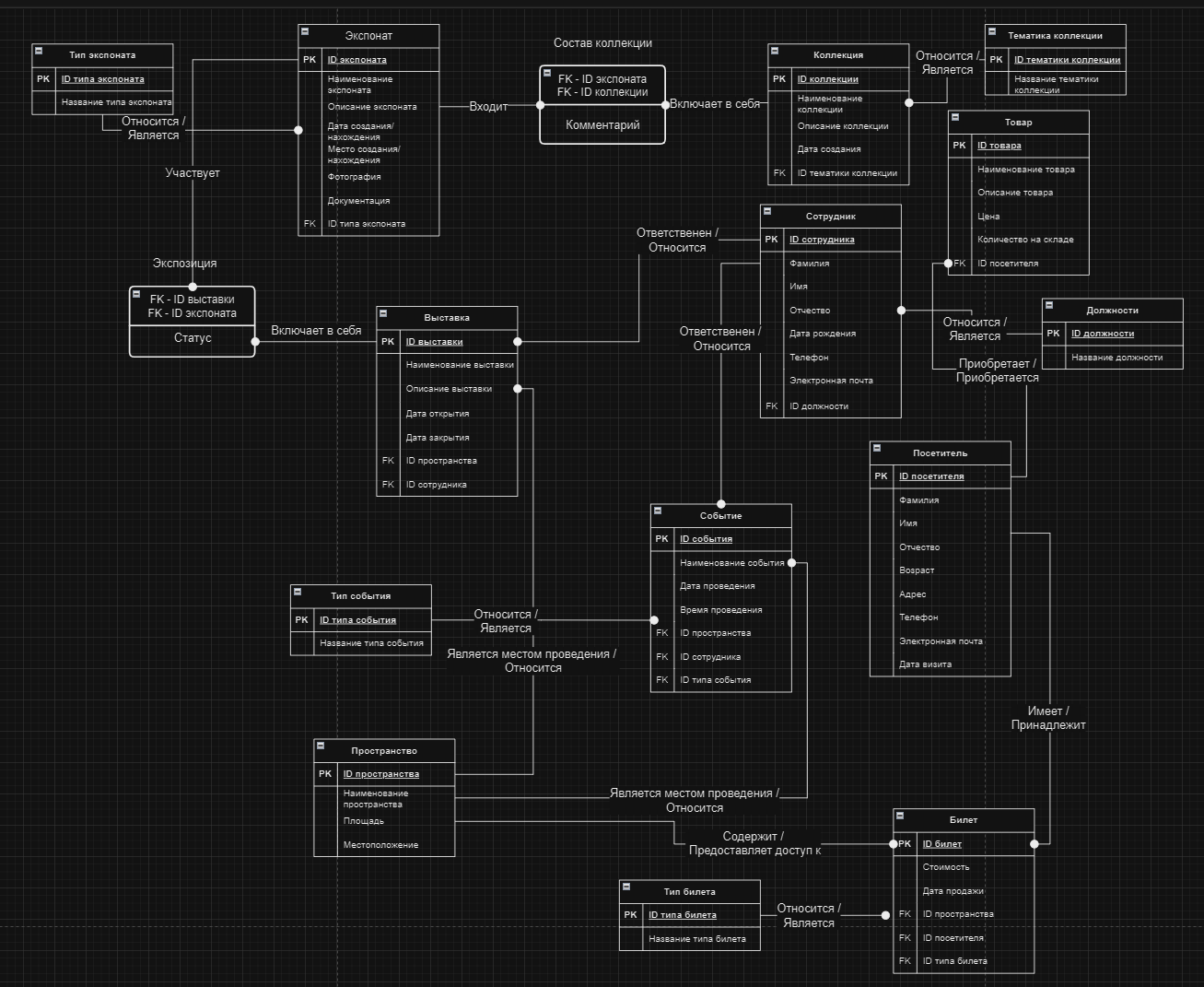
*1:M – (один ко многим):*

* Коллекция -> Экспонат (Одна коллекция может содержать множество экспонатов)
* Сотрудник -> Выставка (Один сотрудник может быть ответственным за несколько выставок)
* Посетитель -> Билет (Один посетитель может купить несколько билетов)
* Выставка -> Экспонат (Одна выставка может включать множество экспонатов)
* Сотрудник -> Событие (Один сотрудник может быть ответственным за несколько событий)
* Пространство -> Выставка (Одно пространство может быть местом проведения нескольких выставок)
* Пространство -> Событие (Одно пространство может быть местом проведения нескольких событий)
* Пространство -> Билет (Одно пространство может содержать множество билет на него)
* Тип экспоната -> Экспонат (Один тип может иметь множество экземпляров)
* Тематика коллекции -> Коллекция (Один тип может иметь множество экземпляров)
* Тип события -> Событие (Один тип может иметь множество экземпляров)
* Должность -> Сотрудник (Один тип может иметь множество экземпляров)
* Тип билета -> Билет (Один тип может иметь множество реализаций/экземпляров. А экземпляр может иметь только один тип)

