МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное   
учреждение высшего образования

**"Южно-Уральский государственный университет**

**(национальный исследовательский университет)"**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

**СВОДНЫЙ ОТЧЕТ**

о выполнении практического задания

по дисциплине

«Хранилища данных»

|  |  |
| --- | --- |
|  | Выполнил:   студент группы КЭ-303  Старостенок Д.В.  Проверил:  ст. преподаватель кафедры СП  Петрова Л.Н. |

Челябинск-2022

ОГЛАВЛЕНИЕ

[1. РАЗРАБОТКА МОДЕЛИ ПРЕДМЕТНОЙ ОБЛАСТИ 3](#_Toc114435818)

[2. РАЗРАБОТКА СХЕМЫ БАЗЫ ДАННЫХ И ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ 3](#_Toc114435819)

[3. ПРЕДСТАВЛЕНИЯ 6](#_Toc114435820)

[4. ЗАПРОСЫ 19](#_Toc114435821)

[5. ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ БЕЗОПАСНОСТИ ДАННЫХ. РАЗРАБОТКА ТРИГГЕРОВ 35](#_Toc114435822)

# 1. Разработка модели предметной области

Разработана формальная модель предметной области для небольшой строительной фирмы, которая занимается ремонтом помещений, в виде ER диаграммы по нотации Чена (Рис. 1), созданная на сайте draw.io.

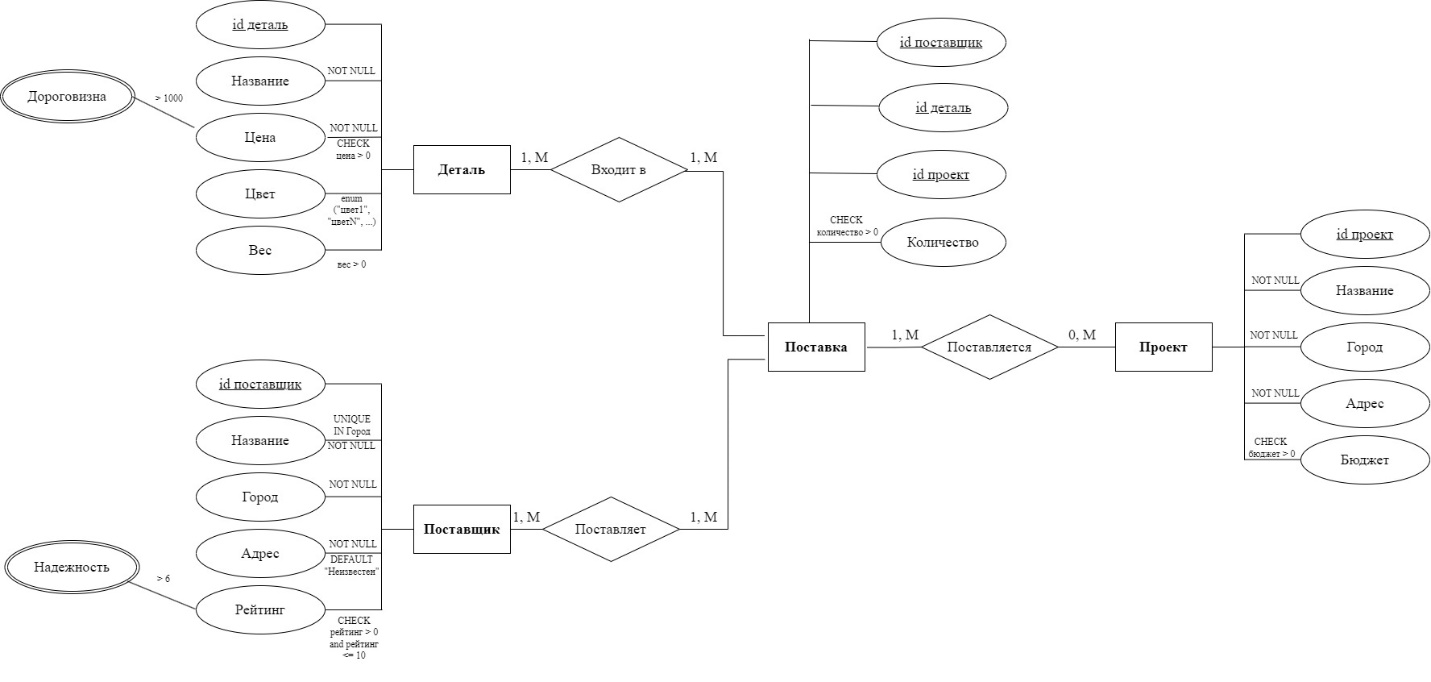


Рис. 1 – ER диаграмма

# 2. Разработка схемы базы данных и ограничений целостности

Были разработаны следующие запросы, для создания реляционной базы данных:

