

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки

Кафедра системотехніки

ЗВІТ
з виконання завдань практичного заняття № 3
дисципліни «Проектування високонавантажених систем
зберігання даних»
на тему: «РЕАЛІЗАЦІЯ ПРОГРАМНОЇ ЛОГІКИ Й ЗАБЕЗПЕЧЕННЯ
ЦЛІСНОСТІ ЗВ'ЯЗКІВ ЗА ДОПОМОГОЮ ТРИГЕРІВ ДЛЯ
ВИСОКОНАВАНТАЖЕНИХ БАЗ ДАНИХ
НА ПЛАТФОРМІ СУБД MySQL»

Виконав
студент WILDAU - KHARKIV
Якунін Ігор

Перевірив
професор кафедри СТ
Колесник Л.В.

Харків, 2024

1.1 Мета заняття

- набуття практичних навичок з розробки тригерів для підтримки цілісності зв'язків, модифікації даних і забезпечення основних бізнес-процесів високонавантаженої інформаційної системи;
- формування необхідних практичних умінь для аналізу плану виконання SQL-запитів за допомогою оператора EXPLAIN;
- формування необхідних практичних умінь для створення тригерів, з урахуванням особливостей реалізації логіки роботи інтерфейсу високонавантаженої інформаційної системи зберігання даних

Завдання на самостійну роботу

Обрати предметну область для виконання індивідуальних завдань.
Обрана область – «**Сайт сервісного центру по ремонту техніки**»

Таблиця 3.1 – Перелік доступних операцій тригера

Доступність операцій тригера	BEFORE RE INSER T	AFTER INSER T	BEFORE RE UPDA TE	AFTER UPDA TE	BEFORE RE DELET E	AFTER DELET E
1. NEW доступний – Так/Ні	Так	Так	Так	Так	Ні	Ні
2. OLD доступний – Так/Ні	Ні	Ні	Так	Ні	Так	Ні
3. Доступ до зміни полів NEW – Так/Ні	Так	Так	Так	Так	Ні	Ні
4. Значення автоінкремента (PK) доступно – Так/Ні	Так	Так	Ні	Ні	Ні	Ні
5. Доступ до значень полів та їх значень (DEFAULT), що явно в інструкціях INSERT, UPDATE, DELETE не фігурували – Так/Ні	Так	Ні	Так	Ні	Ні	Ні
6. Можливість скасування операцій INSERT / / UPDATE / DELETE – Так/Ні	Так	Ні	Так	Ні	Так	Ні

Таблиця 3.2 – Переваги використання тригерів

№	Аналізовані параметри	Тригер	SQL-запит
1	Принцип виконання SQL-коду в СУБД MySQL	виконуються автоматично при настанні певної події (наприклад, <code>INSERT</code> , <code>UPDATE</code> , <code>DELETE</code>) на конкретній таблиці. Вони запускаються безпосередньо в базі даних, та виконуються або до, або після потрібної події	виконуються тільки при явному виклику з боку клієнта (наприклад, програми або користувача) та виконуються на запит чи модифікацію даних
2	Вплив апаратні ресурси	можуть створювати додаткове навантаження на сервер бази даних, особливо при інтенсивних операціях (наприклад, масові <code>INSERT</code> або <code>UPDATE</code>). Це може впливати на продуктивність через додаткове використання процесора та пам'яті.	впливають на ресурси тільки в момент виконання. Контроль за їх кількістю і моментом виконання дозволяє оптимізувати ресурсомісткі операції для зменшення навантаження.
3	Вплив на мережевий трафік	виконуються на стороні сервера бази даних, тому не потребують додаткового трафіку для активації. Це може зменшити мережеве навантаження, оскільки тригери працюють без постійних запитів між	передаються по мережі кожного разу, коли викликаються з боку застосунку, що може збільшувати мережевий трафік при великій кількості звернень.

		застосунком та сервером бази даних.	
4	Можливість, що-небудь автоматизувати	підходять для автоматизації дій при певних подіях. Робить їх ефективними, наприклад, для забезпечення цілісності даних або реакція на події	автоматизація можлива, але вимагає додаткового програмного забезпечення або скриптів для запуску запитів за певним розкладом або у відповідь на події. Це може ускладнити підтримку автоматизації.
5	SQL-синтаксис (обмеження)	у MySQL мають обмеження. Наприклад, існують обмеження на виклик інших тригерів для уникнення безкінечних циклів, і підтримка DDL-операцій (наприклад, CREATE, ALTER) відсутня. Також не можна змінювати дані в таблиці, на якій викликаний тригер.	дозволяють повний контроль над синтаксисом і можливостями SQL, включаючи складні операції та налаштування. Їх можна легко комбінувати, оптимізувати та адаптувати до вимог застосунку.
6	Принцип (фізичне місце) зберігання SQL-коду	зберігаються в базі даних і прив'язані до певної таблиці. Це означає, що логіка обробки зберігається разом із даними, що може ускладнювати їх перенесення в інші системи чи бази даних.	зазвичай зберігаються в коді застосунку або у вигляді окремих SQL-скриптів, що полегшує їх редагування, зберігання та перенесення між різними базами чи застосунками.

Скріншот схеми фізичної моделі бази даних з таблицями типу InnoDB у нотації IDEF1X наведен на рис. 1.1

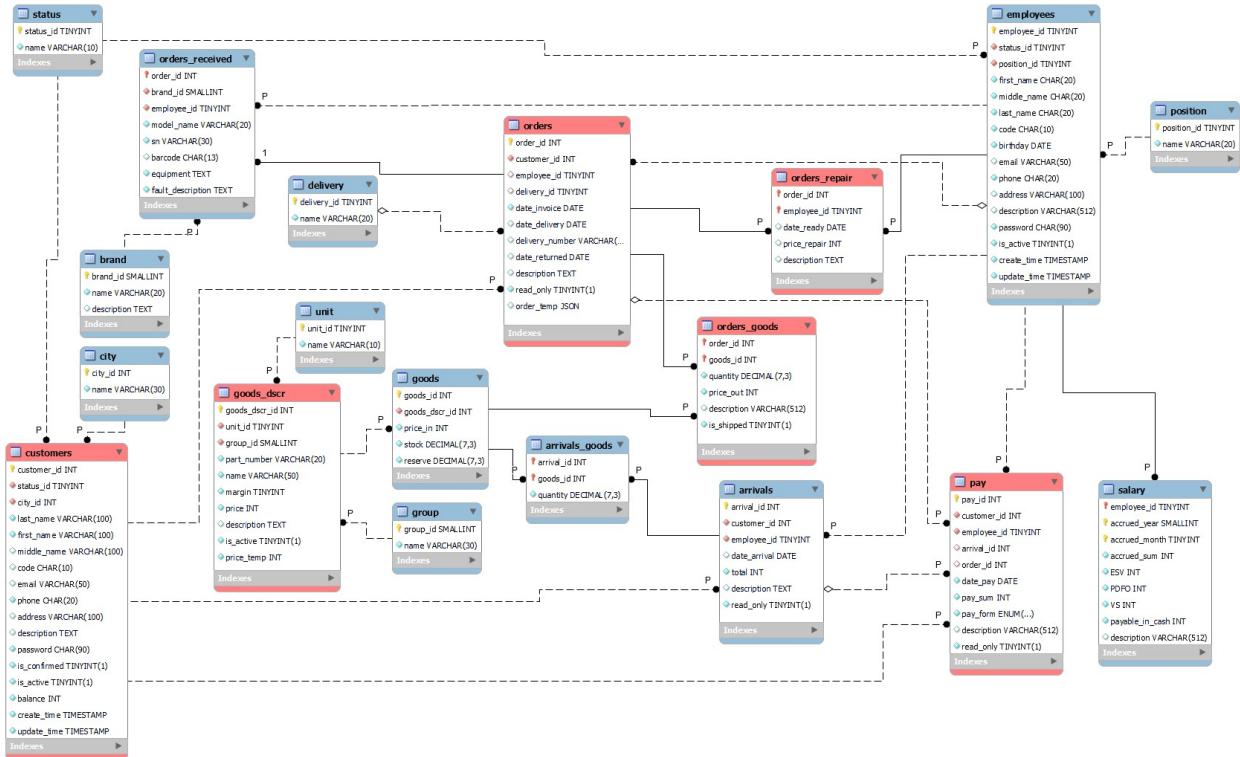


Рис. 1.1 - Скріншот схеми фізичної моделі бази даних з таблицями типу InnoDB

Завдання 3.1. Для бази даних з таблицями типу MyISAM, створеної відповідно до завдання 1.2 практичного заняття № 1, розробити тригери, що забезпечують цілісність зв'язків реалізованої схеми даних. Для всіх тригерів реалізувати скасування операції (INSERT/REPLACE; UPDATE; DELETE) у випадку невиконання умов цілісності зв'язків, з генерацією повідомлення за допомогою речення SIGNAL.

Таблиця 3.3 – Перелік типів посилальної цілісності зв'язків за зовнішнім ключем для всіх таблиць типу Mylsam.

№	Ім'я таблиці 1, зовнішній ключ	Ім'я таблиці 2, первинний ключ	SQL інструкція для таблиці 1	Тип посилальної цілісності	Тригер
1	customers, status_id	status, status_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
2	customers, city_id	city, city_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
3	goods_dscr, unit_id	unit, unit_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
4	goods_dscr, group_id	group, group_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
5	goods, goods_dscr_id	goods_dscr, goods_dscr_id	Update	CASCADE	Before
			Delete	CASCADE	Before
			Insert	CASCADE	Before, After
6	orders_goods, goods_id	goods, goods_id	Update	CASCADE	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
7	orders_goods, order_id	orders, order_id	Update	CASCADE	Before
			Delete	CASCADE	Before
			Insert	RESTRICT	Before

Таблиця 3.3, продовження

8	orders_repair, order_id	orders, order_id	Update	CASCADE	Before
			Delete	RESTRICT	Before
			Insert	CASCADE	Before, After
9	orders_repair, employee_id	employees, employee_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
10	orders_received, order_id	orders, order_id	Update	CASCADE	Before
			Delete	RESTRICT	Before
			Insert	CASCADE	Before, After
11	orders_received, brand_id	brand, brand_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
12	orders_received, employee_id	employees, employee_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
13	orders, employee_id	employees, employee_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
14	orders, delivery_id	delivery, delivery_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
15	orders, customer_id	customers, customer_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
16	salary, employee_id	employees, employee_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
17	pay, employee_id	employees, employee_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before

Таблиця 3.3, продовження

18	pay, customer_id	customers, customer_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
19	pay, arrival_id	arrivals, arrival_id	Update	CASCADE	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
20	pay, order_id	orders, order_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
21	arrivals, employee_id	employees, employee_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
22	arrivals, customer_id	customers, customer_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
23	arrival_goods, arrival_id	arrivals, arrival_id	Update	CASCADE	Before
			Delete	CASCADE	Before
			Insert	RESTRICT	Before
24	arrival_goods, good_id	goods, good_id	Update	CASCADE	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
25	employees, status_id	status, status_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before
26	employees, position_id	position, position_id	Update	RESTRICT	Before
			Delete	RESTRICT	Before
			Insert	RESTRICT	Before

SQL-код тригерів для усіх таблиць типу MyIsam

3.1.1. Таблиця customers

3.1.1.1 Before Insert

```
DELIMITER //  
CREATE TRIGGER customers_before_insert  
BEFORE INSERT ON customers  
FOR EACH ROW  
BEGIN  
    -- Перевірка, чи існує вказаний city_id в таблиці city  
    IF (SELECT COUNT(*) FROM city WHERE city.city_id = NEW.city_id) = 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: вказаний city_id не існує в таблиці city.';  
    END IF;  
  
    -- Перевірка, чи існує вказаний status_id в таблиці status  
    IF (SELECT COUNT(*) FROM status WHERE status.status_id = NEW.status_id) = 0  
    THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: вказаний status_id не існує в таблиці status.';  
    END IF;  
  
    -- Перевірка, чи телефон є унікальним  
    IF (SELECT COUNT(*) FROM customers WHERE phone = NEW.phone) > 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: телефон вже існує в базі даних.';  
    END IF;  
  
    -- Перевірка, чи всі обов'язкові поля заповнені  
    IF NEW.city_id IS NULL OR  
        NEW.status_id IS NULL OR  
        NEW.last_name IS NULL OR  
        NEW.first_name IS NULL OR  
        NEW.phone IS NULL OR
```

```

NEW.password IS NULL THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: всі обов'язкові поля мають бути заповнені.';

END IF;
END//;

DELIMITER ;

```

До вставки даних

```

1 •  SELECT customer_id, city_id, status_id, last_name, first_name, phone, password
2   FROM ss_myisam2.customers
3   ORDER BY customer_id DESC;

```

	customer_id	city_id	status_id	last_name	first_name	phone	password
▶	10001	1	1	Кучеров	Степан	+380987654321	asaw34rsfa
	10000	6	1	Гузій	Марія	+380 99 946 57 37	\$pbkdf2-sha256\$29000\$irEWAqDU
	9999	6	8	Тимчук	Давид	+380 96 827 46 92	\$pbkdf2-sha256\$29000\$35uTsnbO

Після вставки даних

	customer_id	city_id	status_id	last_name	first_name	phone	password
	10002	1	1	Кучеров	Василь	+380987654320	asaw34rsfa
	10001	1	1	Кучеров	Степан	+380987654321	asaw34rsfa

Помилки при вставці даних, що визвав тригер

```

④ 10 16:18:17 INSERT INTO customers(city_id, status_id, last_name, first_name, phone, description, password) VA... Error Code: 1644. Помилка: вказаний city_id не існує в таблиці city.
④ 11 16:18:37 INSERT INTO customers(city_id, status_id, last_name, first_name, phone, description, password) VA... Error Code: 1644. Помилка: вказаний status_id не існує в таблиці status.
④ 12 16:18:43 INSERT INTO customers(city_id, status_id, last_name, first_name, phone, description, password) VA... Error Code: 1644. Помилка: телефон вже існує в базі даних.