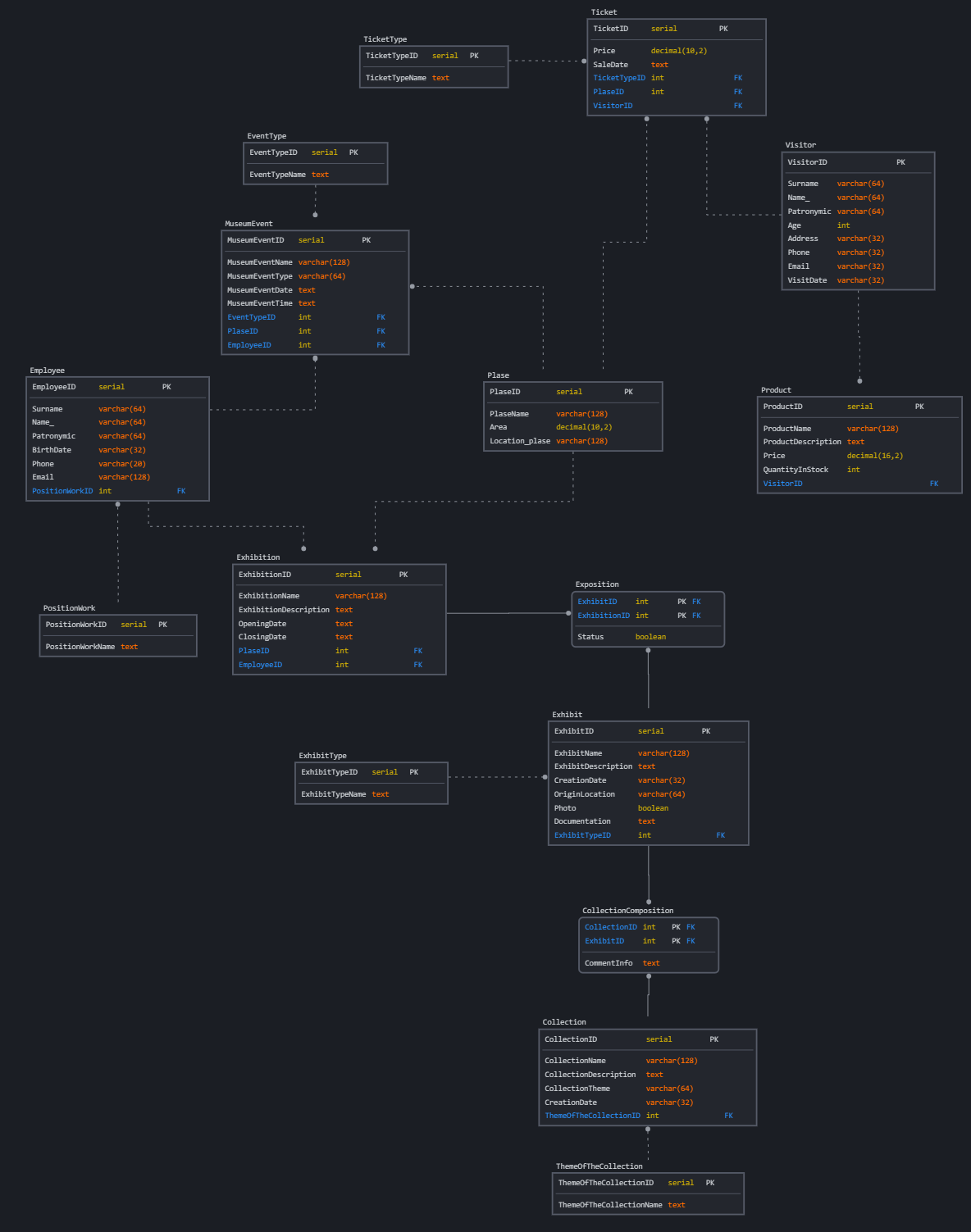
*M:M – (многие ко многим):*

* Экспонат -> Выставка (Один экспонат может участвовать в нескольких выставках, а одна выставка может включать множество экспонатов)
* Экспонат -> Коллекция (Один экспонат может входить в состав нескольких коллекций, а одна коллекция может включать множество экспонатов)

**Логическая модель базы данных:**

****

**Физическая модель базы данных:**



**Задание 1:**

SQL-скрипты:

*Шаг 1 - PostgreSQL (Создание базы данных):*

-- Создание базы данных

CREATE DATABASE region\_history\_museum

WITH

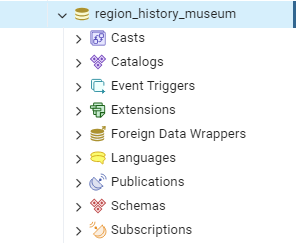
OWNER = postgres

ENCODING = 'UTF8'

LOCALE\_PROVIDER = 'libc'

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;



==================================================================================================

*Шаг 2 - PostgreSQL (Создание таблиц, запись первичных и внешних ключей):*

-- Создание таблицы "Должность"

CREATE TABLE PositionWork

(

PositionWorkID SERIAL PRIMARY KEY,

PositionWorkName text NOT NULL UNIQUE

);

-- Создание таблицы "Тип экспоната"

CREATE TABLE ExhibitType

(

ExhibitTypeID SERIAL PRIMARY KEY,

ExhibitTypeName text NOT NULL UNIQUE

);

-- Создание таблицы "Тематика коллекции"

CREATE TABLE ThemeOfTheCollection

(

ThemeOfTheCollectionID SERIAL PRIMARY KEY,

ThemeOfTheCollectionName text NOT NULL UNIQUE

);

-- Создание таблицы "Тип события"

CREATE TABLE EventType

(

EventTypeID SERIAL PRIMARY KEY,

EventTypeName text NOT NULL UNIQUE

);

-- Создание таблицы "Тип билета"

CREATE TABLE TicketType

(

TicketTypeID SERIAL PRIMARY KEY,

TicketTypeName text NOT NULL UNIQUE

);

-- Создание таблицы "Экспонат"

CREATE TABLE Exhibit

(

ExhibitID SERIAL PRIMARY KEY,

ExhibitName varchar(128) NOT NULL,

ExhibitDescription text,

CreationDate varchar(32) NOT NULL,

OriginLocation varchar(64),

Photo boolean,

Documentation text,

ExhibitTypeID\_FK integer NOT NULL,

FOREIGN KEY (ExhibitTypeID\_FK) REFERENCES ExhibitType(ExhibitTypeID)

);

-- Создание таблицы "Коллекция"

CREATE TABLE Collection

(

CollectionID SERIAL PRIMARY KEY,

CollectionName varchar(128) NOT NULL,

CollectionDescription text,

CollectionTheme varchar(64) NOT NULL,

CreationDate varchar(32) NOT NULL,

ThemeOfTheCollectionID\_FK integer NOT NULL,

FOREIGN KEY (ThemeOfTheCollectionID\_FK) REFERENCES ThemeOfTheCollection(ThemeOfTheCollectionID)

);

-- Создание таблицы "Сотрудник"

CREATE TABLE Employee

(

EmployeeID SERIAL PRIMARY KEY,

Surname varchar(64) NOT NULL,

Name\_ varchar(64) NOT NULL,

Patronymic varchar(64) NOT NULL,

BirthDate varchar(32),

Phone varchar(20) NOT NULL UNIQUE,

Email varchar(128) UNIQUE,

PositionWorkID\_FK integer NOT NULL,

FOREIGN KEY (PositionWorkID\_FK) REFERENCES PositionWork(PositionWorkID)

);

-- Создание таблицы "Посетитель"

CREATE TABLE Visitor

(

VisitorID SERIAL PRIMARY KEY,

Surname varchar(64) NOT NULL,

Name\_ varchar(64) NOT NULL,

Patronymic varchar(64) NOT NULL,

Age integer,

Address varchar(64),

Phone varchar(20) NOT NULL UNIQUE,

Email varchar(128) UNIQUE,

VisitDate varchar(32) NOT NULL

);

-- Создание таблицы "Пространство"

CREATE TABLE Plase

(

PlaseID SERIAL PRIMARY KEY,

PlaseName varchar(128) NOT NULL UNIQUE,

Area decimal(10, 2) CHECK (Area > 0),

Location\_plase varchar(128)

);

-- Создание таблицы "Выставка"