Запрос для создания собственного типа данных набора цветов:

CREATE TYPE COLORS\_CUSTOM

AS ENUM ('белый', 'черный', 'красный', 'синий', 'серый', 'зеленый', 'желтый', 'оранжевый');

Создание таблицы “Детали”:

CREATE TABLE details(

id\_detail INT PRIMARY KEY NOT NULL,

detail\_name VARCHAR(120) NOT NULL,

detail\_price INT NOT NULL CHECK (detail\_price > 0),

detail\_color colors\_custom NOT NULL,

detail\_weight INT NOT NULL CHECK (detail\_weight > 0))

Создание таблицы “Города”:

CREATE TABLE cities(

id\_city INT PRIMARY KEY NOT NULL,

city\_name VARCHAR(120) NOT NULL)

Создание таблицы “Поставщик”:

CREATE TABLE providers(

id\_provider INT PRIMARY KEY NOT NULL,

provider\_name VARCHAR(120) NOT NULL,

id\_city INT NOT NULL,

provider\_address VARCHAR(120) DEFAULT 'Неизвестен' NOT NULL,

provider\_rating INT NOT NULL CHECK (provider\_rating > 0 AND provider\_rating <= 10),

FOREIGN KEY (id\_city) REFERENCES cities(id\_city) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT Same\_provider\_name\_in\_city

UNIQUE (provider\_name, id\_city))

Создание таблицы “Проект”:

CREATE TABLE projects(

id\_project INT PRIMARY KEY NOT NULL,

project\_name VARCHAR(120) NOT NULL,

id\_city INT NOT NULL,

address VARCHAR(120),

budget INT NOT NULL CHECK(budget>0),

FOREIGN KEY (id\_city) REFERENCES cities(id\_city) ON DELETE CASCADE ON UPDATE CASCADE)

Создание таблицы “Поставка”:

CREATE TABLE supplies(

id\_provider INT NOT NULL,

id\_detail INT NOT NULL,

id\_project INT NOT NULL,

supplies\_quantity INT NOT NULL CHECK(supplies\_quantity>0),

PRIMARY KEY(id\_provider, id\_detail, id\_project),

FOREIGN KEY (id\_provider) REFERENCES providers(id\_provider) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (id\_detail) REFERENCES details(id\_detail) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (id\_project) REFERENCES projects(id\_project) ON DELETE CASCADE ON UPDATE CASCADE)

# 3. Представления

Задание 4. Разработка тестовой базы данных.

Разработайте тестовую базу данных таким образом, чтобы в каждой таблице содержалось не менее 20 записей с помощью команды INSERT и выгрузкой из csv файлов.

Вставка данных в таблицу “Города” с помощью команды INSERT:

INSERT INTO cities(city\_name)

VALUES ('Челябинск');

Вставка данных в таблицу “Города” с помощью команды COPY:

COPY cities(city\_name)

FROM 'C:\Users\Dmitriy\Desktop\PostgreQuery\csv files\cities.csv'

DELIMITER ','

CSV HEADER encoding 'UTF8';

Вставка данных в таблицу “Детали” с помощью команды INSERT:

INSERT INTO details(

detail\_name, detail\_price, detail\_color, detail\_weight)

VALUES ('Шуруп', 16, 'черный', 3);

Вставка данных в таблицу “Детали” с помощью команды COPY:

COPY details(detail\_name, detail\_price, detail\_color, detail\_weight)

FROM 'C:\Users\Dmitriy\Desktop\PostgreQuery\csv files\details.csv'

DELIMITER ';'

CSV HEADER encoding 'UTF8';

Вставка данных в таблицу “Проекты” с помощью команды INSERT:

INSERT INTO projects(

project\_name, id\_city, address, budget)

VALUES ('Стена', 3, 'Иванова 22', 12000);

Вставка данных в таблицу “Проекты” с помощью команды COPY:

COPY projects(project\_name, id\_city, address, budget)