```

Рис. 3.1.1.1 Робота тригера Before Insert при вставці даних в таблицю customers

3.1.1.2 Before Update

```

DELIMITER //

CREATE TRIGGER customers_before_update
BEFORE UPDATE ON customers
FOR EACH ROW
BEGIN
    -- Оновлення поля update_time при кожному оновленні
    SET NEW.update_time = CURRENT_TIMESTAMP;

```

```

-- Перевірка, чи існує новий city_id в таблиці city
IF NEW.city_id IS NOT NULL AND
  (SELECT COUNT(*) FROM city WHERE city.city_id = NEW.city_id) = 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: вказаний city_id не існує в таблиці city.';

  END IF;

-- Перевірка, чи існує новий status_id в таблиці status
IF NEW.status_id IS NOT NULL AND
  (SELECT COUNT(*) FROM status WHERE status.status_id = NEW.status_id) = 0
THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: вказаний status_id не існує в таблиці status.';

  END IF;

-- Перевірка, чи телефон є унікальним, якщо він змінюється
IF NEW.phone <> OLD.phone AND
  (SELECT COUNT(*) FROM customers WHERE phone = NEW.phone) > 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: телефон вже існує в базі даних.';

  END IF;

-- Перевірка, щоб обов'язкові поля не були NULL
IF NEW.city_id IS NULL OR
  NEW.status_id IS NULL OR
  NEW.last_name IS NULL OR
  NEW.first_name IS NULL OR
  NEW.phone IS NULL OR
  NEW.password IS NULL THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: всі обов'язкові поля мають бути заповнені.';

  END IF;

-- Оновлення зв'язків для збереження цілісності даних, якщо customer_id змінюється
IF NEW.customer_id <> OLD.customer_id THEN

```

```

UPDATE orders SET customer_id = NEW.customer_id WHERE customer_id =
OLD.customer_id;

UPDATE arrivals SET customer_id = NEW.customer_id WHERE customer_id =
OLD.customer_id;

UPDATE pay SET customer_id = NEW.customer_id WHERE customer_id =
OLD.customer_id;

END IF;

END//;

DELIMITER ;

```

До оновлення даних

```

2 •  SELECT customer_id, city_id, status_id, last_name, first_name, phone
3   FROM ss_myisam2.customers
4 WHERE customer_id IN (9990, 10003)
5 ORDER BY customer_id DESC;

```

Result Grid					
customer_id	city_id	status_id	last_name	first_name	phone
9990	6	8	Гайворонський	Віктор	+380 (95) 462-24-58
NULL	NULL	NULL	NULL	NULL	NULL

```

3 •  SELECT arrival_id, customer_id          3 •  SELECT order_id, customer_id
4   FROM arrivals                          4   FROM orders
5 WHERE customer_id IN (9990, 10003);      5 WHERE customer_id IN (9990, 10003);

```

result Grid		result Grid	
arrival_id	customer_id	order_id	customer_id
620	9990	6224	9990
NULL	NULL	NULL	NULL

```

3 •  SELECT *
4   FROM ss_myisam2.pay
5 WHERE customer_id IN (9990, 10003);

```

result Grid				
pay_id	customer_id	employee_id	arrival_id	order_id
4039	9990	19	NULL	6224
5460	9990	2	NULL	6224
10619	9990	14	620	NULL
NULL	NULL	NULL	NULL	NULL

```

43 20:32:27 SET SQL_SAFE_UPDATES = 0          0 row(s) affected
44 20:32:33 UPDATE ss_myisam2.customers SET customer_id = 10003 WHERE customer_id = 9990      1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

```

Після оновлення даних

customer_id	city_id	status_id	last_name	first_name	phone
10003	7	7	Гайворонський	Віктор	+380954622458

arrival_id	customer_id	order_id	customer_id	pay_id	customer_id	arrival_id	order_id
620	10003	6224	10003	4039	10003	NULL	6224

Помилки при оновленні даних, що визвав тригер

- ③ 38 20:27:57 UPDATE ss_myisam2.customers SET city_id = 17, status_id = 7, phone = '+380954622458' WHERE customer_id = 10003 Error Code: 1644. Помилка: вказаний city_id не існує в таблиці city.
- ③ 39 20:28:17 UPDATE ss_myisam2.customers SET status_id = 17, phone = '+380954622458' WHERE customer_id = 10003 Error Code: 1644. Помилка: вказаний status_id не існує в таблиці status.
- ③ 40 20:28:52 UPDATE ss_myisam2.customers SET phone = '+380987654320' WHERE customer_id = 9990 Error Code: 1644. Помилка: телефон вже існує в базі даних.
- ③ 41 20:29:46 UPDATE ss_myisam2.customers SET phone = NULL WHERE customer_id = 9990 Error Code: 1644. Помилка: всі обов'язкові поля мають бути заповнені.

Рис. 3.1.1.2 Робота тригеру Before Update при оновленні даних в таблицю customers

3.1.1.3 Before Delete (RESTRICT)

```

DELIMITER //

CREATE TRIGGER customers_before_delete
BEFORE DELETE ON customers
FOR EACH ROW
BEGIN
    -- Перевірка, чи користувач є активним
    IF OLD.is_active <> 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: користувач активний, неможливо
видалити。';
    END IF;
    -- Перевірка наявності пов'язаних записів у таблиці orders
    IF (SELECT COUNT(*) FROM orders WHERE customer_id = OLD.customer_id) > 0
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: Неможливо видалити запис, оскільки існують
пов'язані записи в таблиці orders。';
    END IF;

```

```

-- Перевірка наявності пов'язаних записів у таблиці arrivals
IF (SELECT COUNT(*) FROM arrivals WHERE customer_id = OLD.customer_id) >
0 THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: Неможливо видалити запис, оскільки існують
пов'язані записи в таблиці arrivals.';

END IF;

-- Перевірка наявності пов'язаних записів у таблиці pay
IF (SELECT COUNT(*) FROM pay WHERE customer_id = OLD.customer_id) > 0
THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: Неможливо видалити запис, оскільки існують
пов'язані записи в таблиці pay.';

END IF;
END//;
DELIMITER ;

```

До видалення даних

```

3 •  SELECT customer_id, city_id, status_id, last_name, first_name, is_active
4      FROM ss_myisam2.customers
5      WHERE customer_id IN (10001, 10002, 10003);

```

customer_id	city_id	status_id	last_name	first_name	is_active
10001	1	1	Кучеров	Степан	0
10002	1	1	Кучеров	Василь	1
10003	7	7	Гайворонський	Віктор	0
NULL	NULL	NULL	NULL	NULL	NULL

```

3 •  SELECT arrival_id, customer_id
4      FROM arrivals
5      WHERE customer_id IN (10001, 10002, 10003);

```

arrival_id	customer_id
620	10003
NULL	NULL

```

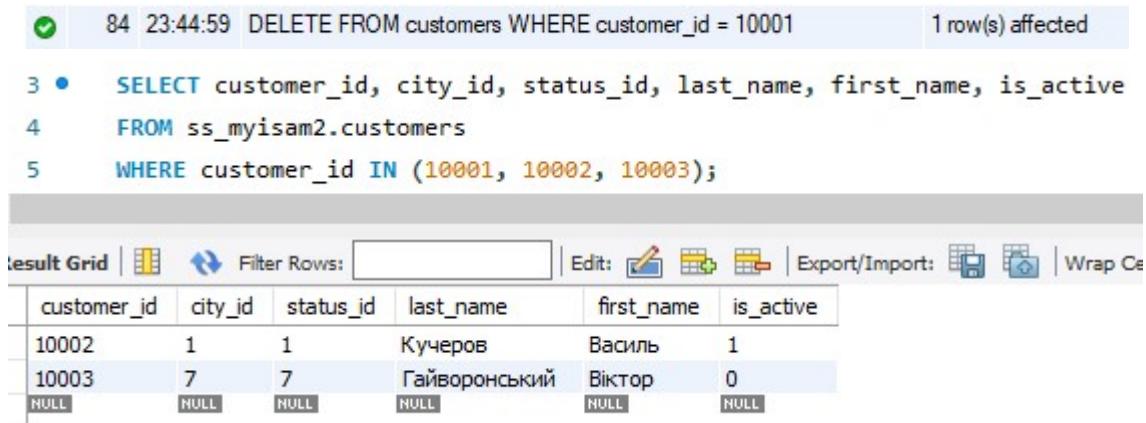
3 •  SELECT order_id, customer_id
4      FROM orders
5      WHERE customer_id IN (10001, 10002, 10003);

```

order_id	customer_id
6224	10003
NULL	NULL

Рис. 3.1.1.3 Робота триггеру Before Delete при видалені даних з таблиці customers

Після видалення даних



```
84 23:44:59 DELETE FROM customers WHERE customer_id = 10001 1 row(s) affected
3 •  SELECT customer_id, city_id, status_id, last_name, first_name, is_active
4   FROM ss_myisam2.customers
5   WHERE customer_id IN (10001, 10002, 10003);

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Ce
customer_id | city_id | status_id | last_name | first_name | is_active
10002       | 1        | 1          | Кучеров  | Василь    | 1
10003       | 7        | 7          | Гайворонський | Віктор | 0
NULL        | NULL    | NULL      | NULL     | NULL     | NULL
```

Помилки при видаленні даних, що визвав тригер

81 23:38:46 DELETE FROM customers WHERE customer_id = 10003 Error Code: 1644. Помилка: Неможливо видалити запис, оскільки існують пов'язані записи в таблиці orders.

82 23:39:27 DELETE FROM customers WHERE customer_id = 10002 Error Code: 1644. Помилка: користувач активний, неможливо видалити.

Рис. 3.1.1.3 Робота тригера Before Delete при видалені даних з таблиці customers (продовження)

3.1.1.4 Before Update (перебудова у RESTRICT)

При проектуванні тригера Before Update до таблиці customers не були враховані деякі нюанси побудови БД, зокрема таблиц orders, arrivals, pay (ці таблиці розглянуті нижче, але я повертаюсь к розгляду цього тригера вже після розгляду вищезгаданих таблиць, та таблиці employees). При створенні тригерів до таблиці employees було вирішено не використовувати тригер **Update CASCADE** ввіду не надійності його роботи при деяких окремих випадках, наприклад, коли запис в обновлюемої таблиці закрит для редагування, тригер зупинить свою роботу на півдорозі, частина записів буде оновлена, частина – ні. Тому я вирішив використати тригер **Update RESTRICT** як для таблиці employees, так і для таблиці customers, яка дуже схожа з employees.

Код тригера після корекції та скриншоти його випробування:

```
DELIMITER //
DROP TRIGGER IF EXISTS customers_before_update;
```

```
CREATE TRIGGER customers_before_update
BEFORE UPDATE ON customers
FOR EACH ROW
BEGIN
    -- Оновлення поля update_time при кожному оновленні
    SET NEW.update_time = CURRENT_TIMESTAMP;

    -- Перевірка, чи існує новий city_id в таблиці city
    IF NEW.city_id <> OLD.city_id AND NOT EXISTS
        (SELECT 1 FROM city WHERE city.city_id = NEW.city_id) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, customers: вказаний city_id не існує в таблиці
        city.';

    END IF;

    -- Перевірка, чи існує новий status_id в таблиці status
    IF NEW.status_id <> OLD.status_id AND NOT EXISTS
        (SELECT 1 FROM status s WHERE s.status_id = NEW.status_id) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, customers: вказаний status_id не існує в таблиці
        status.';

    END IF;

    -- Перевірка, чи телефон є унікальним, якщо він змінюється
    IF NEW.phone <> OLD.phone AND EXISTS
        (SELECT 1 FROM customers WHERE phone = NEW.phone) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, customers: телефон вже існує в базі даних.';

    END IF;

    -- Перевірка, щоб обов'язкові поля не були NULL
    IF NEW.city_id IS NULL OR
        NEW.status_id IS NULL OR
        NEW.last_name IS NULL OR
```

```

    NEW.first_name IS NULL OR
    NEW.phone IS NULL OR
    NEW.password IS NULL THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, customers: всі обов'язкові поля мають бути
заповнені.';

END IF;

-- Зміна customer_id заборонена
IF NEW.customer_id <> OLD.customer_id THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, customers: зміна customer_id
заборонена.';

END IF;
END//  

DELIMITER ;

```

До оновлення даних

```

1 •  SELECT customer_id, city_id, status_id, last_name, first_name, phone, update_time
2   FROM ss_myisam2.customers ORDER BY customer_id DESC;
3
4 •  UPDATE customers
5   SET city_id = 1, status_id = 1, last_name = 'Гайворонська',
6       first_name = "Вікторія", phone = "+380954622455"
7   WHERE customer_id = 10003;

```

result Grid						
customer_id	city_id	status_id	last_name	first_name	phone	update_time
10003	7	7	Гайворонський	Віктор	+380954622458	2024-11-18 23:38:15
10002	1	1	Кучеров	Василь	+380987654320	2024-11-18 16:19:33

Після оновлення даних

27 19:35:48 UPDATE customers SET city_id = 1, status_id = 1, last_name = 'Гайворонська', first_name = "Вікторія", phone = "+3... 1 row(s) affected Rows matched: 1 Changed: 1

customer_id	city_id	status_id	last_name	first_name	phone	update_time
10003	1	1	Гайворонська	Вікторія	+380954622455	2024-11-25 19:35:48
10002	1	1	Кучеров	Василь	+380987654320	2024-11-18 16:19:33

Помилки при оновлені даних, що визвав тригер

✖	29	19:40:05	UPDATE customers SET city_id = 21, status_id = 1, last_name = 'Гайворонська', first_name = 'Bi...	Error Code: 1644. Помилка, customers: вказаний city_id не існує в таблиці city.
✖	30	19:40:20	UPDATE customers SET city_id = 1, status_id = 11, last_name = 'Гайворонська', first_name = 'Bi...	Error Code: 1644. Помилка, customers: вказаний status_id не існує в таблиці status.
✖	31	19:41:05	UPDATE customers SET city_id = 1, status_id = 1, last_name = NULL, first_name = 'Вікторія', ph...	Error Code: 1644. Помилка, customers: всі обов'язкові поля мають бути заповнені.
✖	32	19:41:57	UPDATE customers SET phone = "+380987654320" WHERE customer_id = 10003	Error Code: 1644. Помилка, customers: телефон вже існує в базі даних.
✓	33	19:42:24	SET SQL_SAFE_UPDATES = 0	0 row(s) affected
✖	34	19:42:47	UPDATE customers SET customer_id = 10004 WHERE customer_id = 10003	Error Code: 1644. Помилка, customers: зміна customer_id заборонена.