CREATE TABLE Exhibition

(

ExhibitionID SERIAL PRIMARY KEY,

ExhibitionName varchar(128) NOT NULL,

ExhibitionDescription text,

OpeningDate text,

ClosingDate text,

PlaseID\_FK integer NOT NULL,

EmployeeID\_FK integer NOT NULL,

FOREIGN KEY (EmployeeID\_FK) REFERENCES Employee(EmployeeID),

FOREIGN KEY (PlaseID\_FK) REFERENCES Plase(PlaseID)

);

-- Создание таблицы "Билет"

CREATE TABLE Ticket

(

TicketID SERIAL PRIMARY KEY,

Price decimal(10, 2) NOT NULL,

SaleDate text NOT NULL,

PlaseID\_FK integer NOT NULL,

VisitorID\_FK integer NOT NULL,

TicketTypeID\_FK integer NOT NULL,

FOREIGN KEY (TicketTypeID\_FK) REFERENCES TicketType(TicketTypeID),

FOREIGN KEY (PlaseID\_FK) REFERENCES Plase(PlaseID),

FOREIGN KEY (VisitorID\_FK) REFERENCES Visitor(VisitorID)

);

-- Создание таблицы "Событие"

CREATE TABLE MuseumEvent

(

MuseumEventID SERIAL PRIMARY KEY,

MuseumEventName varchar(128) NOT NULL,

MuseumEventDate text NOT NULL,

MuseumEventTime text NOT NULL,

PlaseID\_FK integer NOT NULL,

EmployeeID\_FK integer NOT NULL,

EventTypeID\_FK integer NOT NULL,

FOREIGN KEY (EventTypeID\_FK) REFERENCES EventType(EventTypeID),

FOREIGN KEY (PlaseID\_FK) REFERENCES Plase(PlaseID),

FOREIGN KEY (EmployeeID\_FK) REFERENCES Employee(EmployeeID)

);

-- Создание таблицы "Товар"

CREATE TABLE Product

(

ProductID SERIAL PRIMARY KEY,

ProductName varchar(128) NOT NULL,

ProductDescription text,

Price decimal(16, 2) NOT NULL CHECK (Price > 0),

QuantityInStock integer NOT NULL,

VisitorID\_FK integer,

FOREIGN KEY (VisitorID\_FK) REFERENCES Visitor(VisitorID)

);

-- Создание таблицы "Экспозиция" (Экспонат; Выставка)

CREATE TABLE Exposition

(

ExhibitionID\_FK integer NOT NULL,

ExhibitID\_FK integer NOT NULL,

Status boolean NOT NULL,

FOREIGN KEY (ExhibitionID\_FK) REFERENCES Exhibition(ExhibitionID),

FOREIGN KEY (ExhibitID\_FK) REFERENCES Exhibit(ExhibitID),

CONSTRAINT ExhibitionExhibitID PRIMARY KEY (ExhibitionID\_FK, ExhibitID\_FK)

);

-- Создание таблицы "Состав коллекции" (Экспонат; Коллекция)

CREATE TABLE CollectionComposition

(

ExhibitID\_FK integer NOT NULL,

CollectionID\_FK integer NOT NULL,

CommentInfo text,

FOREIGN KEY (ExhibitID\_FK) REFERENCES Exhibit(ExhibitID),

FOREIGN KEY (CollectionID\_FK) REFERENCES Collection(CollectionID),

CONSTRAINT ExhibitionCollectionID PRIMARY KEY (ExhibitID\_FK, CollectionID\_FK)

);

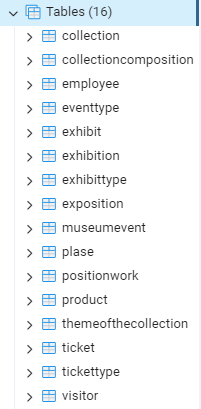
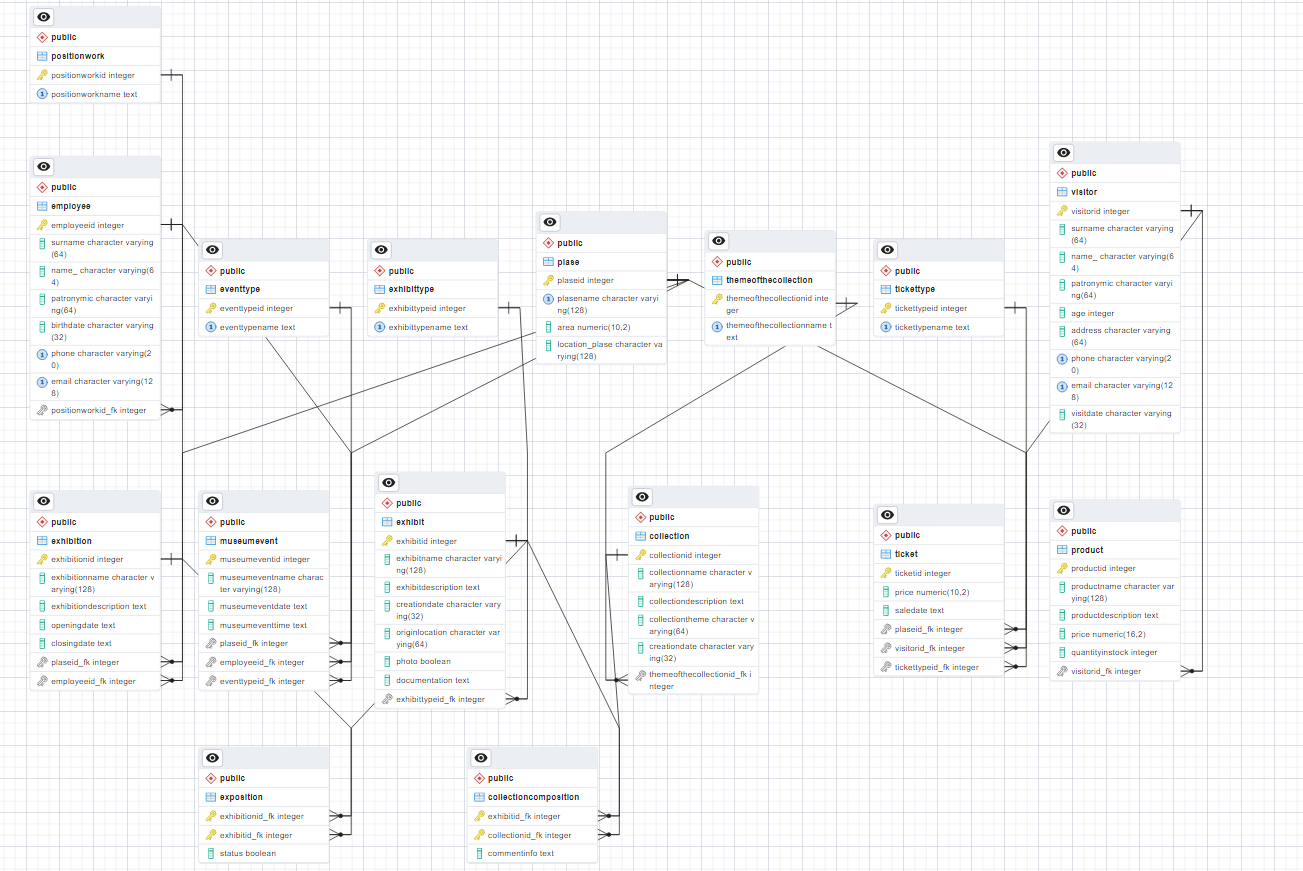