FROM 'C:\Users\Dmitriy\Desktop\PostgreQuery\csv files\projects.csv'

DELIMITER ';'

CSV HEADER encoding 'UTF8';

Вставка данных в таблицу “Поставщики” с помощью команды INSERT:

INSERT INTO providers(

provider\_name, id\_city, provider\_address, provider\_rating)

VALUES ('Ремонт в дом', 3, 'Ленина 25', 7);

Вставка данных в таблицу “Поставщики” с помощью команды COPY:

COPY providers(provider\_name, id\_city, provider\_address, provider\_rating)

FROM 'C:\Users\Dmitriy\Desktop\PostgreQuery\csv files\providers.csv'

DELIMITER ';'

CSV HEADER encoding 'UTF8';

Вставка данных в таблицу “Поставки” с помощью команды INSERT:

INSERT INTO public.supplies(

id\_provider, id\_detail, id\_project, supplies\_quantity)

VALUES (1, 1, 1, 22);

Вставка данных в таблицу “Поставки” с помощью команды COPY:

COPY supplies(id\_provider, id\_detail, id\_project, supplies\_quantity)

FROM 'C:\Users\Dmitriy\Desktop\PostgreQuery\csv files\supplies.csv'

DELIMITER ';'

CSV HEADER encoding 'UTF8';

Задание 5. Разработка представлений с помощью VIEW для ролей “Экономист” и “Директор”.

Для роли Экономиста.

1. Экономист имеет право осуществлять выборку, добавление, обновление и удаление данных о поставщиках и поставках, а также выборку данных о деталях и проектах.

Выборка данных о поставщиках:

SELECT id\_provider, provider\_name, cities.city\_name, provider\_address, provider\_rating

FROM providers

INNER JOIN cities ON cities.id\_city = providers.id\_city

Добавление данных о поставщиках:

INSERT INTO providers(

id\_provider, provider\_name, id\_city, provider\_address, provider\_rating)

VALUES (?, ?, ?, ?, ?);

Обновление данных о поставщиках:

UPDATE public.providers

SET id\_provider=?, provider\_name=?, id\_city=?, provider\_address=?, provider\_rating=?

WHERE <condition>;

Удаление данных о поставщиках:

DELETE FROM public.providers

WHERE <condition>;

Выборка данных о поставках:

SELECT providers.provider\_name, details.detail\_name, projects.project\_name, supplies.supplies\_quantity

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

INNER JOIN projects ON projects.id\_project = supplies.id\_project

Добавление данных о поставках:

INSERT INTO public.supplies(

id\_provider, id\_detail, id\_project, supplies\_quantity)

VALUES (?, ?, ?, ?);

Обновление данных о поставках:

UPDATE public.supplies

SET id\_provider=?, id\_detail=?, id\_project=?, supplies\_quantity=?

WHERE <condition>;

Удаление данных о поставках:

DELETE FROM public.supplies

WHERE <condition>;

Выборка данных о деталях:

SELECT \* FROM details

Выборка данных о проектах:

SELECT id\_project, project\_name, cities.city\_name, address, budget

FROM projects

INNER JOIN cities ON cities.id\_city = projects.id\_city

1. При выборке данных Поставщик для Экономиста должен быть представлен следующими атрибутами: название, город, адрес, рейтинг, признак «ненадежный» или «надежный» (символьная строка). Данные должны быть отсортированы следующим образом: по городу, названию, убыванию рейтинга.

CREATE VIEW data\_from\_provider\_economist AS

SELECT

provider\_name AS Название,

cities.city\_name AS Город,

provider\_address AS Адрес,

provider\_rating AS Рейтинг,

CASE

WHEN provider\_rating>=6

THEN 'Надежный'

WHEN provider\_rating<6

THEN 'Ненадежный'

END Надежность

FROM providers

INNER JOIN cities ON cities.id\_city = providers.id\_city

ORDER BY cities.city\_name ASC, providers.provider\_name ASC, providers.provider\_rating DESC

1. Деталь для Экономиста должна быть представлена следующими атрибутами: название, цена в тыс.руб., цвет, вес в килограммах, признак «дешевая» или «дорогая» (символьная строка). Данные должны быть отсортированы следующим образом: по убыванию цены, названию, цвету, возрастанию веса.

