Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра программного обеспечения информационных технологий
Дисциплина базы данных

ОТЧЕТ

по лабораторной работе №5

Тема работы: «Улучшение модели базы данных»

Выполнил
студент: гр. 251003 Панкратьев Е. С.

Проверил: Фадеева Е. Е.

Минск 2024

Вариант 13 (Фирма по обслуживанию и ремонту компьютеров)

1. **Определите, какие каскадные операции необходимы в этой базе данных. Настройте соответствующие каскадные операции.**

Удаление данных:

- **Таблица client**:
  - При удалении клиента записи в таблицах:
    - request — устанавливается NULL в поле client.
    - invoice — устанавливается NULL в поле client.
    - payment — удаляются записи, связанные с данным клиентом.
- **Таблица request**:
  - При удалении заявки записи в таблицах:
    - request__request_status_map — удаляются связанные записи.
    - provided_service — удаляются связанные записи.
    - invoice — данные заявки остаются, так как действие ON DELETE указано как NO ACTION.
    - employee_request_map — удаляются связанные записи.
- **Таблица invoice**:
  - При удалении счета записи в таблицах:
    - provided_service — устанавливается NULL в поле invoice.
    - service_part — удаляются связанные записи.
    - payment — удаляются записи, связанные с данным счетом.
- **Таблица provided_service**:
  - При удалении предоставленной услуги записи в таблице:
    - service_part — удаляются связанные записи.
- **Таблица part**:
  - При удалении запчасти записи в таблицах:
    - service_part — устанавливается NULL в поле part.
    - contract_part — удаляются связанные записи.
    - supply — удаляются связанные записи.
- **Таблица supplier**:
  - При удалении поставщика записи в таблице:
    - contract — удаляются связанные записи.
    - supply — удаляются связанные записи.
- **Таблица contract**:
  - При удалении контракта записи в таблицах:
    - terms — удаляются связанные записи.
    - contract_part — удаляются связанные записи.
    - supply — удаляются связанные записи.

- **Таблица contract_part**:
  - При удалении элемента контракта записи в таблице:
    - supply — устанавливается NULL в поле contract_part.
- **Таблица employee**:
  - При удалении сотрудника записи в таблицах:
    - employee_request_map — удаляются связанные записи.
    - salary — удаляются записи, связанные с данным сотрудником.

Обновление данных:

- **Таблица client**:
  - При изменении идентификатора клиента изменения автоматически распространяются в таблицы:
    - request — обновляется поле client.
    - invoice — обновляется поле client.
    - payment — обновляется поле client.
- **Таблица request**:
  - При изменении идентификатора заявки изменения распространяются в таблицы:
    - request__request_status_map — обновляется поле request.
    - provided_service — обновляется поле request.
    - invoice — обновляется поле request.
    - employee_request_map — обновляется поле request.
- **Таблица invoice**:
  - При изменении идентификатора счета изменения распространяются в таблицы:
    - provided_service — обновляется поле invoice.
    - service_part — обновляется поле invoice.
    - payment — обновляется поле invoice.
- **Таблица provided_service**:
  - При изменении идентификатора предоставленной услуги изменения распространяются в таблицу:
    - service_part — обновляется поле provided_service.
- **Таблица part**:
  - При изменении идентификатора запчасти изменения распространяются в таблицы:
    - service_part — обновляется поле part.
    - contract_part — обновляется поле part.
    - supply — обновляется поле part.
- **Таблица supplier**:
  - При изменении идентификатора поставщика изменения распространяются в таблицы:
    - contract — обновляется поле supplier.
    - supply — обновляется поле supplier.

- **Таблица contract**:
  - При изменении идентификатора контракта изменения распространяются в таблицы:
    - terms — обновляется поле contract.
    - contract_part — обновляется поле contract.
    - supply — обновляется поле contract.
- **Таблица contract_part**:
  - При изменении идентификатора элемента контракта изменения распространяются в таблицу:
    - supply — обновляется поле contract_part.

2. Создайте список представлений, которые нужно добавить в базу данных, а также подсписок задач, выполняемых каждым представлением. Создайте соответствующие представления.

client_requests - Информация о клиентах и их запросах.