Диаграмма БД (ERD For Database):

~~~~

Доказательство 3НФ:

• 1-ая нормальная форма:

- *Нет повторяющихся строк*

- *Все атрибуты простые.* Простой атрибут в базах данных – это атрибут, который не может быть разбит на более мелкие части. Другими словами, он не содержит в себе других атрибутов. Пример простого атрибута «Имя: "Мария"». Пример сложного атрибута «ФИО: "Кирилл Кириллов Кириллович”, который можно разбить на "фамилия", "имя", "отчество".»

- *Все значения скалярные.* Атрибут является скалярным, если он может хранить только одно значение за раз в одной ячейке. Другими словами, атрибут не может иметь список значений.

• 2-ая нормальная форма:

- *Таблица должна быть приведена к 1НФ*

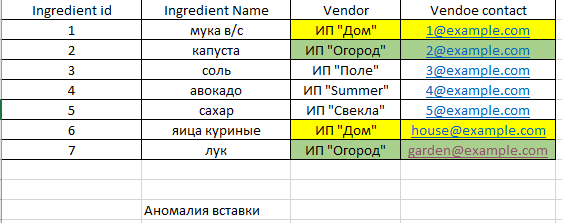
- *У таблицы должен быть первичный ключ*

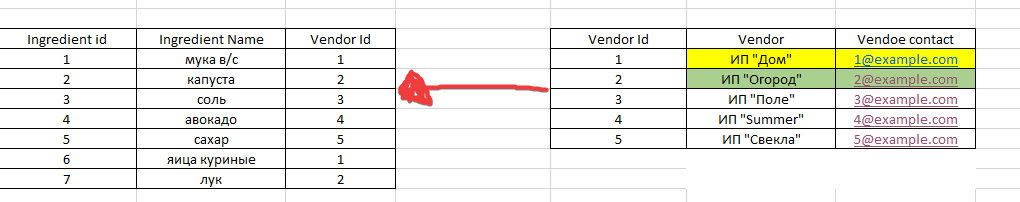
- *Все атрибуты должны описывать первичный ключ целиком, а не какую-то часть первичного ключа.* Проще говоря, в таблице, находящейся в 2НФ, информация, которая не является частью первичного ключа, должна зависеть от всей информации, составляющей первичный ключ, а не только от части его.

• 3-ая нормальная форма:

- *Таблица должна быть приведена к 2НФ*

- *Не должно быть зависимостей одних не ключевых атрибутов от других. Все атрибуты зависят только от первичного ключа.*





Инструкции SQL: *Описаны выше*

Обеспечение целостности (+ обоснование):

Обеспечение целостности в базах данных – это набор мер, направленных на сохранение точности, полноты и согласованности данных в базе данных.

- Каждая запись в таблице должна иметь уникальный идентификатор (первичный ключ).

- Ограничения типа данных: запрещается ввод данных несоответствующего типа (например, текст вместо числа).

- Ограничения проверки: проверяют, соответствуют ли данные определенным условиям (например, возраст должен быть больше 18).

- Ограничения внешнего ключа: обеспечивают целостность связей между таблицами.

- Ограничения уникальности: запрещают дублирование данных в столбце.

- Ограничения проверки нулевых значений: запрещают или допускают ввод нулевых значений в столбец.

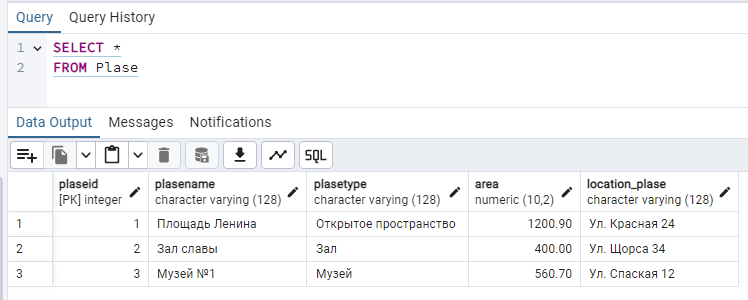
- Атомарность. Атомарность означает, что транзакция либо выполняется полностью, либо не выполняется вовсе.

- Изоляция: транзакции выполняются независимо друг от друга, не влияя на результаты других.

==================================================================================================

**Задание 2:**

*Шаг 3 - PostgreSQL (Добавление данных в таблицы):*



-- Передавать значения в той последовательности, в которой у нас идут столбцы

-- Вставка: INSERT INTO Name\_table VALUES (Data);

-- Вывод данных таблицы: 1) SELECT \* 2) FROM Name\_table

-- Вставка данных в таблицу "Должность"

INSERT INTO PositionWork

VALUES

(1, 'Кассир'),

(2, 'Менеджер'),

(3, 'Регистратор'),

(4, 'Гид экскурсовод'),

(5, 'Охранник');

-- Вставка данных в таблицу "Тип экспоната"

INSERT INTO ExhibitType

VALUES

(1, 'Археология'),

(2, 'Этнография'),

(3, 'Картография'),

(4, 'Изобразительное искусство'),

(5, 'Фольклор');

-- Вставка данных в таблицу "Тематика коллекции"

INSERT INTO ThemeOfTheCollection

VALUES

(1, 'Нумизматика'),

(2, 'Филателия'),

(3, 'Литература'),

(4, 'Искусство'),

(5,'Декоративно-прикладное искусство');

-- Вставка данных в таблицу "Тип события"

INSERT INTO EventType

VALUES

(1, 'Выставка'),

(2, 'Лекция'),

(3, 'Концерт'),

(4, 'Мастер-класс'),

(5, 'Театр');

-- Вставка данных в таблицу "Тип билета"

INSERT INTO TicketType

VALUES

(1, 'Стандартный'),

(2, 'VIP'),

(3, 'Детский'),

(4, 'Студенческий'),

(5, 'Семейный');

-- Вставка данных в таблицу "Пространство"