CREATE VIEW data\_from\_details\_economist AS

SELECT

detail\_name AS Название,

detail\_price AS Цена,

detail\_color AS Цвет,

detail\_weight/1000::float4 AS "Вес, кг",

CASE

WHEN detail\_price>=1000

THEN 'Дорогая'

WHEN detail\_price<1000

THEN 'Дешевая'

END Дороговизна

FROM details

ORDER BY detail\_price DESC, detail\_name ASC, detail\_color ASC, detail\_weight ASC

1. При выборке данных Проект для Экономиста должен быть представлен следующими атрибутами: название, город, адрес, бюджет. Данные должны быть отсортированы следующим образом: по городу, названию, убыванию бюджета.

CREATE VIEW data\_from\_projects\_economist AS

SELECT

project\_name AS Название,

cities.city\_name AS Город,

address AS Адрес,

budget AS Бюджет

FROM projects

INNER JOIN cities ON cities.id\_city = projects.id\_project

ORDER BY cities.city\_name ASC, projects.project\_name ASC, projects.budget DESC

1. При выборке данных Поставка для Экономиста должна быть представлена следующими атрибутами: название детали, город детали, цвет детали, признак детали «дешевая» или «дорогая» (символьная строка), название поставщика, город поставщика, признак поставщика «ненадежный» или «надежный» (символьная строка), количество деталей, вес поставки в килограммах, стоимость поставки в тыс.руб. Данные должны быть отсортированы следующим образом: названию детали, названию поставщика, по убыванию стоимости, по убыванию веса.

CREATE VIEW data\_from\_supplies\_economist AS

SELECT

details.detail\_name AS "Название детали",

details.detail\_color AS "Цвет детали",

CASE

WHEN details.detail\_price>=1000

THEN 'Дорогая'

WHEN details.detail\_price<1000

THEN 'Дешевая'

END Дороговизна\_детали,

providers.provider\_name AS "Название поставщика",

cities.city\_name AS "Город поставщика",

CASE

WHEN providers.provider\_rating>=6

THEN 'Надежный'

WHEN providers.provider\_rating<6

THEN 'Ненадежный'

END Надежность\_провайдера,

supplies.supplies\_quantity AS "Количество деталей",

(details.detail\_weight \* supplies.supplies\_quantity)/1000::float4 AS "Вес поставки, кг",

(details.detail\_price \* supplies.supplies\_quantity) AS "Стоимость поставки"

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

INNER JOIN projects ON projects.id\_project = supplies.id\_project

INNER JOIN cities ON cities.id\_city = providers.id\_city

ORDER BY details.detail\_name,

providers.provider\_name,

(details.detail\_price \* supplies.supplies\_quantity) DESC,

details.detail\_weight DESC

Для роли Директора

1. Директор имеет право осуществлять выборку, добавление, обновление и удаление данных о поставщиках, деталях и проектах, а также выборку и удаление данных о поставках.

Выборка данных о поставщиках:

SELECT id\_provider, provider\_name, cities.city\_name, provider\_address, provider\_rating

FROM providers

INNER JOIN cities ON cities.id\_city = providers.id\_city

Добавление данных о поставщиках:

INSERT INTO providers(

id\_provider, provider\_name, id\_city, provider\_address, provider\_rating)

VALUES (?, ?, ?, ?, ?);

Обновление данных о поставщиках:

UPDATE public.providers

SET id\_provider=?, provider\_name=?, id\_city=?, provider\_address=?, provider\_rating=?

WHERE <condition>;

Удаление данных о поставщиках:

DELETE FROM public.providers

WHERE <condition>;

Выборка данных о деталях:

SELECT \* FROM details

Добавление данных о деталях:

INSERT INTO public.details(

id\_detail, detail\_name, detail\_price, detail\_color, detail\_weight)

VALUES (?, ?, ?, ?, ?);

Обновление данных о деталях:

UPDATE public.details

SET id\_detail=?, detail\_name=?, detail\_price=?, detail\_color=?, detail\_weight=?

WHERE <condition>;

Удаление данных о деталях:

DELETE FROM public.details

WHERE <condition>;

Выборка данных о проектах:

SELECT id\_project, project\_name, cities.city\_name, address, budget

FROM projects

INNER JOIN cities ON cities.id\_city = projects.id\_city

Добавление данных о проектах:

INSERT INTO public.projects(

id\_project, project\_name, id\_city, address, budget)

VALUES (?, ?, ?, ?, ?);

Обновление данных о проектах:

UPDATE public.projects

SET id\_project=?, project\_name=?, id\_city=?, address=?, budget=?