3.1.2. Таблиця goods_dscr

Я вирішив, що в даному випадку зручно використати тригер **Insert Cascade**, та одночасно заповнити одразу дві таблиці goods_dscr та goods. При додаванні нового найменування запчастини, якщо буде отразу додана вхідна ціна, то спрацює тригер **After Insert** і додаст нову ціну для нової запчастини (щоб цей функціонал працював я додав до таблиці goods_dscr тимчасове поле price_temp).

3.1.2.1.a Before Insert

```
DELIMITER //  
CREATE TRIGGER goods_dscr_before_insert  
BEFORE INSERT ON goods_dscr  
FOR EACH ROW  
BEGIN  
    -- Перевірка, чи існує вказаний unit_id в таблиці unit  
    IF (SELECT COUNT(*) FROM unit u WHERE u.unit_id = NEW.unit_id) = 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: вказаний unit_id не існує в таблиці unit.';  
    END IF;  
  
    -- Перевірка, чи існує вказаний group_id в таблиці group  
    IF (SELECT COUNT(*) FROM `group` g WHERE g.group_id = NEW.group_id) = 0  
    THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: вказаний group_id не існує в таблиці group.';
```

```

END IF;

-- Перевірка, чи part_number є унікальним
IF (SELECT COUNT(*) FROM customers WHERE part_number =
NEW.part_number) > 0 THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: вказаний part_number вже існує в базі даних。';
END IF;

-- Перевірка, чи name є унікальним
IF (SELECT COUNT(*) FROM customers WHERE name = NEW.name) > 0 THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: вказаний name вже існує в базі даних。';
END IF;

-- Перевірка, чи всі обов'язкові поля заповнені
IF NEW.unit_id IS NULL OR
    NEW.group_id IS NULL OR
    NEW.part_number IS NULL OR
    NEW.name IS NULL THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: всі обов'язкові поля мають бути заповнені。';
END IF;

END//;

DELIMITER ;

```

До вставки

```

1 •  SELECT goods_dscr_id, unit_id, group_id, part_number, name
2      FROM ss_myisam2.goods_dscr
3      ORDER BY goods_dscr_id DESC;
4
5 •  INSERT INTO goods_dscr (unit_id, group_id, part_number, name)
6      VALUES(1, 1, '4354-dfgsdg', 'Картридж 1615');

```

Result Grid | Filter Rows: | Edit: | Export/Import:

goods_dscr_id	unit_id	group_id	part_number	name
10001	1	1	2345125-21413	Товар 4XAWK
10000	1	4	HOOZM1ATCI	Товар KRA34
9999	6	13	7HUVBOVX8R	Товар 7S8IR

Після вставки

goods_dscr_id	unit_id	group_id	part_number	name
10002	1	1	4354-dfgsdg	Картридж 1615
10001	1	1	2345125-21413	Товар 4XAWK
10000	1	4	HOOZM1ATCI	Товар KRA34

Помилки при вставці даних, що визвав тригер

```

✖ 13 22:38:51 INSERT INTO goods_dscr (unit_id, group_id, part_number, name) VALUES(21, 1, '4354-dfgsdg', 'Картридж 1615') Error Code: 1644. Помилка: вказаний unit_id не існує в таблиці unit.
✖ 14 22:39:14 INSERT INTO goods_dscr (unit_id, group_id, part_number, name) VALUES(1, 21, '4354-dfgsdg', 'Картридж 1615') Error Code: 1644. Помилка: вказаний group_id не існує в таблиці group.

✖ 17 22:41:29 INSERT INTO goods_dscr (unit_id, group_id, part_number, name) VALUES(1, 1, '4354-dfgsdg', 'Товар 4XAWK') Error Code: 1644. Помилка: вказаний part_number вже існує в базі даних.
✖ 18 22:41:46 INSERT INTO goods_dscr (unit_id, group_id, part_number, name) VALUES(1, 1, '4354-dfgsdg1', 'Товар 4XAWK') Error Code: 1644. Помилка: вказаний name вже існує в базі даних.

```

Рис. 3.1.2.1а Робота тригеру Before Insert при вставці даних в таблицю

goods_dscr

3.1.2.1.b After Insert (CASCADE)

DELIMITER //

```

CREATE TRIGGER goods_dscr_after_insert
AFTER INSERT ON goods_dscr
FOR EACH ROW
BEGIN
    -- Перевіряємо, чи переданий price_temp
    IF NEW.price_temp IS NOT NULL THEN
        -- Вставляємо дані у пов'язану таблицю
        INSERT INTO goods (goods_dscr_id, price_in)
        VALUES (NEW.goods_dscr_id, NEW.price_temp);
    END IF;
END//
```

DELIMITER ;

До вставки

```

4 •  SELECT goods_dscr_id, unit_id, group_id, name, price_temp
5   FROM ss_myisam2.goods_dscr
6   ORDER BY goods_dscr_id DESC;
7
8 •  INSERT INTO goods_dscr (unit_id, group_id, part_number, name, price_temp)
9   VALUES(1, 1, '4354-dfgs231', 'Картридж 1210', 30000);

```

goods_dscr_id	unit_id	group_id	name	price_temp
10002	1	1	Картридж 1615	0
10000	1	4	Товар KRA34	0

```

12 •  SELECT *
13   FROM ss_myisam2.goods
14   ORDER BY goods_id DESC;

```

Result Grid | Filter Rows: | Edit: 

	goods_id	goods_dscr_id	price_in	stock	reserve
▶	10002	10000	4581	0.000	0.000
	10000	10000	4650	1.000	0.000

Після вставки

```

4 •  SELECT goods_dscr_id, unit_id, group_id, name, price_temp
5   FROM ss_myisam2.goods_dscr
6   ORDER BY goods_dscr_id DESC;

```

Result Grid | Filter Rows: | Edit:    | Export/Import:

goods_dscr_id	unit_id	group_id	name	price_temp
10004	1	1	Картридж 1210	30000
10002	1	1	Картридж 1615	0
10000	1	4	Товар KRA34	0

```

12 •  SELECT *
13   FROM ss_myisam2.goods
14   ORDER BY goods_id DESC;

```

Result Grid | Filter Rows: | Edit: 

	goods_id	goods_dscr_id	price_in	stock	reserve
▶	10003	10004	30000	0.000	0.000
	10002	10000	4581	0.000	0.000
	10000	10000	4650	1.000	0.000

Рис. 3.1.2.1b Робота тригера After Insert (CASCADE) при вставці даних в таблицю goods_dscr (goods)

3.1.2.2 Before Update

```

DELIMITER //
CREATE TRIGGER goods_dscr_before_update
BEFORE UPDATE ON goods_dscr
FOR EACH ROW
BEGIN

  -- Обнуляємо тимчасове поле
  IF OLD.price_temp <> 0 THEN

```

```

        SET NEW.price_temp = 0;
    END IF;

    -- Перевірка, чи існує новий unit_id в таблиці unit
    IF NEW.unit_id IS NOT NULL AND NOT EXISTS
        (SELECT 1 FROM unit u WHERE u.unit_id = NEW.unit_id) THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка: вказаний unit_id не існує в таблиці unit.';
    END IF;

    -- Перевірка, чи існує новий group_id в таблиці group
    IF NEW.group_id IS NOT NULL AND NOT EXISTS
        (SELECT 1 FROM `group` g WHERE g.group_id = NEW.group_id) THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка: вказаний group_id не існує в таблиці group.';
    END IF;

    -- Перевірка, чи part_number є унікальним, якщо він змінюється
    IF NEW.part_number <> OLD.part_number AND
        (SELECT 1 FROM goods_dscr WHERE part_number = NEW.part_number) THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка: вказаний part_number вже існує в базі даних.';
    END IF;

    -- Перевірка, чи name є унікальним, якщо він змінюється
    IF NEW.name <> OLD.name AND
        (SELECT 1 FROM goods_dscr WHERE name = NEW.name) THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка: вказаний name вже існує в базі даних.';
    END IF;

    -- Перевірка, щоб обов'язкові поля не були NULL
    IF NEW.unit_id IS NULL OR
        NEW.group_id IS NULL OR

```

```

NEW.part_number IS NULL OR
NEW.name IS NULL THEN
  SIGNAL SQLSTATE '45000'
  SET MESSAGE_TEXT = 'Помилка: всі обов'язкові поля мають бути заповнені.';

END IF;

-- Оновлення зв'язків для збереження цілісності даних, якщо goods_dscr_id змінюється
IF NEW.goods_dscr_id <> OLD.goods_dscr_id THEN
  UPDATE goods SET goods_dscr_id = NEW.goods_dscr_id WHERE
  goods_dscr_id = OLD.goods_dscr_id;
END IF;

END//;

DELIMITER ;

```

До оновлення даних

```

1 •  SELECT goods_dscr_id, unit_id, group_id, part_number, name
2   FROM ss_myisam2.goods_dscr
3   WHERE goods_dscr_id IN (10001, 10003);
4
5 •  SET SQL_SAFE_UPDATES = 0;
6 •  UPDATE ss_myisam2.goods_dscr
7   SET goods_dscr_id = 10003
8   WHERE goods_dscr_id = 10001;

```

result Grid | Filter Rows: Edit: Export/Import:

goods_dscr_id	unit_id	group_id	part_number	name
10001	1	1	2345125-21413	Товар 4XAWK
NULL	NULL	NULL	NULL	NULL

```

3 •  SELECT *
4   FROM ss_myisam2.goods
5   WHERE goods_dscr_id IN (10001);

```

result Grid | Filter Rows: Edit:

goods_id	goods_dscr_id	price_in	stock	reserve
10001	10001	10000	0.000	0.000
NULL	NULL	NULL	NULL	NULL

Після оновлення даних

30	23:46:50	SET SQL_SAFE_UPDATES = 0	0 row(s) affected															
31	23:46:54	UPDATE ss_myisam2.goods_dscr SET goods_dscr_id = 10003 WHERE goods_dscr_id = 10001	1 row(s) affected Rows matched: 1 Changed: 1															
<table border="1"><thead><tr><th>goods_dscr_id</th><th>unit_id</th><th>group_id</th><th>part_number</th><th>name</th></tr></thead><tbody><tr><td>10003</td><td>1</td><td>1</td><td>2345125-21413</td><td>Товар 4XAWK</td></tr><tr><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td></tr></tbody></table>				goods_dscr_id	unit_id	group_id	part_number	name	10003	1	1	2345125-21413	Товар 4XAWK	NULL	NULL	NULL	NULL	NULL
goods_dscr_id	unit_id	group_id	part_number	name														
10003	1	1	2345125-21413	Товар 4XAWK														
NULL	NULL	NULL	NULL	NULL														
<table border="1"><thead><tr><th>goods_id</th><th>goods_dscr_id</th><th>price_in</th><th>stock</th><th>reserve</th></tr></thead><tbody><tr><td>10001</td><td>10003</td><td>10000</td><td>0.000</td><td>0.000</td></tr><tr><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td></tr></tbody></table>		goods_id	goods_dscr_id	price_in	stock	reserve	10001	10003	10000	0.000	0.000	NULL	NULL	NULL	NULL	NULL		
goods_id	goods_dscr_id	price_in	stock	reserve														
10001	10003	10000	0.000	0.000														
NULL	NULL	NULL	NULL	NULL														

Помилки при оновленні даних, що визвав тригер

34	23:51:33	UPDATE ss_myisam2.goods_dscr SET unit_id = 21 WHERE goods_dscr_id = 10003	Error Code: 1644. Помилка: вказаний unit_id не існує в таблиці unit.
35	23:51:52	UPDATE ss_myisam2.goods_dscr SET group_id = 21 WHERE goods_dscr_id = 10003	Error Code: 1644. Помилка: вказаний group_id не існує в таблиці group.
36	23:52:58	UPDATE ss_myisam2.goods_dscr SET part_number = '4354-dfgsdg' WHERE goods_dscr_id = 1...	Error Code: 1644. Помилка: вказаний part_number вже існує в базі даних.
37	23:53:56	UPDATE ss_myisam2.goods_dscr SET name = 'Товар KRA34' WHERE goods_dscr_id = 10003	Error Code: 1644. Помилка: вказаний name вже існує в базі даних.
38	23:54:24	UPDATE ss_myisam2.goods_dscr SET name = null WHERE goods_dscr_id = 10003	Error Code: 1644. Помилка: всі обов'язкові поля мають бути заповнені.