INSERT INTO Plase

VALUES

('1', 'Площадь Ленина', 1200.90, 'Ул. Красная 24'),

('2', 'Зал славы', 400.00, 'Ул. Щорса 34'),

('3', 'Музей №1', 560.70, 'Ул. Спаская 12');

-- Вставка данных в таблицу "Посетитель"

INSERT INTO Visitor

VALUES

(1, 'Иванов', 'Иван', 'Иванович', 35, 'ул. Пушкина, д. 10', '+79123456789', 'ivan.ivanov@example.com', '2023-03-15'),

(2, 'Петрова', 'Мария', 'Петрович', 28, 'ул. Лермонтова, д. 5', '+79876543210', 'maria.petrova@example.com', '2023-03-16'),

(3, 'Сидоров', 'Александр', 'Сидорович', 14, 'ул. Толстого, д. 20', '+79121234567', 'alexandr.sidorov@example.com', '2023-03-17'),

(4, 'Козлова', 'Ольга', 'Олеговна', 25, 'ул. Чехова, д. 15', '+79879876543', 'olga.kozlova@example.com', '2023-03-18'),

(5, 'Федоров', 'Дмитрий', 'Петрович', 30, 'ул. Гоголя, д. 8', '+79127894567', 'dmitriy.fedorov@example.com', '2023-03-19');

-- Вставка данных в таблицу "Билет"

/\*

INSERT INTO Ticket

VALUES

(1, 15.50, '2023-03-15', 1, 1, 1),

(2, 30.00, '2023-03-16', 2, 2, 2),

(3, 7.75, '2023-03-17', 3, 3, 3),

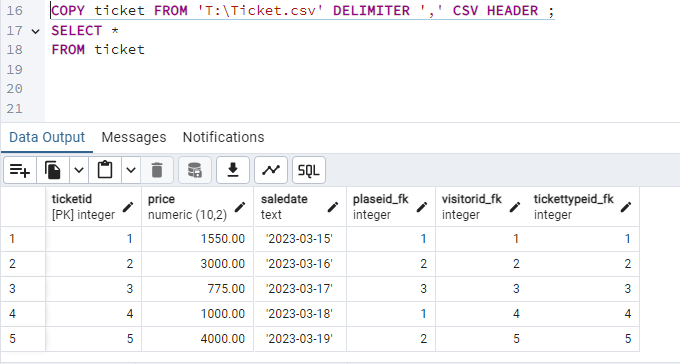
(4, 10.00, '2023-03-18', 1, 4, 4),

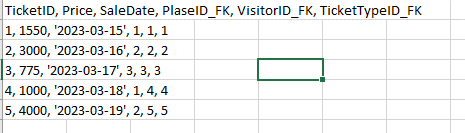
(5, 40.00, '2023-03-19', 2, 5, 5);

\*/

-- Копирование данных с помощью COPY из файла csv

COPY ticket FROM 'T:\Ticket.csv' DELIMITER ',' CSV HEADER ;





-- Вставка данных в таблицу "Товары"

INSERT INTO Product

VALUES

(1, 'Сувенирный нож', 'Нож с изображением герба города', 15.99, 20, 1),

(2, 'Почтовая марка', 'Специальная марка, выпущенная к юбилею музея', 0.50, 50, 2),

(3, 'Книга по истории города', 'Подробный путеводитель по истории города', 12.99, 15, 3),

(4, 'Модель старинного корабля', 'Модель корабля, построенного в XVIII веке', 35.99, 10, 4),

(5, 'Старинная монета', 'Серебряная монета, найденная при раскопках', 50.00, 5, 5);

-- Вставка данных в таблицу "Сотрудник"

INSERT INTO Employee

VALUES

(1, 'Иванов', 'Иван', 'Иванович', '1990-01-15', '+79111111111', 'ivan.ivanov@example.com', 1),

(2, 'Петрова', 'Мария', 'Петровна', '1985-07-20', '+79222222222', 'maria.petrova@example.com', 2),

(3, 'Сидоров', 'Алексей', 'Алексеевич', '1995-03-10', '+79333333333', 'alexey.sidorov@example.com', 3),

(4, 'Кузнецова', 'Екатерина', 'Борисовна', '1988-11-05', '+79444444444', 'ekaterina.kuznetsova@example.com', 4),

(5, 'Васильев', 'Сергей', 'Васильевич', '1992-09-25', '+79555555555', 'sergey.vasilyev@example.com', 5);

-- Вставка данных в таблицу "События"

INSERT INTO MuseumEvent

VALUES

(1, 'Выставка "Древнее искусство"', '2023-12-10', '10:00-18:00', 1, 2, 1),

(2, 'Лекция "История моды Кирова"', '2023-12-15', '14:00-15:30', 2, 2, 2),

(3, 'Концерт классической музыки', '2023-12-20', '19:00-21:00', 3, 4, 3),

(4, 'Мастер-класс по живописи', '2023-12-25', '11:00-13:00', 1, 4, 4),

(5, 'Театральная постановка "Гамлет"', '2023-12-30', '18:00-20:00', 3, 2, 5);

-- Вставка данных в таблицу "Выставка"

INSERT INTO Exhibition

VALUES

(1, 'Выставка современного искусства', 'Выставка работ современных художников.', '2023-10-25', '2023-11-15', 3, 1),

(2, 'Фотовыставка "Мир природы нашего края"', 'Выставка фотографий дикой природы.', '2023-11-01', '2023-11-30', 2, 2),

(3, 'Выставка книг', 'Показ коллекций жителей области.', '2023-12-05', '2023-12-20', 3, 3),

(4, 'Выставка скульптур', 'Выставка работ современных скульпторов.', '2024-01-10', '2024-02-05', 2, 1),

(5, 'Выставка картин', 'Выставка работ классических художников.', '2024-02-15', '2024-03-10', 3, 2);

-- Вставка данных в таблицу "Коллекция"

INSERT INTO Collection

VALUES

(1, 'Коллекция монет', 'Коллекция старинных монет разных стран', 'Нумизматика', '2023-03-15', 1),