WHERE <condition>;

Удаление данных о проектах:

DELETE FROM public.projects

WHERE <condition>;

Выборка данных о поставках:

SELECT providers.provider\_name, details.detail\_name, projects.project\_name, supplies.supplies\_quantity

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

INNER JOIN projects ON projects.id\_project = supplies.id\_project

Удаление данных о поставках:

DELETE FROM public.supplies

WHERE <condition>;

1. При выборке данных Поставщик для Директора должен быть представлен следующими атрибутами: название, город, признак «ненадежный» или «надежный» (символьная строка). Данные должны быть отсортированы следующим образом: по признаку, городу, названию.

CREATE VIEW data\_from\_provider\_director AS

SELECT

provider\_name AS Название,

cities.city\_name AS Город,

CASE

WHEN providers.provider\_rating>=6

THEN 'Надежный'

WHEN providers.provider\_rating<6

THEN 'Ненадежный'

END Надежность\_провайдера

FROM providers

INNER JOIN cities ON cities.id\_city = providers.id\_city

ORDER BY providers.provider\_rating, cities.city\_name, providers.provider\_name

1. При выборке данных Деталь для Директора должна быть представлена следующими атрибутами: название, цена в тыс.руб., цвет, признак «дешевая» или «дорогая» (символьная строка). Данные должны быть отсортированы следующим образом: по убыванию цены, названию.

CREATE VIEW data\_from\_details\_director AS

SELECT

detail\_name AS Название,

detail\_price AS Цена,

detail\_color AS Цвет,

CASE

WHEN details.detail\_price>=1000

THEN 'Дорогая'

WHEN details.detail\_price<1000

THEN 'Дешевая'

END Дороговизна\_детали

FROM details

ORDER BY detail\_price DESC, detail\_name

1. При выборке данных Проект для Директора должен быть представлен следующими атрибутами: название, город, бюджет. Данные должны быть отсортированы следующим образом: по убыванию бюджета, городу, названию.

CREATE VIEW data\_from\_projects\_director AS

SELECT

project\_name AS Название,

cities.city\_name AS Город,

budget AS Бюджет

FROM projects

INNER JOIN cities ON cities.id\_city = projects.id\_city

ORDER BY budget DESC, cities.city\_name, project\_name

1. При выборке данных Поставка для Директора должна быть представлена следующими атрибутами: название детали, город детали, цвет детали, признак детали «дешевая» или «дорогая» (символьная строка), название поставщика, город поставщика, признак поставщика «ненадежный» или «надежный» (символьная строка), количество деталей, вес поставки в килограммах, стоимость поставки в тыс.руб. Данные должны быть отсортированы следующим образом: по признаку детали, городу детали, признаку поставщика, городу поставщика, названию детали, названию поставщика, по убыванию стоимости, по убыванию веса.

CREATE VIEW data\_from\_supplies\_director AS

SELECT

details.detail\_name AS "Название детали",

details.detail\_color AS "Цвет детали",

CASE

WHEN details.detail\_price>=1000

THEN 'Дорогая'

WHEN details.detail\_price<1000

THEN 'Дешевая'

END Дороговизна\_детали,

providers.provider\_name AS "Название\_поставщика",

cities.city\_name AS "Город поставщика",

CASE

WHEN providers.provider\_rating>=6

THEN 'Надежный'

WHEN providers.provider\_rating<6

THEN 'Ненадежный'

END Надежность\_поставщика,

supplies.supplies\_quantity AS "Количество деталей",

(supplies.supplies\_quantity \* details.detail\_weight)/1000::float4 AS "Вес поставки, кг",

(supplies.supplies\_quantity \* details.detail\_price) AS "Цена поставки"

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

INNER JOIN projects ON projects.id\_project = supplies.id\_project

INNER JOIN cities ON cities.id\_city = providers.id\_city

ORDER BY

providers.provider\_rating,

cities.city\_name,

details.detail\_name,

(supplies.supplies\_quantity \* details.detail\_price) DESC,

(supplies.supplies\_quantity \* details.detail\_weight) DESC

# 4. Запросы

Задание 6. Разработка запросов. Разработать запросы к базе данных. Для написания запросов используется команда SELECT.

Экономист должен подготавливать следующие наборы отчетов:

1. Список всех, надежных, ненадежных поставщиков.