Рис. 3.1.2.2 Робота тригера Before Update при оновленні даних в таблицю goods_dscr

3.1.2.3 Before Delete (CASCADE)

Для тригера BEFORE DELETE для таблиці goods_dscr (з типом MyISAM) додаємо перевірки на наявність зв'язків у таблицях goods, orders_goods та arrivals_goods. Якщо в таблицях orders_goods та arrivals_goods немає зв'язаних записів з таблицею goods, тригер дозволить видалення запису з таблиці goods_dscr і видалить усі пов'язані записи в таблиці goods. Інакше буде згенерована помилка.

```
DELIMITER //
CREATE TRIGGER goods_dscr_before_delete
BEFORE DELETE ON goods_dscr
FOR EACH ROW
BEGIN
    DECLARE goods_count INT DEFAULT 0;

    -- Перевірка, чи запчастина є активної
    IF OLD.is_active <> 0 THEN
```

```

        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: запчастина є активної, неможливо видалити.';

    END IF;

    -- Перевірка наявності пов'язаних записів у таблиці goods за goods_dscr_id
    SELECT COUNT(*) INTO goods_count
    FROM goods
    WHERE goods_dscr_id = OLD.goods_dscr_id;

    -- Якщо є пов'язані записи у таблиці goods, продовжуємо перевірку
    IF goods_count > 0 THEN

        -- Перевірка наявності пов'язаних записів у таблицях orders_goods та arrivals_goods за goods_id
        IF (SELECT COUNT(*)
            FROM orders_goods
            WHERE goods_id IN (
                SELECT goods_id FROM goods WHERE goods_dscr_id =
                OLD.goods_dscr_id)) > 0 THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка: неможливо видалити,
оскільки існують пов'язані записи в таблиці orders_goods.';

        END IF;

        IF (SELECT COUNT(*) FROM arrivals_goods WHERE goods_id IN
            (SELECT goods_id FROM goods WHERE goods_dscr_id = OLD.goods_dscr_id)) > 0
        THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка: неможливо видалити,
оскільки існують пов'язані записи в таблиці arrivals_goods.';

        END IF;

        -- Якщо зв'язків немає, видаляємо записи з таблиці goods, пов'язані з goods_dscr_id
        DELETE FROM goods WHERE goods_dscr_id = OLD.goods_dscr_id;
    END IF;

```

END//

DELIMITER ;

До видалення

```
1 •  DELETE FROM goods_dscr
2      WHERE goods_dscr_id = 10001;
3
4 •  SELECT goods_dscr_id, unit_id, group_id, part_number, name, is_active
5      FROM ss_myisam2.goods_dscr
6      WHERE goods_dscr_id IN (10000, 10001, 10002);
```

result Grid					
goods_dscr_id	unit_id	group_id	part_number	name	is_active
10000	1	4	HOOZM1ATCI	Товар KRA34	1
10001	1	1	2345125-21413	Товар 4XAWK	0
10002	1	1	4354-dfgsdg	Картридж 1615	1
NULL	NULL	NULL	NULL	NULL	NULL

```
3 •  SELECT *
4      FROM ss_myisam2.goods
5      WHERE goods_dscr_id IN (10000, 10001, 10002);
```

result Grid				
goods_id	goods_dscr_id	price_in	stock	reserve
10000	10000	4581	7.000	0.000
10001	10001	10000	0.000	0.000
NULL	NULL	NULL	NULL	NULL

```
3 •  SELECT order_id, goods_id
4      FROM ss_myisam2.orders_goods
5      WHERE goods_id IN (10000, 10001);
```

Result Grid		
order_id	goods_id	arrival_id
7971	10000	400
NULL	NULL	616

result Grid		
arrival_id	goods_id	quantity
400	10000	13.000
616	10000	17.000

Після видалення

54 00:49:51 DELETE FROM goods_dscr WHERE goods_dscr_id = 10001 1 row(s) affected

goods_dscr_id	unit_id	group_id	part_number	name	is_active
10000	1	4	HOOZM1ATCI	Товар KRA34	0
10002	1	1	4354-dfgsdg	Картридж 1615	1
NULL	NULL	NULL	NULL	NULL	NULL

goods_id	goods_dscr_id	price_in	stock	order_id	goods_id	arrival_id	goods_id
10000	10000	4581	7.000	7971	10000	400	10000
NULL	NULL	NULL	NULL	NULL	NULL	616	10000

Помилки при видаленні даних, що визвав тригер

```
60 00:56:20 DELETE FROM goods_dscr WHERE goods_dscr_id = 10002      Error Code: 1644. Помилка: запчастина є активної, неможливо видалити.
61 00:56:29 DELETE FROM goods_dscr WHERE goods_dscr_id = 10000      Error Code: 1644. Помилка: неможливо видалити, оскільки існують пов'язані записи в таблиці orders_goods.
```

Рис. 3.1.2.3 Робота тригера Before Delete при видаленні даних з таблиці goods_dscr

3.1.3. Таблиця goods

3.1.3.1 Before Insert

```
DELIMITER //
```

```
CREATE TRIGGER goods_before_insert
```

```
BEFORE INSERT ON goods
```

```
FOR EACH ROW
```

```
BEGIN
```

```
-- Перевірка, чи існує вказаний goods_dscr_id в таблиці goods_dscr
```

```
IF (SELECT COUNT(*) FROM goods_dscr gd WHERE gd.goods_dscr_id =  
NEW.goods_dscr_id) = 0 THEN
```

```
    SIGNAL SQLSTATE '45000'
```

```
    SET MESSAGE_TEXT = 'Помилка: вказаний goods_dscr_id не існує в таблиці  
goods_dscr.';
```

```
END IF;
```

```
-- Перевірка, чи всі обов'язкові поля заповнені
```

```
IF NEW.goods_dscr_id IS NULL OR
```

```
    NEW.price_in IS NULL THEN
```

```
    SIGNAL SQLSTATE '45000'
```

```
    SET MESSAGE_TEXT = 'Помилка: всі обов'язкові поля мають бути заповнені.';
```

```
END IF;
```

```
END//
```

```
DELIMITER ;
```

До вставки даних

```
1 •  INSERT INTO goods (goods_dscr_id, price_in)
2      VALUES(10000, 4600);
3
4 •  SELECT *
5      FROM ss_myisam2.goods
6      ORDER BY goods_id DESC;
```

Result Grid				
goods_id	goods_dscr_id	price_in	stock	reserve
10000	10000	4581	7.000	0.000
9999	9999	7212	4.000	0.000

Після вставки даних

5 15:35:09	INSERT INTO goods (goods_dscr_id, price_in) VALUES(10000, 4600)	1 row(s) affected															
<table border="1"><thead><tr><th>goods_id</th><th>goods_dscr_id</th><th>price_in</th><th>stock</th><th>reserve</th></tr></thead><tbody><tr><td>10002</td><td>10000</td><td>4600</td><td>0.000</td><td>0.000</td></tr><tr><td>10000</td><td>10000</td><td>4581</td><td>7.000</td><td>0.000</td></tr></tbody></table>			goods_id	goods_dscr_id	price_in	stock	reserve	10002	10000	4600	0.000	0.000	10000	10000	4581	7.000	0.000
goods_id	goods_dscr_id	price_in	stock	reserve													
10002	10000	4600	0.000	0.000													
10000	10000	4581	7.000	0.000													

Помилки при вставці даних, що визвав тригер

```
7 15:38:10 INSERT INTO goods (goods_dscr_id, price_in) VALUES(10100, 4600)          Error Code: 1644. Помилка: вказаній goods_dscr_id не існує в таблиці goods_dscr.
8 15:38:48 INSERT INTO goods (goods_dscr_id, price_in) VALUES(10000, null)           Error Code: 1644. Помилка: всі обов'язкові поля мають бути заповнені.
```

Рис. 3.1.3.1 Робота тригера Before Insert при вставці даних в таблицю goods

3.1.3.2 Before Update

```
DELIMITER //
CREATE TRIGGER goods_before_update
BEFORE UPDATE ON goods
FOR EACH ROW
BEGIN
```

```
-- Перевірка, чи існує новий goods_dscr_id в таблиці goods_dscr
```

```
IF NEW.goods_dscr_id IS NOT NULL AND
(SELECT COUNT(*)
 FROM goods_dscr gd
```

```

        WHERE gd.goods_dscr_id = NEW.goods_dscr_id) = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: вказаний goods_dscr_id не існує в таблиці
goods_dscr.';

    END IF;

    -- Перевірка, щоб обов'язкові поля не були NULL
    IF NEW.goods_dscr_id IS NULL OR
        NEW.price_in IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: всі обов'язкові поля мають бути заповнені.';

    END IF;

    -- Оновлення зв'язків для збереження цілісності даних, якщо goods_id змінюється
    IF NEW.goods_dscr_id <> OLD.goods_dscr_id THEN
        UPDATE orders_goods SET goods_id = NEW.goods_id WHERE goods_id =
        OLD.goods_id;
        UPDATE arrivals_goods SET goods_id = NEW.goods_id WHERE goods_id =
        OLD.goods_id;
    END IF;
END//
```

DELIMITER ;

До оновлення даних

```

5 •  UPDATE ss_myisam2.goods
6      SET goods_id = 10002
7      WHERE goods_id = 10000;
8
9 •  SELECT *
10     FROM ss_myisam2.goods
11     WHERE goods_dscr_id IN (10000, 10001, 10002);
```

result Grid | Filter Rows: | Edit:

goods_id	goods_dscr_id	price_in	stock	reserve
10000	10000	4581	7.000	0.000
10001	10000	4600	0.000	0.000
NULL	NULL	NULL	NULL	NULL

```

4 •   SELECT goods_dscr_id, unit_id, group_id, name
5     FROM ss_myisam2.goods_dscr
6   WHERE goods_dscr_id IN (10000, 10001, 10002);

```

goods_dscr_id	unit_id	group_id	name
10000	1	4	Товар KRA34
10002	1	1	Картридж 1615
NULL	NULL	NULL	NULL


```

3 •   SELECT order_id, goods_id
4     FROM ss_myisam2.orders_goods
5   WHERE goods_id IN (10000, 10001, 10002);

```

order_id	goods_id
7971	10000
NULL	NULL


```

3 •   SELECT *
4     FROM ss_myisam2.arrivals_goods
5   WHERE goods_id IN (10000, 10001, 10002);

```

arrival_id	goods_id	quantity
400	10000	13.000
616	10000	17.000
NULL	NULL	NULL

Після оновлення даних

26 18:01:54 UPDATE ss_myisam2.goods SET goods_id = 10002 WHERE goods_id = 10000 1 row(s) affected Rows matched: 1 Changed: 1

goods_id	goods_dscr_id	price_in
10002	10000	4581
10001	10000	4600
NULL	NULL	NULL

order_id	goods_id
7971	10002
NULL	NULL

arrival_id	goods_id	quantity
400	10002	13.000
616	10002	17.000
NULL	NULL	NULL

Помилки при оновленні даних, що визвав тригер

✓ 39 18:08:59 UPDATE ss_myisam2.goods SET goods_dscr_id = 10001 WHERE goods_id = 1... Error Code: 1644. Помилка: вказаний goods_dscr_id не існує в таблиці goods_dscr.
✗ 40 18:09:59 UPDATE ss_myisam2.goods SET price_in = null WHERE goods_id = 10001 Error Code: 1644. Помилка: всі обов'язкові поля повинні бути заповнені.

Рис. 3.1.3.2 Робота тригера Before Update при оновленні даних в таблицю goods

3.1.3.3 Before Delete (RESTRICT)

```

DELIMITER //
CREATE TRIGGER goods_before_delete
BEFORE DELETE ON goods
FOR EACH ROW
BEGIN
  -- Перевірка, чи stock <> 0 or reserve <> 0
  IF OLD.stock <> 0 OR OLD.reserve <> 0 THEN
    SIGNAL SQLSTATE '45000'
  END IF;
END //

```

```
SET MESSAGE_TEXT = 'Помилка: запчастина є на складі, або в резерві,  
неможливо видалити.';
```

```
END IF;
```

```
-- Перевірка наявності пов'язаних записів у таблиці orders
```

```
IF (SELECT COUNT(*)
```

```
FROM arrivals_goods WHERE goods_id = OLD.goods_id) > 0 THEN
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'Помилка: Неможливо видалити запис, оскільки існують  
пов'язані записи в таблиці arrivals_goods.';
```

```
END IF;
```

```
-- Перевірка наявності пов'язаних записів у таблиці arrivals
```

```
IF (SELECT COUNT(*)
```

```
FROM orders_goods WHERE goods_id = OLD.goods_id) > 0 THEN
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'Помилка: Неможливо видалити запис, оскільки існують  
пов'язані записи в таблиці orders_goods.';
```

```
END IF;
```

```
END//
```

```
DELIMITER ;
```

До видалення даних

```
9 •  DELETE FROM goods  
10 WHERE goods_id = 10001;  
11  
12 •  SELECT *  
13  FROM ss_myisam2.goods  
14 WHERE goods_dscr_id IN (10000, 10001, 10002);
```

Result Grid | Filter Rows: | Edit:

goods_id	goods_dscr_id	price_in	stock	reserve
10002	10000	4581	7.000	0.000
10001	10000	4600	0.000	0.000
10000	10000	4650	1.000	0.000
NULL	NULL	NULL	NULL	NULL

```

3 •  SELECT order_id, goods_id
4   FROM ss_myisam2.orders_goods
5   WHERE goods_id
6       IN (10000, 10001, 10002);

```

order_id	goods_id
7971	10002
NULL	NULL


```

3 •  SELECT *
4   FROM ss_myisam2.arrivals_goods
5   WHERE goods_id
6       IN (10000, 10001, 10002);

```

arrival_id	goods_id	quantity
400	10002	13.000
616	10002	17.000
NULL	NULL	NULL

Після видалення даних

46 18:32:07 DELETE FROM goods WHERE goods_id = 10001 1 row(s) affected

goods_id	goods_dscr_id	price_in	stock
10002	10000	4581	0.000
10000	10000	4650	1.000
NULL	NULL	NULL	NULL

Помилки при оновленні даних, що визвав тригер

52 18:36:06 DELETE FROM goods WHERE goods_id = 10002 Error Code: 1644. Помилка: Неможливо видалити запис, оскільки існують пов'язані записи в таблиці arrivals_goods.
 53 18:36:17 DELETE FROM goods WHERE goods_id = 10000 Error Code: 1644. Помилка: запчастина є на складі, або в резерві, неможливо видалити.