(2, 'Коллекция марок', 'Коллекция почтовых марок из разных стран', 'Филателия', '2023-04-20', 2),

(3, 'Коллекция книг', 'Коллекция книг по истории и литературе', 'Литература', '2023-05-10', 3),

(4, 'Коллекция картин', 'Коллекция картин известных художников', 'Искусство', '2023-06-05', 4),

(5, 'Коллекция статуэток', 'Коллекция статуэток из разных материалов', 'Декоративно-прикладное искусство', '2023-07-12', 5);

-- Вставка данных в таблицу "Экспонат"

INSERT INTO Exhibit

VALUES

(1, 'Древняя глиняная посуда', 'Фрагменты глиняной посуды, датируемые 2-м тысячелетием до н.э., найденные на территории области.', '2000 г. до н.э.', 'Село Ивановка', TRUE, 'Описание археологической находки, фото, датировка', 1),

(2, 'Старинный ткацкий станок', 'Ткацкий станок 19 века, использовавшийся для изготовления тканей из шерсти и льна.', '1850', 'Город Петровск', TRUE, 'Описание станка, фото, история создания', 2),

(3, 'Карта области 18 века', 'Подробная карта области, составленная в 18 веке, с указанием населенных пунктов, рек и лесов.', '1780', 'Неизвестно', TRUE, 'Описание карты, фото, история создания', 3),

(4, 'Портрет местного художника', 'Портрет известного художника, уроженца нашей области, написанный в начале 20 века.', '1910', 'Город Хлынов', TRUE, 'Описание картины, фото, биография художника', 4),

(5, 'Сборник народных песен', 'Сборник народных песен, записанных в 19 веке в разных селах области.', '1880', 'Разные села области', FALSE, 'Описание сборника, тексты песен', 5);

-- Заполнение таблицы "Экспозиция"

INSERT INTO Exposition (ExhibitionID\_FK, ExhibitID\_FK, Status) VALUES

(1, 2, TRUE),

(5, 3, TRUE),

(2, 3, TRUE),

(4, 4, FALSE),

(3, 1, TRUE),

(3, 2, TRUE);

-- Заполнение таблицы "Состав коллекции"

INSERT INTO CollectionComposition (ExhibitID\_FK, CollectionID\_FK, CommentInfo) VALUES

(1, 2, 'Основной экспонат'),

(2, 1, 'Дополнительный экспонат'),

(3, 2, 'Ключевой экспонат'),

(4, 3, 'Временно снят с экспозиции'),

(5, 5, 'Предмет временной экспозиции'),

(2, 5, 'Дополняющий экспонат');

==================================================================================================

**Задание 3:**

**3.1:**

- c применение ключевого слова LIMIT:

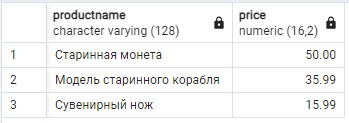
SELECT ProductName, Price

FROM Product

WHERE Price <> 1

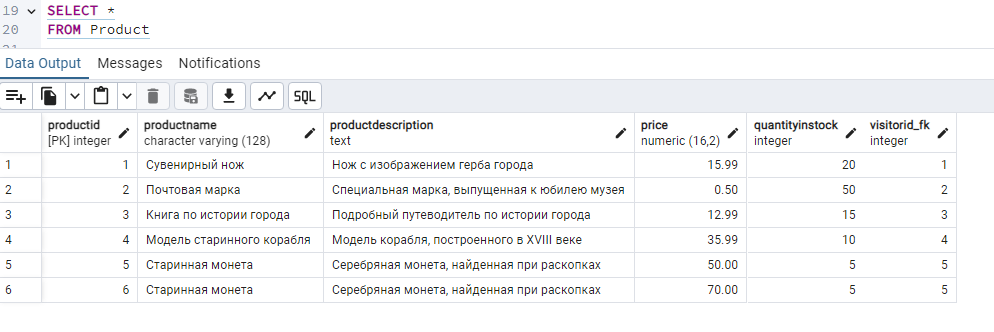
ORDER BY Price DESC

LIMIT 3 -- ограничивает кол-во выводимых значений



- с применением ключевого слова DISTINCT:

-- Ключевое слово DISTINCT в SQL используется для удаления дубликатов из результата запроса.

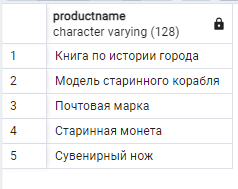


-- Задача: вывести список всех уникальных названий товаров, отсортированных по алфавиту.

SELECT DISTINCT ProductName -- удаляет дубликаты

FROM Product

ORDER BY ProductName;



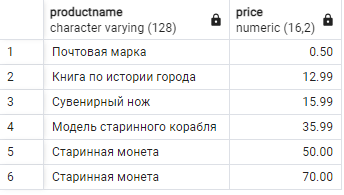
- с применением сортировки:

-- Задача: получить список продуктов, отсортированный по цене по возрастанию, а затем по имени по убыванию.

SELECT ProductName, Price

FROM Product

ORDER BY price ASC, ProductName DESC;



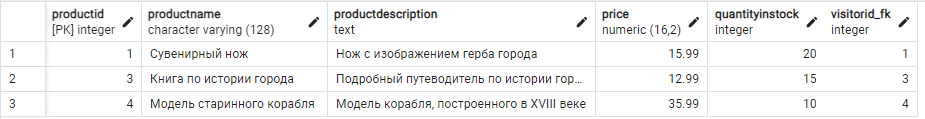
- с условием на принадлежность диапазону значений:

-- Задачи: вывести все продукты с ценой в диапазоне от 10 до 40

SELECT \*

FROM Product

WHERE Price BETWEEN 10 AND 40;

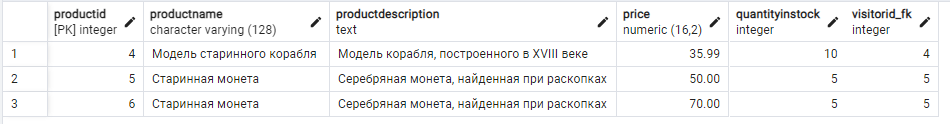


- с условием на соответствие шаблону:

SELECT \*

FROM Product

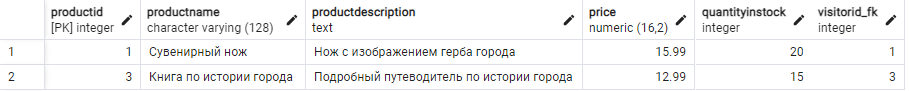
WHERE Price > 20;



SELECT \*

FROM Product

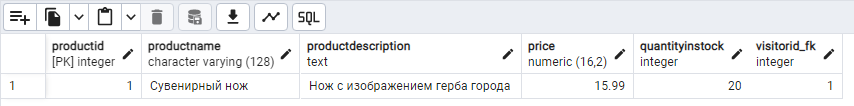
WHERE QuantityInStock IN (15, 20);



SELECT \*

FROM Product

WHERE ProductName LIKE 'Сувенирный %%%';



- со сложным условием:

SELECT \*