Список всех поставщиков:

SELECT id\_provider, provider\_name, id\_city, provider\_address, provider\_rating

FROM providers;

Надежные поставщики:

SELECT

provider\_name AS "Имя поставщика",

CASE

WHEN providers.provider\_rating>=6

THEN 'Надежный'

END Надежность\_поставщика

FROM providers

WHERE providers.provider\_rating>=6

Ненадежные поставщики:

SELECT

provider\_name AS "Имя поставщика",

CASE

WHEN providers.provider\_rating<6

THEN 'Ненадежный'

END Надежность\_поставщика

FROM providers

WHERE providers.provider\_rating<6

1. Список всех, дорогих, дешевых деталей.

Все детали:

SELECT id\_detail, detail\_name, detail\_price, detail\_color, detail\_weight

FROM details;

Дорогие детали:

SELECT

detail\_name AS "Название детали",

CASE

WHEN details.detail\_price>=1000

THEN 'Дорогая'

END Дороговизна\_детали

FROM details

WHERE details.detail\_price>=1000

Дешевые детали:

SELECT

detail\_name AS "Название детали",

CASE

WHEN details.detail\_price<1000

THEN 'Дешевая'

END Дороговизна\_детали

FROM details

WHERE details.detail\_price<1000

1. Список всех проектов, проектов из заданного города, проектов с бюджетом в заданном диапазоне.

Все проекты:

SELECT project\_name AS "Название проекта", cities.city\_name AS "Город", address AS "Адрес", budget AS "Бюджет"

FROM projects

INNER JOIN cities ON cities.id\_city = projects.id\_city

Проекты из заданного города:

SELECT project\_name AS "Название проекта", cities.city\_name AS "Город", address AS "Адрес", budget AS "Бюджет"

FROM projects

INNER JOIN cities ON cities.id\_city = projects.id\_city

WHERE cities.city\_name = 'Москва'

Проекты с бюджетом из диапазона:

SELECT project\_name AS "Название проекта", cities.city\_name AS "Город", address AS "Адрес", budget AS "Бюджет"

FROM projects

INNER JOIN cities ON cities.id\_city = projects.id\_city

WHERE budget BETWEEN 10 AND 10000

1. Список всех поставок, поставок дорогих и дешевых деталей, поставок от надежных и ненадежных поставщиков, поставок со стоимостью из заданного диапазона.

Все поставки:

SELECT providers.provider\_name AS "Имя поставщика", details.detail\_name AS "Название детали", projects.project\_name AS "Имя проекта", supplies.supplies\_quantity AS "Размер поставки"

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

INNER JOIN projects ON projects.id\_project = supplies.id\_project

Поставки дорогих деталей:

SELECT

providers.provider\_name AS "Имя поставщика",

details.detail\_name AS "Название детали",

projects.project\_name AS "Имя проекта",

supplies.supplies\_quantity AS "Размер поставки",

CASE

WHEN details.detail\_price>=1000

THEN 'Дорогая'

END Дороговизна\_детали

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

INNER JOIN projects ON projects.id\_project = supplies.id\_project

WHERE details.detail\_price>=1000

Поставки дешевых деталей:

SELECT

providers.provider\_name AS "Имя поставщика",

details.detail\_name AS "Название детали",

projects.project\_name AS "Имя проекта",

supplies.supplies\_quantity AS "Размер поставки",

CASE

WHEN details.detail\_price<1000

THEN 'Дешевая'

END Дороговизна\_детали

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

INNER JOIN projects ON projects.id\_project = supplies.id\_project

WHERE details.detail\_price<1000

Поставки надежных поставщиков:

SELECT

providers.provider\_name AS "Имя поставщика",

details.detail\_name AS "Название детали",

projects.project\_name AS "Имя проекта",

supplies.supplies\_quantity AS "Размер поставки",

CASE

WHEN provider\_rating>=6

THEN 'Надежный'

END Надежность

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

INNER JOIN projects ON projects.id\_project = supplies.id\_project

WHERE provider\_rating>=6

Поставки ненадежных поставщиков:

SELECT

providers.provider\_name AS "Имя поставщика",

details.detail\_name AS "Название детали",

projects.project\_name AS "Имя проекта",

supplies.supplies\_quantity AS "Размер поставки",

CASE

WHEN provider\_rating<6

THEN 'Ненадежный'

END Надежность

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

INNER JOIN projects ON projects.id\_project = supplies.id\_project

WHERE provider\_rating<6

Поставки из заданного диапазона:

SELECT

providers.provider\_name AS "Имя поставщика",

details.detail\_name AS "Название детали",

projects.project\_name AS "Имя проекта",

supplies.supplies\_quantity AS "Размер поставки",

(supplies.supplies\_quantity \* details.detail\_price) AS "Стоимость поставки"

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

INNER JOIN projects ON projects.id\_project = supplies.id\_project

WHERE (supplies.supplies\_quantity \* details.detail\_price) BETWEEN 10 AND 10000

Директор должен подготавливать следующие наборы отчетов:

1. Список всех поставщиков, список поставщиков с рейтингом не ниже/не выше среднего, список поставщиков с максимальным/минимальным рейтингом – с указанием у каждого поставщика общей суммы сделанной им поставок (непременное условие для всех списков).

Список всех поставщиков:

SELECT

provider\_name AS "Имя поставщика",

cities.city\_name AS "Город",

provider\_address AS "Адрес",

provider\_rating AS "Рейтинг",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM providers

INNER JOIN cities ON cities.id\_city = providers.id\_city

INNER JOIN supplies ON supplies.id\_provider = providers.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY provider\_name, cities.city\_name, provider\_address, provider\_rating

Список поставщиков с рейтингом выше среднего:

SELECT

provider\_name AS "Имя поставщика",

CASE

WHEN providers.provider\_rating>=6

THEN 'Надежный'

WHEN providers.provider\_rating<6

THEN 'Ненадежный'

END Надежность\_поставщика,

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM providers

INNER JOIN supplies ON supplies.id\_provider = providers.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY provider\_name, providers.provider\_rating

HAVING (providers.provider\_rating > (SELECT AVG(providers.provider\_rating) FROM providers))

Список поставщиков с рейтингом ниже среднего:

SELECT

provider\_name AS "Имя поставщика",

CASE

WHEN providers.provider\_rating>=6

THEN 'Надежный'

WHEN providers.provider\_rating<6

THEN 'Ненадежный'

END Надежность\_поставщика,

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM providers

INNER JOIN supplies ON supplies.id\_provider = providers.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY provider\_name, providers.provider\_rating

HAVING (providers.provider\_rating < (SELECT AVG(providers.provider\_rating) FROM providers))

Список поставщиков с максимальным рейтингом:

SELECT

provider\_name AS "Имя поставщика",

provider\_rating AS "Рейтинг",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM providers

INNER JOIN cities ON cities.id\_city = providers.id\_city

INNER JOIN supplies ON supplies.id\_provider = providers.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY provider\_name, cities.city\_name, provider\_address, provider\_rating

HAVING provider\_rating = (SELECT MAX(provider\_rating) FROM providers)

Список поставщиков с минимальным рейтингом:

SELECT

provider\_name AS "Имя поставщика",

provider\_rating AS "Рейтинг",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM providers

INNER JOIN cities ON cities.id\_city = providers.id\_city

INNER JOIN supplies ON supplies.id\_provider = providers.id\_provider

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY provider\_name, cities.city\_name, provider\_address, provider\_rating

HAVING provider\_rating = (SELECT MIN(provider\_rating) FROM providers)

1. Список всех деталей, список деталей с ценой не ниже/не выше средней, список деталей максимальной/минимальной ценой – с указанием у каждой детали общей суммы ее поставок (непременное условие для всех списков).

Список всех деталей:

SELECT

detail\_name AS "Имя детали",

detail\_price AS "Цена",

detail\_color AS "Цвет",

detail\_weight AS "Вес",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM details

INNER JOIN supplies ON supplies.id\_detail = details.id\_detail

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY detail\_name, detail\_price, detail\_color, detail\_weight

Список деталей с ценой выше средней:

SELECT

detail\_name AS "Имя детали",

detail\_price AS "Цена",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM details

INNER JOIN supplies ON supplies.id\_detail = details.id\_detail

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY detail\_name, detail\_price

HAVING (details.detail\_price > (SELECT AVG(details.detail\_price) FROM details))

Список деталей с ценой ниже средней:

SELECT

detail\_name AS "Имя детали",

detail\_price AS "Цена",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM details

INNER JOIN supplies ON supplies.id\_detail = details.id\_detail

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY detail\_name, detail\_price

HAVING (details.detail\_price < (SELECT AVG(details.detail\_price) FROM details))