Рис. 3.1.3.3 Робота тригеру Before Delete при видалені даних з таблиці Goods

3.1.4. Таблиця orders

Я вирішив тут використати тригер **Insert (UPDATE)** виходячи з таких міркувань:

- якщо при вставці даних вказан співробітник для таблиці orders, це означає, що надійшла техніка в ремонт, та потрібно одразу робити вставку в три таблиці: orders, orders_received, orders_repair;
- якщо при вставці даних НЕ вказан співробітник, це означає, що йде виписка запчастин (можливо онлайн) і треба робити вставку одразу в дві таблиці: orders та orders_goods. Однак цей пункт у такому вигляді не має сенс використовувати у високонавантажених БД, тому що поки Замовник подумає

та наповнить замовлення запчастинами, декілька з них можуть бути не доступні на складі.

Тому другий пункт відкинемо і реалізуємо перший пункт в дещо зміненому вигляді:

- якщо при вставці даних в таблицю orders вказано додаткове тимчасове поле (яке ми щойно додали до таблиці orders), це означає, що надійшла техніка в ремонт, та потрібно одразу робити вставку в три таблиці: orders, orders_received, orders_repair.

orders	
order_id	INT
customer_id	INT
employee_id	TINYINT
delivery_id	TINYINT
date_invoice	DATE
date_delivery	DATE
delivery_number	VARCHAR(255)
date_returned	DATE
description	TEXT
read_only	TINYINT(1)
order_temp	JSON

Я розумію, що цей підхід для профі, можливо, не є гарним варіантом(особливо для MyISAM), але ми ж експериментуємо та надалі будемо досліджувати продуктивність отриманої системи, тому я вирішив зробити такий варіант тригера, а використовувати його чи ні ми вирішемо у наступних лабораторних роботах.

3.1.4.1a Before Insert (RESTRICT, тільки валідація даних)

```
DELIMITER //
CREATE TRIGGER orders_before_insert
BEFORE INSERT ON orders
FOR EACH ROW
BEGIN
    -- Встановлення початкових значень
    SET NEW.date_invoice = CURDATE();
    SET NEW.employee_id = NULL;
```

```

SET NEW.delivery_id = NULL;
SET NEW.date_delivery = NULL;
SET NEW.delivery_number = NULL;
SET NEW.date_returned = NULL;
-- Перевірка, чи існує вказаний customer_id в таблиці customers
IF NOT EXISTS
    (SELECT 1 FROM customers c WHERE c.customer_id = NEW.customer_id)
THEN    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: вказаний customer_id не існує в таблиці
customers.';

END IF;

-- employee_id = NULL або перевіряємо, чи існує вказаний employee_id в таблиці employee
IF NEW.employee_id IS NOT NULL THEN
    IF NOT EXISTS (SELECT 1 FROM employees e WHERE e.employee_id =
NEW.employee_id) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: вказаний employee_id не
існує в таблиці employees.';

    END IF;
END IF;
END//;
DELIMITER ;

```

До вставки даних

```

1 •  INSERT INTO orders (customer_id, employee_id)
2     VALUES (1, NULL), (2, 2);
3
4 •  SELECT *
5     FROM ss_myisam2.orders
6     ORDER BY order_id DESC;

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows: |

order_id	customer_id	employee_id	delivery_id	date_invoice	date_delivery	delivery_number	date_returned	description	read_only	order_temp
10016	2	2	NULL	2024-11-12	NULL	NULL	NULL	NULL	0	NULL
10015	1	NULL	NULL	2024-11-11	NULL	NULL	NULL	NULL	0	NULL
10014	1	NULL	NULL	2024-11-06	NULL	NULL	NULL	NULL	0	NULL
10012	1	NULL	NULL	2024-11-06	NULL	NULL	NULL	NULL	0	NULL
10011	6638	15	NULL	2024-09-11	NULL	NULL	NULL	NULL	0	NULL

Після вставки даних

order_id	customer_id	employee_id	delivery_id	date_invoice	date_delivery	delivery_number	date_returned	description	read_only	order_temp
10018	2	2	NULL	2024-11-21	NULL	NULL	NULL	NULL	0	NULL
10017	1	NULL	NULL	2024-11-21	NULL	NULL	NULL	NULL	0	NULL
10016	2	2	NULL	2024-11-12	NULL	NULL	NULL	NULL	0	NULL
10015	1	NULL	NULL	2024-11-11	NULL	NULL	NULL	NULL	0	NULL

Помилки при вставці даних, що визвав тригер

- ✖ 33 16:24:26 INSERT INTO orders (customer_id, employee_id) VALUES (22, 42) Error Code: 1644. Помилка: вказаний employee_id не існує в таблиці employees.
- ✖ 34 16:24:48 INSERT INTO orders (customer_id, employee_id) VALUES (22000, 2) Error Code: 1644. Помилка: вказаний customer_id не існує в таблиці customers.
- ✖ 35 16:25:04 INSERT INTO orders (customer_id, employee_id) VALUES (NULL, 2) Error Code: 1644. Помилка: вказаний customer_id не існує в таблиці customers.

Рис. 3.1.4.1a - Робота тригеру Before Insert при вставці даних в таблицю orders

3.1.4.1b AFTER Insert (CASCADE)

```

DELIMITER //

CREATE TRIGGER orders_after_insert
AFTER INSERT ON orders
FOR EACH ROW
BEGIN

    DECLARE rc_brand_id SMALLINT;
    DECLARE rc_employee_id TINYINT;
    DECLARE rc_model_name VARCHAR(20);
    DECLARE rc_sn VARCHAR(30);
    DECLARE rc_equipment TEXT;
    DECLARE rc_fault_description TEXT;
    DECLARE rp_employee_id TINYINT;

    -- Якщо orders_temp не NULL, зберігаємо дані в orders_received та orders_repair
    IF NEW.order_temp IS NOT NULL THEN

        SET rc_brand_id = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp,
        '$.brand_id'));
    END IF;

```

```

        SET rc_employee_id = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp,
        '$.rc_employee_id'));

        SET rc_model_name = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp,
        '$.model_name'));

        SET rc_sn = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp, '$.sn'));

        SET rc_equipment = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp,
        '$.equipment'));

        SET rc_fault_description =
        JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp, '$.fault_description'));

        SET rp_employee_id = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp,
        '$.rp_employee_id'));


        INSERT INTO orders_received (order_id, brand_id, employee_id,
model_name, sn, equipment, fault_description)
        VALUES (NEW.order_id, rc_brand_id, rc_employee_id,
rc_model_name, rc_sn, rc_equipment, rc_fault_description);

        INSERT INTO orders_repair (order_id, employee_id)
        VALUES (NEW.order_id, rp_employee_id);

    END IF;
END//;
DELIMITER ;

```

До вставки данных

```

1 •  SELECT *
2   FROM ss_myisam2.orders
3   order by order_id DESC;
4
5 •  INSERT INTO orders (customer_id, employee_id, orders_temp)
6   VALUES (1, 2, '{"brand_id":3, "rc_employee_id":5,"model_name":"1610", "sn":"SN123-126",
7   "equipment":"без кабеля живлення","fault_description": "торохтить під час увімкнення", "rp_employee_id":17}');

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows: |


| order_id | customer_id | employee_id | delivery_id | date_invoice | date_delivery | delivery_number | date_returned | description | read_only | order_temp |
|----------|-------------|-------------|-------------|--------------|---------------|-----------------|---------------|-------------|-----------|------------|
| 10018    | 2           | 2           | NULL        | 2024-11-21   | NULL          | NULL            | NULL          | NULL        | 0         | NULL       |
| 10017    | 1           | NULL        | NULL        | 2024-11-21   | NULL          | NULL            | NULL          | NULL        | 0         | NULL       |
| 10016    | 2           | 2           | NULL        | 2024-11-12   | NULL          | NULL            | NULL          | NULL        | 0         | NULL       |


```

```

3 •   SELECT order_id, brand_id, employee_id, model_name, equipment, fault_description
4   FROM ss_myisam2.orders_received
5   order by order_id DESC;

```

Result Grid					
order_id	brand_id	employee_id	model_name	equipment	fault_description
10012	1	1	2015	без шнура живлення	погана подача паперу, неякісний друк
10005	1	8	3115	Близько подвір'я мо...	Комунізм й командир кільце керівник інш...

```

2 •   SELECT order_id, employee_id,
3           date_ready, price_repair
4   FROM ss_myisam2.orders_repair
5   order by order_id DESC;

```

Result Grid			
order_id	employee_id	date_ready	price_repair
10005	8	2024-09-08	51000
10005	9	2024-09-06	30500
9999	12	2024-05-15	135343

Після вставки даних

✓ 9 18:25:15 INSERT INTO orders (customer_id, employee_id, order_temp) VALUES (1, 2, {"brand_id":3, "rc_employ... 1 row(s) affected

order_id	customer_id	employee_id	delivery_id	date_invoice	date_delivery	delivery_number	date_returned	description	read_only	order_temp
10020	1	2	NULL	2024-11-21	NULL	NULL	NULL	NULL	0	{"sn": "S...
10018	2	2	NULL	2024-11-21	NULL	NULL	NULL	NULL	0	NULL
10017	1	1	NULL	2024-11-21	NULL	NULL	NULL	NULL	0	NULL
10005	1	8	NULL	2024-09-08	NULL	NULL	NULL	NULL	1	NULL

order_id	brand_id	employee_id	model_name	equipment	fault_description
10020	3	5	1610	без кабелю живлення	торохтить під час увімкнення
10012	1	1	2015	без шнура живлення	погана подача паперу, неякісний друк
10005	1	8	3115	Близько подвір'я мо...	Комунізм й командир кільце керівник інш...

order_id	employee_id	date_ready	price_repair
10020	17	NULL	0
10005	8	2024-09-08	51000
10005	9	2024-09-06	30500
9999	12	2024-05-15	135343

Рис. 3.1.4.1b - Робота триггеру After Insert при вставці даних в таблиці orders, orders_received та orders_repair

3.1.4.2 Before Update

```

DELIMITER //
CREATE TRIGGER orders_before_update
BEFORE UPDATE ON orders
FOR EACH ROW

```

```
BEGIN
```

```
-- Перевірка, чи read_only = 1
IF OLD.read_only = 1 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: це замовлення тільки до читання, правити не
можна';
END IF;

-- Обнуляємо тимчасове поле
IF OLD.order_temp IS NOT NULL THEN
    SET NEW.order_temp = NULL;
END IF;

-- Перевірка, чи існує новий customer_id в таблиці customers
IF NOT EXISTS (SELECT 1 FROM customers c WHERE c.customer_id =
NEW.customer_id) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: вказаний customer_id не існує в таблиці
customers.!';
END IF;

-- employee_id = NULL або перевіряємо, чи існує вказаний employee_id в таблиці employee
IF NEW.employee_id IS NOT NULL THEN
    IF NOT EXISTS (SELECT 1 FROM employees e WHERE e.employee_id =
NEW.employee_id) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: вказаний employee_id не
існує в таблиці employees.!';
    END IF;
END IF;

-- Перевірка, чи існує новий delivery_id в таблиці delivery
IF NEW.delivery_id IS NOT NULL THEN
```

```
IF NOT EXISTS (SELECT 1 FROM delivery d WHERE d.delivery_id =
    NEW.delivery_id) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: вказаний delivery_id не існує в таблиці delivery.';
END IF;
END IF;
```

```
-- Перевірка дат
IF NEW.date_invoice <> OLD.date_invoice and NEW.date_invoice <> CURDATE()
THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: вказана date_invoice не можлива.';

END IF;

IF NEW.date_returned <> OLD.date_returned and NEW.date_returned <>
CURDATE() THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: вказана date_returned не можлива.';

END IF;

IF NEW.date_delivery <> OLD.date_delivery and DATEDIFF(NEW.date_delivery,
NEW.date_returned) > 5 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: вказана date_delivery не можлива.';

END IF;
```

```
-- Перевірка, щоб обов'язкові поля не були NULL
IF NEW.customer_id IS NULL OR
    NEW.date_invoice IS NULL THEN
    SIGNAL SQLSTATE '45000'
```

```
        SET MESSAGE_TEXT = 'Помилка: всі обов'язкові поля мають бути заповнені.';
```

```
    END IF;
```

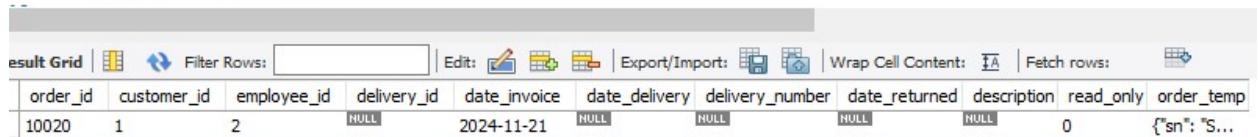
```
-- Оновлення зв'язків для збереження цілісності даних, якщо order_id змінюється
```

```
IF NEW.order_id <> OLD.order_id THEN
    UPDATE orders_goods SET order_id = NEW.order_id WHERE order_id =
    OLD.order_id;
    UPDATE orders_received SET order_id = NEW.order_id WHERE order_id =
    OLD.order_id;
    UPDATE orders_repair SET order_id = NEW.order_id WHERE order_id =
    OLD.order_id;
END IF;
END//
```

```
DELIMITER ;
```

До оновлення даних

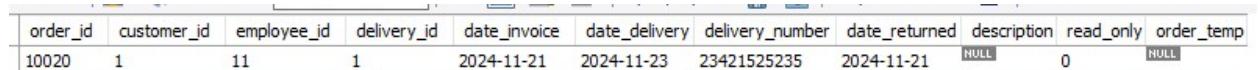
```
2 • UPDATE ss_myisam2.orders
3     SET date_returned = '2024-11-21', date_delivery = '2024-11-23',
4         delivery_id = 1, delivery_number = '23421525235', employee_id = 11
5     WHERE order_id = 10020;
6
7 • SELECT *
8     FROM ss_myisam2.orders
9     ORDER BY order_id DESC;
```



order_id	customer_id	employee_id	delivery_id	date_invoice	date_delivery	delivery_number	date_returned	description	read_only	order_temp
10020	1	2	NULL	2024-11-21	NULL	NULL	NULL	NULL	0	{"sn": "S...

Після оновлення даних