FROM Exhibit

WHERE

-- Проверка года создания

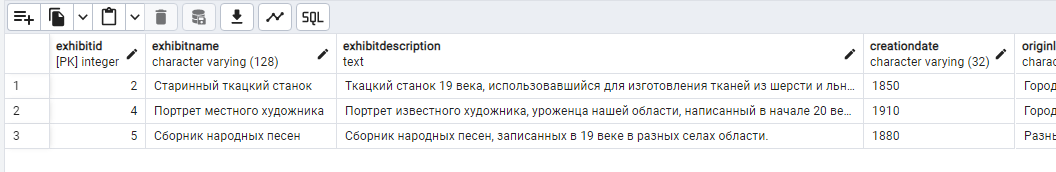
CreationDate BETWEEN '1810' AND '1910'

-- И проверка наличия фотографии

AND Photo IN (TRUE)

-- ИЛИ проверка объекта

OR ExhibitName = 'Сборник народных песен'

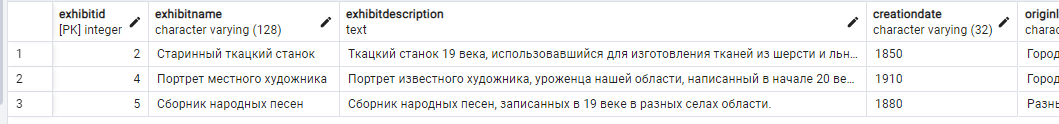


- с условием на принадлежность множеству значений:

SELECT \*

FROM Exhibit

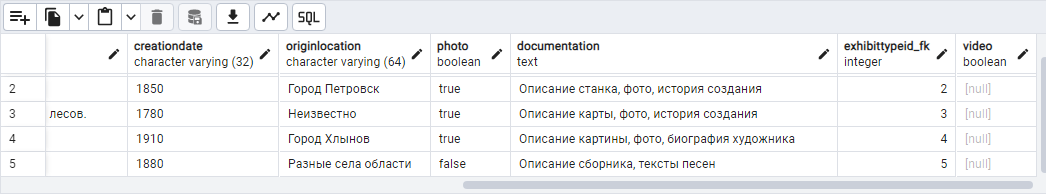
WHERE CreationDate IN ('1880', '1910', '1850');



- с формированием нового поля:

ALTER TABLE Exhibit

ADD COLUMN Video boolean



- с группировкой записей и подсчетом агрегатных показателей (не менее 2-х запросов):

Запрос 1:

-- Группировка по категории товаров и подсчет количества товаров в каждой категории

SELECT

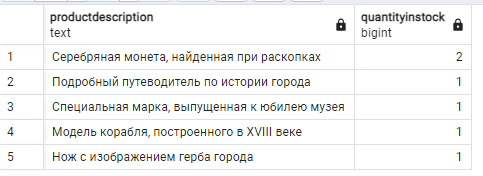
ProductDescription,

COUNT(\*) AS QuantityInStock

FROM Product

GROUP BY ProductDescription

ORDER BY QuantityInStock DESC;



Запрос 2:

-- Группировка по цене и подсчет количества товаров с такой ценой

SELECT

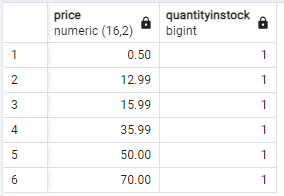
Price,

COUNT(\*) AS QuantityInStock

FROM Product

GROUP BY Price

ORDER BY Price ASC;



- на явное внутреннее и внешнее соединение таблиц (не менее 2-х запросов):

Внутреннее соединение (INNER JOIN). Определение: возвращает только строки, где есть совпадение значений в столбцах соединения обеих таблиц.

Внешнее соединение (OUTER JOIN). Определение: возвращает все строки из одной таблицы, независимо от того, есть ли совпадения в другой.

Внутреннее соединение используется для получения информации только о записях, которые присутствуют в обеих таблицах.

Внешнее соединение используется для получения информации о всех записях одной таблицы, даже если в другой нет соответствующих записей.

*Внутреннее явное соединение таблиц:*

1)

-- Этот запрос вернет таблицу, содержащую информацию о сотрудниках и посетителях, которые имеют одинаковые фамилию, имя и отчество.

SELECT

e.EmployeeID,

e.Surname AS EmployeeSurname,

e.Name\_ AS EmployeeName,

e.Patronymic AS EmployeePatronymic,

v.VisitorID,

v.Surname AS VisitorSurname,

v.Name\_ AS VisitorName,

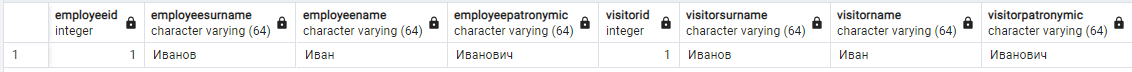
v.Patronymic AS VisitorPatronymic

FROM

Employee e

INNER JOIN

Visitor v ON e.Surname = v.Surname AND e.Name\_ = v.Name\_ AND e.Patronymic = v.Patronymic;



2)

SELECT

c.CollectionName,

c.CollectionDescription,

c.CreationDate,

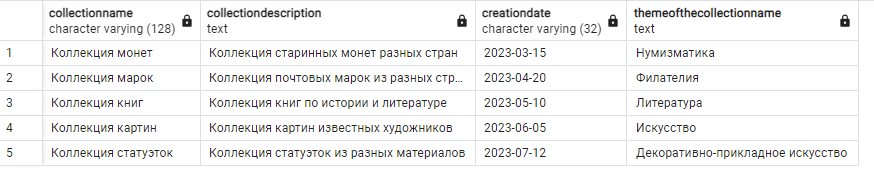
t.ThemeOfTheCollectionName

FROM

Collection AS c

INNER JOIN

ThemeOfTheCollection AS t ON c.ThemeOfTheCollectionID\_FK = t.ThemeOfTheCollectionID;



*Внешнее явное соединение таблиц:*

1)

-- В результате запроса будет отображена таблица с информацией о всех событиях в музее, а также о товарах, которые были проданы во время каждого события. В строках, где товар не продавался, поля, относящиеся к товару, будут содержать NULL.

SELECT

me.MuseumEventName,

me.MuseumEventDate,

me.MuseumEventTime,

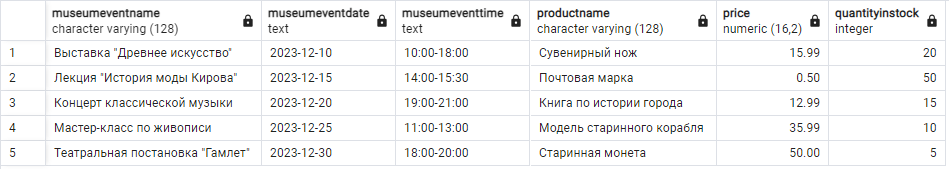
pr.ProductName,

pr.Price,

pr.QuantityInStock

FROM MuseumEvent me

LEFT JOIN Product pr ON me.MuseumEventID = pr.ProductID;



2)

SELECT

v.VisitorID,

v.Surname,

v.Name\_,

v.Patronymic,

v.Age,

v.Address,

v.Phone,

v.Email,

v.VisitDate,

pr.PlaseID,

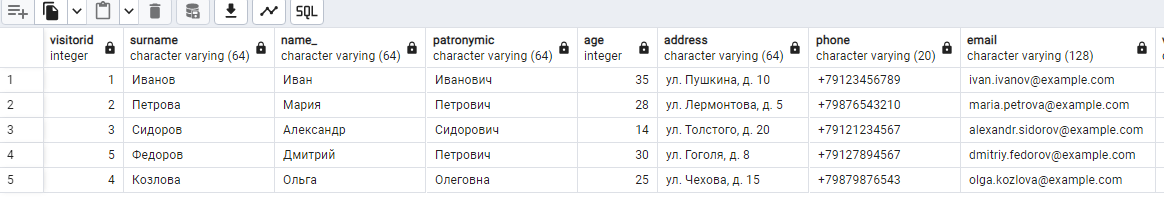
pr.PlaseName,

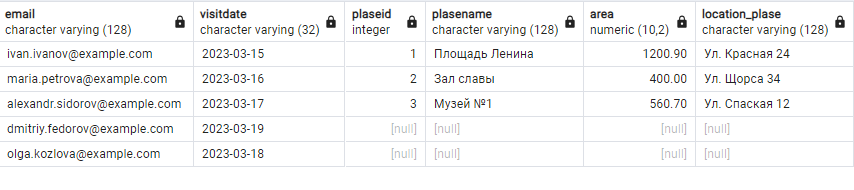
pr.Area,

pr.Location\_plase

FROM Visitor v

LEFT JOIN Plase pr ON v.VisitorID = pr.PlaseID;





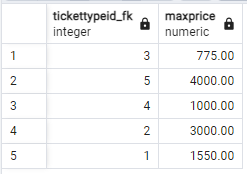
- с использованием вложенных запросов – подзапросов (не менее 2-х запросов):

*Находит самый дорогой билет каждого типа:*

SELECT TicketTypeID\_FK, MAX(Price) AS MaxPrice

FROM Ticket