Список деталей с максимальной ценой:

SELECT

detail\_name AS "Имя детали",

detail\_price AS "Цена",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM details

INNER JOIN supplies ON supplies.id\_detail = details.id\_detail

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY detail\_name, detail\_price

HAVING (details.detail\_price = (SELECT MAX(details.detail\_price) FROM details))

Список деталей с минимальной ценой:

SELECT

detail\_name AS "Имя детали",

detail\_price AS "Цена",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM details

INNER JOIN supplies ON supplies.id\_detail = details.id\_detail

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY detail\_name, detail\_price

HAVING (details.detail\_price = (SELECT MIN(details.detail\_price) FROM details))

1. Список всех проектов, список проектов с бюджетом не ниже/не выше среднего, список проектов с максимальным/минимальным бюджетом – с указанием общей суммы поставок по данному проекту (непременное условие для всех списков).

Список всех проектов:

SELECT

project\_name AS "Название проекта",

cities.city\_name AS "Город",

address AS "Адрес",

budget AS "Бюджет",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM projects

INNER JOIN cities ON cities.id\_city = projects.id\_city

INNER JOIN supplies ON supplies.id\_project = projects.id\_project

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY project\_name, cities.city\_name, address, budget

Список проектов с минимальным бюджетом:

SELECT

project\_name AS "Название проекта",

budget AS "Бюджет",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM projects

INNER JOIN supplies ON supplies.id\_project = projects.id\_project

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY project\_name, budget

HAVING (projects.budget = (SELECT MIN(projects.budget) FROM projects))

Список проектов с максимальным бюджетом:

SELECT

project\_name AS "Название проекта",

budget AS "Бюджет",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM projects

INNER JOIN supplies ON supplies.id\_project = projects.

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY project\_name, budget

HAVING (projects.budget = (SELECT MAX(projects.budget) FROM projects))

Список проектов с ценой выше средней:

SELECT

project\_name AS "Название проекта",

budget AS "Бюджет",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM projects

INNER JOIN supplies ON supplies.id\_project = projects.id\_project

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY project\_name, budget

HAVING (projects.budget > (SELECT AVG(projects.budget) FROM projects))

Список проектов с ценой ниже средней:

SELECT

project\_name AS "Название проекта",

budget AS "Бюджет",

SUM(supplies.supplies\_quantity \* details.detail\_price) AS "Общая сумма поставок"

FROM projects

INNER JOIN supplies ON supplies.id\_project = projects.id\_project

INNER JOIN details ON details.id\_detail = supplies.id\_detail

GROUP BY project\_name, budget

HAVING (projects.budget < (SELECT AVG(projects.budget) FROM projects))

1. Сумма всех поставок, сумма поставок дорогих деталей, сумма поставок дешевых деталей, сумма поставок от надежных поставщиков, сумма поставок от ненадежных поставщиков, сумма поставок от заданного поставщика.

Сумма всех поставок:

SELECT

SUM(supplies.supplies\_quantity) AS "Сумма всех поставок"

FROM supplies

INNER JOIN details ON details.id\_detail = supplies.id\_detail

Сумма всех поставок дорогих деталей:

SELECT

SUM(supplies.supplies\_quantity) AS "Сумма всех поставок"

FROM supplies

INNER JOIN details ON details.id\_detail = supplies.id\_detail

WHERE details.detail\_price >= 1000

Сумма всех поставок дешевых деталей:

SELECT

SUM(supplies.supplies\_quantity) AS "Сумма всех поставок"

FROM supplies

INNER JOIN details ON details.id\_detail = supplies.id\_detail

WHERE details.detail\_price < 1000

Сумма всех поставок от надежных поставщиков:

SELECT

SUM(supplies.supplies\_quantity) AS "Сумма всех поставок"

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

WHERE providers.provider\_rating >= 6

Сумма всех поставок от ненадежных поставщиков:

SELECT

SUM(supplies.supplies\_quantity) AS "Сумма всех поставок"

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

WHERE providers.provider\_rating < 6

Сумма всех поставок от заданного поставщика:

SELECT

SUM(supplies.supplies\_quantity) AS "Сумма всех поставок"

FROM supplies

INNER JOIN providers ON providers.id\_provider = supplies.id\_provider

WHERE providers.provider\_name = 'Юнипро'

# 5. Определение ограничений безопасности данных. Разработка триггеров