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.

1. **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;

1. **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());

1. **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$$

1. **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. |

1. **Создайте список хранимых подпрограмм, которые будут добавлены в базу данных, а также подсписок задач, выполняемых каждой подпрограммой. Создайте соответствующие подпрограммы.**
2. 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 $$

1. 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 $$

1. 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 $$

1. 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 $$

1. 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**