- Задачи:
  - Получение списка клиентов с их активными запросами.

```
CREATE VIEW client_requests AS

SELECT

    c.id_client,

    CONCAT(c.first_name, ' ', c.last_name) AS full_name,

    c.phone_number,

    r.id_request,

    r.description,

    r.request_date,

    r.completion_date

FROM

    client c

    LEFT JOIN request r ON c.id_client = r.client;
```

active_requests_status - Отображает статус активных запросов.

- Задачи:
    - Отслеживание текущего статуса запросов.

```
CREATE VIEW active_requests_status AS

SELECT

    r.id_request,

    r.description,

    rs.name AS status_name

FROM

    request r

    JOIN request__request_status_map rsm ON r.id_request = rsm.request

    JOIN request_status rs ON rsm.request_status = rs.id_request_status

WHERE

    r.completion_date IS NULL;
```

financial_report - Отчет по счетам и платежам.

- Задачи:
    - Финансовый анализ поступлений.

```
CREATE VIEW financial_report AS

SELECT

    i.id_invoice,

    i.invoice_date,

    i.total_cost,

    COALESCE(SUM(p.amount), 0) AS paid_amount,

    (i.total_cost - COALESCE(SUM(p.amount), 0)) AS outstanding_amount

FROM

    invoice i

    LEFT JOIN payment p ON i.id_invoice = p.invoice

GROUP BY

    i.id_invoice;
```

employee_performance - Оценка выполнения задач сотрудниками.

- Задачи:
  - Анализ активности сотрудников.

```
CREATE VIEW employee_performance AS

SELECT

    e.id_employee,

    CONCAT(e.first_name, ' ', e.last_name) AS full_name,

    COUNT(er.request) AS assigned_requests,

    SUM(s.bonuses) AS total_bonuses

FROM

    employee e

    LEFT JOIN employee_request_map er ON e.id_employee = er.employee

    LEFT JOIN salary s ON e.id_employee = s.employee

GROUP BY

    e.id_employee;
```

3. Создайте список проверок для добавления в базу данных вместе с подсписком задач, выполняемых каждой проверкой. Создайте соответствующие проверки.

Проверки для таблицы request

- Дата завершения не может быть раньше даты запроса**:**

```
ALTER TABLE request

ADD CONSTRAINT CHK_request_completion_date CHECK (completion_date
IS NULL OR completion_date >= request_date);
```

- Дата запроса не может быть в будущем:

```
ALTER TABLE request

ADD CONSTRAINT CHK_request_date_not_future CHECK (request_date <=
CURDATE());
```

Проверки для таблицы invoice

- Стоимость счета должна быть положительной:

```
ALTER TABLE invoice
ADD CONSTRAINT CHK_invoice_total_cost CHECK (total_cost >= 0);
```

Проверки для таблицы provided_service

- Стоимость услуги должна быть положительной:

```
ALTER TABLE provided_service
ADD CONSTRAINT CHK_provided_service_cost CHECK (cost > 0);
```

- Дата услуги не может быть раньше даты запроса:

```
 ALTER TABLE provided_service
 ADD CONSTRAINT CHK_provided_service_date CHECK (service_date >=
(SELECT MIN(request_date) FROM request WHERE id_request =
provided_service.request));
```

Проверки для таблицы part

- Количество запчастей должно быть положительным:

```
ALTER TABLE part
ADD CONSTRAINT CHK_part_quantity CHECK (quantity > 0);
```

- Цена за единицу запчасти должна быть положительной:

```
ALTER TABLE part
ADD CONSTRAINT CHK_part_cost_per_unit CHECK (cost_per_unit > 0);
```

Проверки для таблицы contract

- Дата начала контракта не может быть позже даты окончания:

```
ALTER TABLE contract
ADD CONSTRAINT CHK_contract_date CHECK (contract_start_date <=
contract_end_date);
```

Проверки для таблицы payment

- Сумма платежа должна быть положительной:

```
ALTER TABLE payment
ADD CONSTRAINT CHK_payment_amount CHECK (amount > 0);
```

- Метод оплаты не может быть пустым:

```
ALTER TABLE payment
ADD CONSTRAINT CHK_payment_method_not_empty CHECK (payment_method <>
'');
```

Проверки для таблицы salary

- Базовая зарплата должна быть положительной**:**

```
ALTER TABLE salary
ADD CONSTRAINT CHK_salary_base CHECK (base_salary >= 0);
```

- Бонусы не могут быть отрицательными**:**

```
ALTER TABLE salary
ADD CONSTRAINT CHK_salary_bonuses CHECK (bonuses >= 0);
```