```
27 21:40:45 UPDATE ss_myisam2.orders SET date_returned = '2024-11-21', date_delivery = '2024... 1 row(s) affected Rows matched: 1 Changed: 1
```



order_id	customer_id	employee_id	delivery_id	date_invoice	date_delivery	delivery_number	date_returned	description	read_only	order_temp
10020	1	11	1	2024-11-21	2024-11-23	23421525235	2024-11-21	NULL	0	NULL

Помилки при оновленні даних, що визвав тригер

- ✖ 33 21:52:23 UPDATE ss_myisam2.orders SET date_returned = '2024-11-21', date_delivery = '2024... Error Code: 1644. Помилка: вказана date_delivery не можлива.
- ✖ 34 21:52:47 UPDATE ss_myisam2.orders SET date_returned = '2024-11-22', date_delivery = '2024... Error Code: 1644. Помилка: вказана date_returned не можлива.
- ✖ 35 21:53:02 UPDATE ss_myisam2.orders SET date_returned = '2024-11-21', date_delivery = '2024... Error Code: 1644. Помилка: вказаний delivery_id не існує в таблиці delivery.
- ✖ 36 21:53:17 UPDATE ss_myisam2.orders SET date_returned = '2024-11-21', date_delivery = '2024... Error Code: 1644. Помилка: вказаний employee_id не існує в таблиці employees.

38 21:56:29 UPDATE ss_myisam2.orders SET date_returned = '2024-11-21', date_delivery = '2024... Error Code: 1644. Помилка: вказаній customer_id не існує в таблиці customers.

Рис. 3.1.4.2a Робота тригеру Before Update при оновленні даних в таблицю orders

CASCADE - до оновлення даних

```

1 •  SET SQL_SAFE_UPDATES = 0;
2 •  UPDATE ss_myisam2.orders
3   SET order_id = 10019
4   WHERE order_id = 10020;
5
6 •  SELECT *
7   FROM ss_myisam2.orders
8   order by order_id DESC;

```

order_id	customer_id	employee_id	delivery_id	date_invoice	date_delivery	delivery_number	date_returned	description	read_only	order_temp
10020	1	11	1	2024-11-21	2024-11-23	23421525235	2024-11-21	NULL	0	NULL
10018	2	2	NULL	2024-11-21	NULL	NULL	NULL	NULL	0	NULL


```

2 •  SELECT order_id, employee_id,
3   date_ready, price_repair
4   FROM ss_myisam2.orders_repair
5   order by order_id DESC;

```

order_id	brand_id	employee_id
10020	3	5
10012	1	1

order_id	employee_id	date_ready	price_repair
10020	17	NULL	0
10005	8	2024-09-08	51000

Після оновлення даних

42 22:06:40 UPDATE ss_myisam2.orders SET order_id = 10019 WHERE order_id = 10020 1 row(s) affected Rows matched: 1 Changed: 1

order_id	customer_id	employee_id	delivery_id	date_invoice	date_delivery	delivery_number	date_returned	description	read_only	order_temp
10019	1	11	1	2024-11-21	2024-11-23	23421525235	2024-11-21	NULL	0	NULL
10018	2	2	NULL	2024-11-21	NULL	NULL	NULL	NULL	0	NULL

order_id	brand_id	employee_id
10019	3	5
10012	1	1

order_id	employee_id	date_ready	price_repair
10019	17	NULL	0
10005	9	2024-09-06	30500

Рис. 3.1.4.2b Робота тригеру Before Update при каскадному оновленні даних в таблиці orders, а також orders_received та orders_repair

3.1.4.3 Before Delete (CASCADE)

Для тригера BEFORE DELETE для таблиці orders (з типом MyISAM) додаємо перевірки на наявність зв'язків у таблицях orders, orders_received, orders_goods. Якщо в таблиці orders поле date_returned=NULL, а в таблиці orders_goods поле is_shipped =0, та немає зв'зку з таблицями orders_received та pay для даного order_id, тригер дозволить видалення запису з таблиці orders і видалить усі пов'язані записи в таблиці orders_goods. Інакше буде згенерована помилка.

```
DELIMITER //  
CREATE TRIGGER orders_before_delete  
BEFORE DELETE ON orders  
FOR EACH ROW  
BEGIN  
    -- Перевірка, чи замовлення відгружене  
    IF OLD.date_returned IS NOT NULL THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: замовлення відгружене, неможливо  
видалити.';  
    END IF;  
    -- Перевірка, чи є пов'язана запис у таблиці orders_received  
    IF EXISTS (SELECT 1 FROM orders_received WHERE order_id =  
OLD.order_id) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = "Помилка: неможливо видалити,  
оскільки існує пов'язана запис у таблиці orders_received.";  
    END IF;  
    -- Перевірка, чи є пов'язаний запис у таблиці pay  
    IF EXISTS (SELECT 1 FROM pay WHERE order_id = OLD.order_id) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = "Помилка: неможливо видалити,  
оскільки існує пов'язана запис у таблиці pay.";  
    END IF;
```

```

-- Якщо немає записів у таблиці orders_goods, де is_shipped = 1, то
-- видаляємо записи з таблиці orders_goods, пов'язані з orders по order_id
IF (SELECT COUNT(*) FROM orders_goods WHERE order_id =
OLD.order_id AND is_shipped = 1) = 0 THEN
    DELETE FROM orders_goods WHERE order_id = OLD.order_id;
ELSE
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: неможливо видалити,
оскільки існують відвантажені запчастини у таблиці orders_goods.';
END IF;
END//  

DELIMITER ;

```

До видалення

```

6 •  DELETE FROM orders
7   WHERE order_id = 10014;
8
9 •  SELECT *
10  FROM ss_myisam2.orders
11  order by order_id DESC;

```

order_id	customer_id	employee_id	delivery_id	date_invoice	date_delivery	delivery_number	date_returned	description	read_only	order_temp
10019	1	11	1	2024-11-21	2024-11-23	23421525235	2024-11-21	NULL	0	NULL
10018	2	2	NULL	2024-11-21	NULL	NULL	NULL	NULL	0	NULL
10017	1	NULL	NULL	2024-11-21	NULL	NULL	NULL	NULL	0	NULL
10016	2	2	NULL	2024-11-12	NULL	NULL	NULL	NULL	0	NULL
10015	1	NULL	NULL	2024-11-11	NULL	NULL	NULL	NULL	0	NULL
10014	1	NULL	NULL	2024-11-06	NULL	NULL	NULL	NULL	0	NULL

```

3 •  SELECT order_id, goods_id,
        quantity, is_shipped
      FROM ss_myisam2.orders_goods
      order by order_id DESC;

```

order_id	goods_id	quantity	is_shipped
10014	4	2.000	0
10014	2	1.000	0
10014	1	1.000	0
10011	1	2.000	0
10009	3	1.000	1
10009	2	1.000	0
10009	1	1.000	0

order_id	brand_id	employee_id
10019	3	5
10011	1	1
10005	1	8

Після видалення

59 23:10:03 DELETE FROM orders WHERE order_id = 10014 1 row(s) affected

order_id

10019
10018
10017
10016
10015
10012
10011

4 • SELECT order_id FROM orders_goods
5 order by order_id DESC;

result Grid | Filter Rows: | Ex

order_id

10011

Помилки при видаленні даних, що визвав тригер

- ✖ 69 23:28:44 DELETE FROM orders WHERE order_id = 10011 Error Code: 1644. Помилка: неможливо видалити, оскільки існує пов'язана запис у таблиці orders_received.
- ✖ 70 23:28:56 DELETE FROM orders WHERE order_id = 10019 Error Code: 1644. Помилка: замовлення відгружене, неможливо видалити.
- ✖ 71 23:29:02 DELETE FROM orders WHERE order_id = 10009 Error Code: 1644. Помилка: неможливо видалити, оскільки існують відвантажені запчастини у таблиці orders_goo...

Рис. 3.1.4.3 Робота тригеру Before Delete при видаленні даних з таблиці orders та orders_goods

3.1.5. Таблиця arrivals

3.1.5.1 Before Insert (RESTRICT)

```

DELIMITER //
CREATE TRIGGER arrivals_before_insert
BEFORE INSERT ON arrivals
FOR EACH ROW
BEGIN
    -- Перевірка, чи існує вказаний customer_id в таблиці customer
    IF NOT EXISTS (SELECT 1 FROM customers c WHERE c.customer_id =
    NEW.customer_id) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: вказаний customer_id не існує в таблиці
        customers.';
    END IF;
    - - перевіряємо, чи існує вказаний employee_id в таблиці employee
    IF NOT EXISTS
        (SELECT 1 FROM employees e WHERE e.employee_id = NEW.employee_id) THEN
            SIGNAL SQLSTATE '45000'

```

```

        SET MESSAGE_TEXT = 'Помилка: вказаний employee_id не існує в
таблиці employees.';

END IF;

```

```

-- Перевірка, чи total is null
IF NEW.total IS NULL THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка: не вказана загальна сума
накладної';
END IF;
END///
DELIMITER ;

```

До вставки даних

```

8 •  INSERT INTO arrivals (customer_id, employee_id, total)
9     VALUES (1, 2, 100000);
10

```

```

11 •  SELECT arrival_id, customer_id,
12      employee_id, date_arrival, total
13  FROM ss_myisam2.arrivals
14  ORDER BY arrival_id DESC;

```

arrival_id	customer_id	employee_id	date_arrival	total
1003	1	1	2024-11-06	260000
1002	3964	10	2024-11-05	315000

Після вставки даних

arrival_id	customer_id	employee_id	date_arrival	total
1004	1	2	NULL	100000
1003	1	1	2024-11-06	260000

Помилки при додаванні даних, що визвав тригер

- ✖ 9 21:08:59 INSERT INTO arrivals (customer_id, employee_id, total) VALUES (1, 54, 100000) Error Code: 1644. Помилка: вказаний employee_id не існує в таблиці employees.
- ✖ 10 21:09:16 INSERT INTO arrivals (customer_id, employee_id, total) VALUES (21000, 11, 100000) Error Code: 1644. Помилка: вказаний customer_id не існує в таблиці customers.
- ✖ 11 21:09:37 INSERT INTO arrivals (customer_id, employee_id, total) VALUES (21, 11, null) Error Code: 1644. Помилка: не вказана загальна сума накладної

Рис. 3.1.4.2 Робота тригеру Before Insert при додаванні даних в таблицю arrivals

3.1.5.2 Before Update

```
DELIMITER //  
CREATE TRIGGER arrivals_before_update  
BEFORE UPDATE ON arrivals  
FOR EACH ROW  
BEGIN  
    -- Перевірка, чи read_only = 1  
    IF OLD.read_only = 1 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: цей приход тільки до читання, правити не  
можна';  
    END IF;  
    -- Перевірка, чи існує новий customer_id в таблиці customers  
    IF NOT EXISTS (SELECT 1 FROM customers c WHERE c.customer_id =  
    NEW.customer_id) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: вказаний customer_id не існує в таблиці  
customers.';  
    END IF;  
    -- перевіряємо, чи існує вказаний employee_id в таблиці employee  
    IF NOT EXISTS (SELECT 1 FROM employees e WHERE e.employee_id =  
    NEW.employee_id) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: вказаний employee_id не існує в  
таблиці employees.';  
    END IF;  
    -- Перевірка дати  
    IF NEW.date_arrival <> OLD.date_arrival AND (  
        DATEDIFF(CURDATE(), NEW.date_arrival) > 15 OR  
        DATEDIFF(NEW.date_arrival, CURDATE()) > 0) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: вказана date_arrival не допустима.';  
    END IF;
```

```

IF NEW.date_arrival IS NULL AND NEW.read_only = 1 THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: цей приход ще не отриман, не можна ставити
тільки читання';
END IF;
-- Перевірка, щоб обов'язкові поля не були NULL
IF NEW.customer_id IS NULL OR
    NEW.employee_id IS NULL OR
    NEW.total IS NULL THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: всі обов'язкові поля мають бути заповнені。';
END IF;
-- Оновлення зв'язків для збереження цілісності даних, якщо arrival_id змінюється
IF NEW.arrival_id <> OLD.arrival_id THEN
    UPDATE arrivals_goods SET arrival_id = NEW.arrival_id WHERE arrival_id =
    OLD.arrival_id;
    UPDATE pay SET arrival_id = NEW.arrival_id WHERE arrival_id =
    OLD.arrival_id;
END IF;
END///
DELIMITER ;

```

До оновлення даних

```

15 •  SELECT arrival_id, customer_id, employee_id, date_arrival, total, read_only
16      FROM arrivals ORDER BY arrival_id DESC;
17
18 •  UPDATE arrivals SET date_arrival = '2024-11-24' WHERE arrival_id = 1004;

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co

arrival_id	customer_id	employee_id	date_arrival	total	read_only
1004	1	2	NULL	100000	0
1003	1	1	NULL	260000	0

Після оновлення даних

10 21:04:58 UPDATE arrivals SET date_arrival = '2024-11-24' WHERE arrival_id = 1004 1 row(s) affected Rows matched: 1 Changed: 1

	arrival_id	customer_id	employee_id	date_arrival	total	read_only
1004	1	2		2024-11-24	100000	0

Помилки при оновленні даних, що визвав тригер

15 21:13:55 UPDATE arrivals SET date_arrival = '2024-11-07' WHERE arrival_id = 1004	Error Code: 1644. Помилка: вказана date_arrival не допустима.
16 21:14:28 UPDATE arrivals SET date_arrival = '2024-11-25' WHERE arrival_id = 1004	Error Code: 1644. Помилка: вказана date_arrival не допустима.
17 21:14:53 UPDATE arrivals SET customer_id = 22000 WHERE arrival_id = 1004	Error Code: 1644. Помилка: вказаний customer_id не існує в таблиці customers.
18 21:15:03 UPDATE arrivals SET employee_id = 22000 WHERE arrival_id = 1004	Error Code: 1264. Out of range value for column 'employee_id' at row 1
19 21:15:31 UPDATE arrivals SET employee_id = 54 WHERE arrival_id = 1004	Error Code: 1644. Помилка: вказаний employee_id не існує в таблиці employees.
20 21:16:15 UPDATE arrivals SET employee_id = NULL WHERE arrival_id = 1004	Error Code: 1644. Помилка: вказаний employee_id не існує в таблиці employees.
21 21:16:52 UPDATE arrivals SET total = NULL WHERE arrival_id = 1004	Error Code: 1644. Помилка: всі обов'язкові поля мають бути заповнені.
49 22:04:39 UPDATE arrivals SET read_only = 1 WHERE arrival_id = 1003	Error Code: 1644. Помилка: цей приход ще не отриман, не можна ставити тільки читання