GROUP BY TicketTypeID\_FK;



*Находит типы билетов, которые были куплены больше всего раз:*

SELECT TicketType.TicketTypeName, COUNT(\*) AS TotalTicketsSold

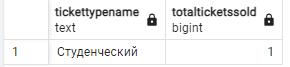
FROM Ticket

JOIN TicketType ON Ticket.TicketTypeID\_FK = TicketType.TicketTypeID

GROUP BY TicketType.TicketTypeName

ORDER BY TotalTicketsSold DESC

LIMIT 1;



**3.2:**

1) Поиск максимальной и минимальной стоимости товара

CREATE OR REPLACE FUNCTION get\_max\_min(OUT max\_price int, OUT min\_price int) AS $$

BEGIN

SELECT MAX(Price), MIN(Price)

INTO max\_price, min\_price

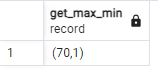
FROM Product;

END;

$$ LANGUAGE plpgsql;

SELECT get\_max\_min();

-- SELECT \* FROM get\_max\_min();



2) Суммирование двух значений

CREATE OR REPLACE FUNCTION get\_sum(x int, y int, out result int) AS $$

BEGIN

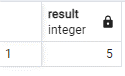
result = x + y

RESULT;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM get\_sum(2,3);



3)

-- Функция для поиска площади по имени места

CREATE OR REPLACE FUNCTION get\_area\_by\_plase\_name(plase\_name varchar(128))

RETURNS decimal(10, 2) AS $$

BEGIN

RETURN (SELECT Area FROM Plase WHERE PlaseName = plase\_name);

END;

$$ LANGUAGE plpgsql;

-- Пример использования функции

SELECT get\_area\_by\_plase\_name('Площадь Ленина');



4) Функция для добавление записи в таблицу, с проверкой на корректность вводимых данных

-- Функция для добавления записи в таблицу Places

CREATE OR REPLACE FUNCTION add\_place(

id\_ int,

plase\_name VARCHAR(128),

area DECIMAL(10, 2),

location\_plase VARCHAR(128)

)

RETURNS VOID AS $$

BEGIN

-- Проверка наличия дубликатов по имени места

IF EXISTS (SELECT 1 FROM Plase WHERE PlaseName = plase\_name) THEN

RAISE EXCEPTION 'Место с таким именем уже существует.';

END IF;

-- Проверка корректности площади

IF area <= 0 THEN

RAISE EXCEPTION 'Площадь должна быть положительной.';

END IF;

-- Вставка новой записи в таблицу

INSERT INTO Plase (PlaseID, PlaseName, Area, Location\_plase)

VALUES (id\_, plase\_name, area, location\_plase);

END;

$$ LANGUAGE plpgsql;

-- Пример использования функции

SELECT add\_place(4,'Музей №2', 341.00, 'Ул Белая 34');



==================================================================================================

**Задание 4:**

Определение: триггер – это хранимая в базе данных процедура, автоматически вызываемая СУБД при возникновении соответствующих условий.

1) Триггер проверки заполнения поля name. Этот триггер привязан к таблице Plase. Он запускается перед каждой операцией INSERT или UPDATE.

CREATE OR REPLACE FUNCTION check\_plase\_name()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.name = '' THEN

RAISE EXCEPTION 'Название места не может быть пустым.';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER check\_place\_name\_trigger

BEFORE INSERT OR UPDATE ON Plase

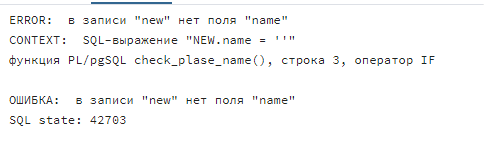
FOR EACH ROW

EXECUTE PROCEDURE check\_plase\_name();

INSERT INTO Plase VALUES (

'4', '', 300.00, 'Ул. Ленина 10'

);



2) Этот триггер проверяет, что количество товара на складе (QuantityInStock) не становится отрицательным при обновлении записи.

CREATE OR REPLACE FUNCTION check\_quantity\_on\_update()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.QuantityInStock < 0 THEN

RAISE EXCEPTION 'Количество товара на складе не может быть отрицательным.';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER check\_quantity\_on\_update

BEFORE UPDATE ON Product

FOR EACH ROW

EXECUTE PROCEDURE check\_quantity\_on\_update();

UPDATE Product

SET QuantityInStock = -1

WHERE ProductName = 'Сувенирный нож';