Проверки для таблицы supply

- Количество поставки должно быть положительным**:**

```
ALTER TABLE supply
ADD CONSTRAINT CHK_supply_quantity CHECK (quantity > 0);
```

- Дата поставки не может быть в будущем**:**

```
ALTER TABLE supply
ADD CONSTRAINT CHK_supply_date_not_future CHECK (supply_date <=
CURDATE());
```

4. **Создайте список триггеров, которые нужно добавить в базу данных, а также подсписок задач, выполняемых каждым триггером. Создайте соответствующие триггеры.**

Триггер для таблицы invoice

- **Задачи:**
  - Установка текущей даты при создании счета.

```
CREATE TRIGGER before_invoice_insert

BEFORE INSERT ON invoice

FOR EACH ROW

BEGIN

  SET NEW.invoice_date = CURDATE();

END$$
```

Триггер для таблицы provided_service

- **Задачи:**
  - Синхронизация с общей стоимостью в таблице invoice.

```
CREATE TRIGGER after_provided_service_insert

AFTER INSERT ON provided_service

FOR EACH ROW

BEGIN

  UPDATE invoice

  SET total_cost = total_cost + NEW.cost

  WHERE id_invoice = NEW.invoice;

END$$
```

Триггер для таблицы service_part

- **Задачи:**
  - Синхронизация с общей стоимостью в таблице invoice.

```
CREATE TRIGGER after_service_part_insert

AFTER INSERT ON service_part

FOR EACH ROW

BEGIN

  UPDATE invoice

  SET total_cost = total_cost + (NEW.quantity * NEW.cost_per_unit)

  WHERE id_invoice = NEW.invoice;

END$$
```

5. Определите, какие хранимые подпрограммы необходимы для реализации перечня требований заказчика к проектируемой базе данных.

Номер подпрограммы	Имя	Описание
1	GET_INVOICE_SUM	Процедура для получения общей суммы всех счетов для конкретного клиента.
2	GET_TASKS_BY_EMPLOYEE	Процедура для получения всех задач, назначенных конкретному сотруднику, включая информацию о запросах и услугах.
3	GET_PARTS_USAGE	Процедура для получения статистики использования деталей по каждому типу услуги или запросу.
4	GET_SUPPLIER_CONTRACTS	Процедура для получения всех контрактов с поставщиками, включая детали контрактов и поставки деталей.
5	GET_EMPLOYEE_SALARY	Процедура для получения информации о зарплате сотрудника, включая основные выплаты и бонусы.

6. Создайте список хранимых подпрограмм, которые будут добавлены в базу данных, а также подсписок задач, выполняемых каждой подпрограммой. Создайте соответствующие подпрограммы.

1. GET_INVOICE_SUM

```
CREATE PROCEDURE GET_INVOICE_SUM(IN client_id INT)
BEGIN
    SELECT SUM(i.total_cost) AS total_invoice_sum
    FROM invoice i
    WHERE i.client = client_id;
END $$
```

2. GET_TASKS_BY_EMPLOYEE

```
CREATE PROCEDURE GET_TASKS_BY_EMPLOYEE(IN employee_id INT)
BEGIN
    SELECT r.id_request, r.request_date, r.completion_date,
ps.service_description
    FROM request r
    JOIN employee_request_map erm ON r.id_request = erm.request
    JOIN provided_service ps ON r.id_request = ps.request
    WHERE erm.employee = employee_id;
```

```
END $$
```

3. GET_PARTS_USAGE

```
CREATE PROCEDURE GET_PARTS_USAGE()
BEGIN
    SELECT p.part_name, SUM(sp.quantity) AS total_used
    FROM part p
    JOIN service_part sp ON p.id_part = sp.part
    GROUP BY p.id_part;
END $$
```

4. GET_SUPPLIER_CONTRACTS

```
CREATE PROCEDURE GET_SUPPLIER_CONTRACTS(IN supplier_id INT)
BEGIN
    SELECT c.id_contract, c.contract_start_date, c.contract_end_date
    FROM contract c
    WHERE c.supplier = supplier_id;
END $$
```

5. GET_EMPLOYEE_SALARY

```
CREATE PROCEDURE GET_EMPLOYEE_SALARY(IN employee_id INT)
BEGIN
    SELECT s.base_salary, s.bonuses
    FROM salary s
    WHERE s.employee = employee_id;
END $$
```

Итоговый вариант