Рис. 3.1.5.2а Робота тригеру Before Update при оновленні даних в таблиці arrivals

До оновлення даних - CASCADE

```
15 •   SELECT arrival_id, customer_id, employee_id, date_arrival, total, read_only
16     FROM ss_myisam2.arrivals ORDER BY arrival_id DESC;
17
18 •   UPDATE arrivals SET arrival_id = 1005 WHERE arrival_id = 1004;
```

Result Grid					
arrival_id	customer_id	employee_id	date_arrival	total	read_only
1004	1	2	2024-11-24	100000	0
Result Grid					
3 • SELECT *	7 • SELECT pay_id, customer_id, employee_id,				
4 FROM arrivals_goods	8 arrival_id, date_pay, pay_sum				
5 WHERE arrival_id = 1004;	9 FROM pay				
Result Grid					
10 WHERE arrival_id = 1004;	11003	1	14	1004	2024-11-20 100000
	NULL	NULL	NULL	NULL	NULL

Після оновлення даних - CASCADE

```
30 21:32:57 UPDATE arrivals SET arrival_id = 1005 WHERE arrival_id = 1004    1 row(s) affected Rows matched: 1 Changed: 1
```

Result Grid					
arrival_id	customer_id	employee_id	date_arrival	total	read_only
1005	1	2	2024-11-24	100000	0
1003	1	1	2024-11-06	260000	0
Result Grid					
arrival_id	goods_id	quantity	pay_id	customer_id	employee_id
1005	2	20.000	11003	1	14
1005	12	10.000	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL
Result Grid					
arrival_id	date_pay	pay_sum	pay_id	customer_id	employee_id
1005	2024-11-20	100000	11003	1	14
NULL	NULL	NULL	NULL	NULL	NULL

Помилки при оновленні даних, що визвав тригер

```
37 21:35:51 UPDATE arrivals SET read_only = 1 WHERE arrival_id = 1005      1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
38 21:36:26 UPDATE arrivals SET arrival_id = 1004 WHERE arrival_id = 1005      Error Code: 1644. Помилка: цей приход тільки до читання, правити не можна
```

Рис. 3.1.5.2b Робота тригера Before Update при каскадному оновленні даних в таблиці arrivals, а також arrivals_goods, pay

3.1.5.3 Before Delete (CASCADE)

Для тригера BEFORE DELETE для таблиці arrivals (з типом MyISAM) додаємо перевірки на наявність зв'язків з таблицями pay, arrivals_goods. Якщо в таблиці arrivals поле date_arrival = NULL, а в таблиці pay немає зв'зку з таблицію arrivals для даного arrival_id, тригер дозволить видалення запису з таблиці arrivals і видалить усі пов'язані записи в таблиці arrivals_goods. Інакше буде згенерована помилка.

До видалення

```
15 •  SELECT arrival_id, customer_id, employee_id, date_arrival, total
16    FROM arrivals WHERE arrival_id = 1003;
17
18 •  DELETE FROM arrivals WHERE arrival_id = 1003;
```

arrival_id	customer_id	employee_id	date_arrival	total
1003	1	1	NULL	260000
NULL	NULL	NULL	NULL	NULL

```
52 22:13:54 SELECT * FROM arrivals_goods WHERE arrival_id = 1003 LIMIT 0, 10      5 row(s) returned
```

```
7 •  SELECT pay_id, customer_id, employee_id, arrival_id, date_pay, pay_sum
8    FROM pay
9   WHERE arrival_id = 1003;
```

pay_id	customer_id	employee_id	arrival_id	date_pay	pay_sum
NULL	NULL	NULL	NULL	NULL	NULL

Після видалення

```
62 22:41:10 DELETE FROM arrivals WHERE arrival_id = 1003      1 row(s) affected
```

```
63 22:41:34 SELECT arrival_id, customer_id, employee_id, date_arrival, total FROM arrivals WHERE arrival_id = 1003 LIMIT 0, 10      0 row(s) returned
```

```
66 22:43:14 | SELECT * FROM arrivals_goods WHERE arrival_id = 1003 LIMIT 0, 10 | 0 row(s) returned
```

Помилки при видаленні даних, що визвав тригер

```
✖ 72 22:49:58 | DELETE FROM arrivals WHERE arrival_id = 1000 | Error Code: 1644. Помилка: приход отриман, неможливо видалити.  
✖ 73 22:50:12 | DELETE FROM arrivals WHERE arrival_id = 1004 | Error Code: 1644. Помилка: неможливо видалити, оскільки існує пов'язана запис у таблиці pay.
```

Рис. 3.1.5.3 Робота тригеру Before Delete при видаленні даних з таблиці orders та orders_goods

3.1.6. Таблиця employees

3.1.6.1 Before Insert (RESTRICT)

```
DELIMITER //  
CREATE TRIGGER employees_before_insert  
BEFORE INSERT ON employees  
FOR EACH ROW  
BEGIN  
    -- Перевірка, чи існує вказаний position_id в таблиці position  
    IF NOT EXISTS (SELECT 1 FROM position p WHERE p.position_id =  
    NEW.position_id) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: вказаний position_id не існує в таблиці position.';  
    END IF;  
  
    -- Перевірка, чи існує вказаний status_id в таблиці status  
    IF NOT EXISTS (SELECT 1 FROM status WHERE status.status_id = NEW.status_id)  
    THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: вказаний status_id не існує в таблиці status.';  
    END IF;  
  
    -- Перевірка, чи телефон є унікальним  
    IF EXISTS (SELECT 1 FROM employees WHERE phone = NEW.phone) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка: телефон вже існує в базі даних.';
```

END IF;

-- Перевірка, чи всі обов'язкові поля заповнені

IF NEW.position_id IS NULL OR
NEW.status_id IS NULL OR
NEW.last_name IS NULL OR
NEW.first_name IS NULL OR
NEW.middle_name IS NULL OR
NEW.code IS NULL OR
NEW.birthday IS NULL OR
NEW.phone IS NULL OR
NEW.password IS NULL THEN
SIGNAL SQLSTATE '45000'

SET MESSAGE_TEXT = 'Помилка: всі обов'язкові поля мають бути заповнені.';

END IF;

END//

DELIMITER ;

До вставки даних

```
1 •  SELECT employee_id, status_id, position_id, first_name, middle_name, last_name, code, birthday, phone, password  
2   FROM ss_myisam2.employees ORDER BY employee_id DESC;  
3  
4 •  INSERT INTO employees(status_id, position_id, first_name, middle_name, last_name, code, birthday, phone, password)  
5   VALUES(5, 11, "Олег", "Іванович", "Квачко", "0123498567", "1997-09-29", "+380669513267", "234sdfdf345gxdg");
```

result Grid									
employee_id	status_id	position_id	first_name	middle_name	last_name	code	birthday	phone	password
21	5	6	Петро	Іванович	Федорків	0000001234	2002-12-12	+38066 123-4567	asdanhlskj634lk...
20	6	3	Петро	Володимирович	Шаповал	4469264219	1986-08-02	+380 16 431-10-30	\$pbkdf2-sha256...

Після вставки даних

employee_id	status_id	position_id	first_name	middle_name	last_name	code	birthday	phone	password
22	5	6	Олег	Іванович	Квачко	0123498567	1997-09-29	+380669513267	234sdfdf345gxdg

Помилки при вставці даних, що визвав тригер

✖ 84 23:48:14	INSERT INTO employees(status_id, position_id, first_n...	Error Code: 1644. Помилка: вказаний status_id не існує в таблиці status.
✖ 85 23:48:23	INSERT INTO employees(status_id, position_id, first_n...	Error Code: 1644. Помилка: вказаний position_id не існує в таблиці position.
✖ 86 23:50:37	INSERT INTO employees(status_id, position_id, first_n...	Error Code: 1644. Помилка: телефон вже існує в базі даних.
✖ 87 23:51:06	INSERT INTO employees(status_id, position_id, first_n...	Error Code: 1644. Помилка: всі обов'язкові поля мають бути заповнені.
✖ 88 23:51:33	INSERT INTO employees(status_id, position_id, first_n...	Error Code: 1644. Помилка: всі обов'язкові поля мають бути заповнені.
✖ 89 23:51:50	INSERT INTO employees(status_id, position_id, first_n...	Error Code: 1644. Помилка: всі обов'язкові поля мають бути заповнені.
✖ 90 23:52:04	INSERT INTO employees(status_id, position_id, first_n...	Error Code: 1644. Помилка: всі обов'язкові поля мають бути заповнені.
✖ 91 23:52:17	INSERT INTO employees(status_id, position_id, first_n...	Error Code: 1644. Помилка: всі обов'язкові поля мають бути заповнені.
✖ 92 23:52:33	INSERT INTO employees(status_id, position_id, first_n...	Error Code: 1644. Помилка: всі обов'язкові поля мають бути заповнені.

Рис. 3.1.6.1 - Робота тригера Before Insert при вставці даних в таблицю

employees

3.1.6.2a Before Update(CASCADE)

```
DELIMITER //  
CREATE TRIGGER employees_before_update  
BEFORE UPDATE ON employees  
FOR EACH ROW  
BEGIN  
    -- Оновлення поля update_time при кожному оновленні  
    SET NEW.update_time = CURRENT_TIMESTAMP;  
  
    -- Перевірка, чи існує новий position_id в таблиці position  
    IF NEW.position_id IS NOT NULL AND NOT EXISTS  
        (SELECT 1 FROM position p WHERE p.position_id = NEW.position_id) THEN  
            SIGNAL SQLSTATE '45000'  
            SET MESSAGE_TEXT = 'Помилка: вказаний position_id не існує в таблиці position.';  
    END IF;  
  
    -- Перевірка, чи існує новий status_id в таблиці status  
    IF NEW.status_id IS NOT NULL AND NOT EXISTS  
        (SELECT 1 FROM status WHERE status.status_id = NEW.status_id) THEN  
            SIGNAL SQLSTATE '45000'  
            SET MESSAGE_TEXT = 'Помилка: вказаний status_id не існує в таблиці status.';  
    END IF;  
  
    -- Перевірка, чи телефон є унікальним  
    IF NEW.phone <> OLD.phone AND EXISTS  
        (SELECT 1 FROM employees WHERE phone = NEW.phone) THEN  
            SIGNAL SQLSTATE '45000'  
            SET MESSAGE_TEXT = 'Помилка: цей телефон вже існує в базі даних.';  
    END IF;  
  
    -- Перевірка, щоб обов'язкові поля не були NULL  
    IF NEW.position_id IS NULL OR  
        NEW.status_id IS NULL OR
```

```

NEW.last_name IS NULL OR
NEW.first_name IS NULL OR
NEW.middle_name IS NULL OR
NEW.code IS NULL OR
NEW.birthday IS NULL OR
NEW.phone IS NULL OR
NEW.password IS NULL THEN
  SIGNAL SQLSTATE '45000'
  SET MESSAGE_TEXT = 'Помилка: всі обов'язкові поля мають бути заповнені.';

END IF;

-- Оновлення зв'язків для збереження цілісності даних, якщо employee_id змінюється
-- Ця логіка перенесена в тригер After Update
END//;
DELIMITER ;

```

До оновлення даних

```

1 •  SELECT employee_id, status_id, position_id, first_name, middle_name, last_name, code, birthday, phone, update_time
2   FROM ss_myisam2.employees ORDER BY employee_id DESC;
3
4 •  UPDATE employees
5   SET status_id = 4, position_id = 5, phone = '+38066 9513267' WHERE employee_id = 22;

```

result Grid									
employee_id	status_id	position_id	first_name	middle_name	last_name	code	birthday	phone	update_time
22	5	6	Олег	Іванович	Квачко	0123498567	1997-09-29	+380669513267	2024-11-23 23:47:04

Після оновлення даних

3 12:12:02	UPDATE employees SET status_id = 4, position_id = 5, phone = '+38066 9513267' WHERE employee_id = 22	1 row(s) affected Rows matched: 1 Changed: 1							
<hr/>									
employee_id	status_id	position_id	first_name	middle_name	last_name	code	birthday	phone	update_time
22	4	5	Олег	Іванович	Квачко	0123498567	1997-09-29	+380669513267	2024-11-24 12:12:02

Помилки при оновленні даних, що визвав тригер

16 17:42:05	UPDATE employees SET phone = '+38066 9513267' WHERE employee_id = 22	Error Code: 1644. Помилка: цей телефон вже існує в базі даних.
17 17:42:31	UPDATE employees SET status_id = 22 WHERE employee_id = 22	Error Code: 1644. Помилка: вказаний status_id не існує в таблиці status.
18 17:42:51	UPDATE employees SET position_id = 22 WHERE employee_id = 22	Error Code: 1644. Помилка: вказаний position_id не існує в таблиці position.

Рис. 3.1.6.2а Робота тригеру Before Update при оновленні даних в таблиці employees

```
✖ 39 18:18:05 UPDATE employees SET employee_id = 23 WHERE employee_id = 13 Error Code: 1644. Помилка: вказаній employee_id не існує в таблиці employees.
```

Рис. 3.1.6.2b Робота тригеру Before Update при каскадному оновленні даних в таблиці employees

Я стикнувся з ситуацією, коли при зміні employee_id дані в таблиці employees ще не зафіковані та тригери на інших таблицях не бачать цього нового employee_id, отримую помилку.

Щоб вирішити це питання перенесемо логіку каскадного оновлення в тригер After Update.

3.1.6.2b After Update(CASCADE)

```
DELIMITER //
```

```
DROP TRIGGER employees_after_update;
```

```
CREATE TRIGGER employees_after_update
BEFORE UPDATE ON employees
FOR EACH ROW
BEGIN
```

```
-- Оновлення зв'язків для збереження цілісності даних, якщо employee_id змінюється
```

```
IF NEW.employee_id <> OLD.employee_id THEN
    UPDATE orders SET employee_id = NEW.employee_id WHERE employee_id =
    OLD.employee_id;
    UPDATE arrivals SET employee_id = NEW.employee_id WHERE employee_id =
    OLD.employee_id;
```

```

UPDATE pay SET employee_id = NEW.employee_id WHERE employee_id =
OLD.employee_id;

UPDATE orders_received SET employee_id = NEW.employee_id WHERE
employee_id = OLD.employee_id;

UPDATE orders_repair SET employee_id = NEW.employee_id WHERE
employee_id = OLD.employee_id;

UPDATE salary SET employee_id = NEW.employee_id WHERE employee_id =
OLD.employee_id;

END IF;

END//
```

DELIMITER ;

Створивши тригер After Update, з'являється інше питання. Рознесення логіки каскадного оновлення на два тригери веде до порушення цілісності даних в тому случаї, якщо другий тригер не спрацьовує повністю, а перший вже внес правки в таблицю. Приклад наведен на рис. 3.1.6.2c

До оновлення даних - CASCADE

```

1 •  SELECT employee_id, status_id, position_id,
2           first_name, middle_name, last_name
3     FROM ss_myisam2.employees WHERE employee_id = 13;
4
5 •  UPDATE employees
6     SET employee_id = 23 WHERE employee_id = 13;
```

employee_id	status_id	position_id	first_name	middle_name	last_name
13	1	3	Наталія	Ааронівна	Канівець

```

14 •  SELECT * FROM orders
15 WHERE employee_id = 13;
```

order_id	customer_id	employee_id
49	3312	13
107	163	13

```

17 •  SELECT * FROM orders_received
18 WHERE employee_id = 13;
```

order_id	brand_id	employee_id	model_name
49	3	13	1931
107	1	13	5488

```

20 •  SELECT * FROM orders_repair
21 WHERE employee_id = 13;
```

order_id	employee_id	date_ready	price_
49	13	2024-07-14	81669
107	13	2024-03-30	66184

1 • <code>SELECT * FROM salary</code>	3 • <code>SELECT * FROM arrivals</code>	6 • <code>SELECT * FROM pay</code>
2 <code>WHERE employee_id = 13;</code>	4 <code>WHERE employee_id = 13;</code>	7 <code>WHERE employee_id = 13;</code>

Після оновлення даних - CASCADE

✖ 63 19:25:57 UPDATE employees SET employee_id = 23 WHERE employee_id = 13 Error Code: 1644. Помилка: це замовлення тільки до читання, правити не можна

<code>employee_id</code>	<code>status_id</code>	<code>position_id</code>	<code>first_name</code>	<code>middle_name</code>	<code>last_name</code>
23	1	3	Наталія	Ааронівна	Канівець

<code>order_id</code>	<code>customer_id</code>	<code>employee_id</code>	<code>delivery_id</code>	<code>date_invoice</code>	<code>date_delivery</code>	<code>delivery_number</code>	<code>date_returned</code>	<code>description</code>	<code>read_only</code>
764	807	23	3	2024-02-01	2024-02-10	988922842	2024-02-09	Інструк...	0
773	9752	23	3	2024-04-15	2024-05-11	608102714	2024-05-10	Збільшу...	0
774	8141	23	2	2024-03-15	2024-04-10	261839167	2024-04-09	Терапія ...	0
864	787	13	3	2024-01-08	2024-01-30	822539999	2024-01-29	Несподі...	1
892	5941	13	NULL	2024-05-20	NULL	NULL	2024-06-11	Міра пл...	0
923	9504	13	2	2024-06-23	2024-07-22	819213907	2024-07-21	Ефект з...	0
956	5336	13	2	2024-02-10	2024-02-17	945861107	2024-02-16	Бак дея...	0

22 • <code>SELECT * FROM orders_received</code>	25 • <code>SELECT * FROM orders_repair</code>
23 <code>WHERE employee_id in (13, 23);</code>	26 <code>WHERE employee_id in (13, 23);</code>

<code>Result Grid</code>	<code>Filter Rows:</code>		
<code>order_id</code>	<code>brand_id</code>	<code>employee_id</code>	<code>model_name</code>

49	3	13	1931
107	1	13	5488

49	13	2024-07-14	81669
107	13	2024-03-30	66184

<code>id</code>	<code>year</code>	<code>month</code>	<code>sum</code>	<code>arrival_id</code>	<code>custom</code>	<code>employee</code>	<code>pay_id</code>	<code>custo</code>	<code>employee</code>	<code>order_id</code>
13	2024	1	27587	23	2287	13	36	5103	13	39
13	2024	2	27587	34	2260	13	37	3082	13	40
13	2024	3	27587	46	1726	13	90	4244	13	93
13	2024	4	27587	80	9316	13	132	7148	13	135
13	2024	5	27587	83	1920	13	154	1372	13	157

Рис. 3.1.6.2c Робота тригера After Update при каскадному оновленні даних в таблиці employees

Коментарі будуть такі:

- використання двох послідовних тригерів Before та After при каскадному оновленні даних в таблицях MyISAM може привести до втрати цілісності даних, якщо перший тригер дозволить оновлення, а других зіткнеться з

неможливостю оновлення з будь-якої причини (в прикладі це «read only» запис),

- Є вероятність, що в результаті оновлення частина записів оновиться, а частина – ні,
- щоб забезпечити цілісність даних при оновленні в таблицях MyISAM не можна використовувати два послідовних тригера Before та After,
- щоб вирішити конфлікт треба використовувати більш складну логіку роботи, наприклад, тимчасове проміжне поле, або навіть таблиця. Мені здається, що це занадто ускладнить логіку роботи БД і не варте того. Каскадне оновлення первинного ключа, це вкрай рідко потрібна операція,
- тому, Я вирішив відмовитись від зв'язку Update Cascade та використати Update Ristrict, тригер After Update видалив, тригер Before Update змінив.

3.1.6.2d Before Update(RESTRICT). Оновлений тригер
DELIMITER //

```
DROP TRIGGER employees_before_update;
```

```
CREATE TRIGGER employees_before_update  
BEFORE UPDATE ON employees
```

```
FOR EACH ROW
```

```
BEGIN
```

```
-- Оновлення поля update_time при кожному оновленні
```

```
SET NEW.update_time = CURRENT_TIMESTAMP;
```

```
-- Перевірка, чи існує новий position_id в таблиці position
```

```
IF NEW.position_id IS NOT NULL AND NOT EXISTS
```

```
(SELECT 1 FROM position p WHERE p.position_id = NEW.position_id) THEN  
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'Помилка, employees: вказаний position_id не існує в  
таблиці position.';
```

```
END IF;
```

```
-- Перевірка, чи існує новий status_id в таблиці status
```

```
IF NEW.status_id IS NOT NULL AND NOT EXISTS
```

```

(SELECT 1 FROM status WHERE status.status_id = NEW.status_id) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, employees: вказаний status_id не існує в таблиці
status.';

END IF;

-- Перевірка, чи телефон є унікальним, якщо він змінюється
IF NEW.phone <> OLD.phone AND EXISTS
    (SELECT 1 FROM employees WHERE phone = NEW.phone) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, employees: цей телефон вже існує в базі даних.';

END IF;

-- Перевірка, щоб обов'язкові поля не були NULL
IF NEW.position_id IS NULL OR
    NEW.status_id IS NULL OR
    NEW.last_name IS NULL OR
    NEW.first_name IS NULL OR
    NEW.middle_name IS NULL OR
    NEW.code IS NULL OR
    NEW.birthday IS NULL OR
    NEW.phone IS NULL OR
    NEW.password IS NULL THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, employees: всі обов'язкові поля мають бути
заповнені.';

END IF;

-- Заборона оновлення зв'язків, якщо employee_id змінюється - Update RESTRICT
IF NEW.employee_id <> OLD.employee_id THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, employees: зміна employee_id не дозволена.';

END IF;

END//
```

```
DELIMITER ;
```

Після оновлення даних

```
✖ 94 21:26:55 UPDATE employees SET employee_id = 33 WHERE employee_id = 13 Error Code: 1644. Помилка, employees: зміна employee_id не дозволена.
```

Рис. 3.1.6.2d Робота тригера Before Update при оновленні employee_id в таблиці employees

3.1.6.3 Before Delete (RESTRICT).

```
DELIMITER //  
CREATE TRIGGER employees_before_delete  
BEFORE DELETE ON employees  
FOR EACH ROW  
BEGIN  
    -- Перевірка, чи робітник є активним  
    IF OLD.is_active <> 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка, employees: employee активний, неможливо  
видалити.';  
    END IF;  
    -- Перевірка наявності пов'язаних записів у таблиці orders  
    IF (SELECT COUNT(*) FROM orders WHERE employee_id = OLD.employee_id) >  
0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка, employees: Неможливо видалити запис,  
оскільки існують пов'язані записи в таблиці orders.';  
    END IF;  
    -- Перевірка наявності пов'язаних записів у таблиці arrivals  
    IF (SELECT COUNT(*) FROM arrivals WHERE employee_id = OLD.employee_id) >  
0 THEN  
        SIGNAL SQLSTATE '45000'
```

```

        SET MESSAGE_TEXT = 'Помилка, employees: Неможливо видалити запис,
оскільки існують пов'язані записи в таблиці arrivals.';

    END IF;

    -- Перевірка наявності пов'язаних записів у таблиці pay
    IF (SELECT COUNT(*) FROM pay WHERE employee_id = OLD.employee_id) > 0
THEN
        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Помилка, employees: Неможливо видалити запис,
оскільки існують пов'язані записи в таблиці pay.';

    END IF;

    -- Перевірка наявності пов'язаних записів у таблиці orders_received
    IF (SELECT COUNT(*) FROM orders_received WHERE employee_id =
OLD.employee_id) > 0 THEN
        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Помилка, employees: Неможливо видалити запис,
оскільки існують пов'язані записи в таблиці orders_received.';

    END IF;

    -- Перевірка наявності пов'язаних записів у таблиці orders_repair
    IF (SELECT COUNT(*) FROM orders_repair WHERE employee_id =
OLD.employee_id) > 0 THEN
        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Помилка, employees: Неможливо видалити запис,
оскільки існують пов'язані записи в таблиці orders_repair.';

    END IF;

    -- Перевірка наявності пов'язаних записів у таблиці salary
    IF (SELECT COUNT(*) FROM salary WHERE employee_id = OLD.employee_id) > 0
THEN
        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Помилка, employees: Неможливо видалити запис,
оскільки існують пов'язані записи в таблиці salary.';

    END IF;

END//  

DELIMITER ;

```

До видалення даних

```
1 •  DELETE FROM employees
2      WHERE employee_id = 22;
3
4 •  SELECT employee_id, status_id, position_id,
5          first_name, middle_name, last_name, is_active
6      FROM ss_myisam2.employees WHERE employee_id in (20, 21, 22);
```

Result Grid						
employee_id	status_id	position_id	first_name	middle_name	last_name	is_active
20	6	3	Петро	Володимирович	Шаповал	1
21	5	6	Петро	Іванович	Федорків	1
22	5	6	Олег	Іванович	Квачко	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
21 •  SELECT * FROM orders
22      WHERE employee_id in (20, 21, 22);
```

Result Grid				
order_id	customer_id	employee_id	delivery_id	date_invoice
1	441	20	3	2024-08-22
46	8010	20	NULL	2024-08-07
50	7989	20	2	2024-09-08

Після видалення даних

104	22:15:58	DELETE FROM employees WHERE employee_id = 22	1 row(s) affected
<hr/>			
employee_id	status_id	position_id	first_name
20	6	3	Петро
21	5	6	Петро
NULL	NULL	NULL	NULL
middle_name	last_name	is_active	
Володимирович	Шаповал	0	
Іванович	Федорків	1	
NULL	NULL	NULL	NULL

Помилки при оновлені даних, що визвав тригер

```
✖ 109 22:18:14 DELETE FROM employees WHERE employee_id = 20      Error Code: 1644. Помилка, employees: Неможливо видалити запис, оскільки існують пов'язані записи в таблиці orders.
✖ 110 22:18:36 DELETE FROM employees WHERE employee_id = 21      Error Code: 1644. Помилка, employees: employee активний, неможливо видалити.
```

Рис. 3.1.6.3 Робота тригеру Before Delete при видалені даних з таблиці employees

Після створення тригерів для таблиці employees сам собою напрошується такий висновок:

- Що з тригерами таблиці customers, яка дуже схожа з таблицею employees? Стосовно тригера **Update CASCADE**, його треба

переробити на **Update RESTRICT**, тому що він викличе аналогічну помилку при оновленні даних, як і тригер до таблиці employees.

2. Навіщо нам взагалі використовувати більш затратний і дуже проблемний (особливо для таблиць MyISAM) **Update CASCADE** тригер? Тим більше, що цей тригер відповідає за зміну значення первинного ключа, це само по собі дуже рідкісна і небезпечна операція.

Виходячи з вищевикладеного вирішив надалі використовувати **Update RESTRICT** тригер (спробували, використали в деяких тригерах, де це відносно надійно працює, розібрались в нюансах, на тому і завершемо с цим).

3.1.7. Таблиця pay

3.1.7.1 Before Insert (RESTRICT)

```
DELIMITER //  
CREATE TRIGGER pay_before_insert  
BEFORE INSERT ON pay  
FOR EACH ROW  
BEGIN  
    -- Перевірка, чи існує вказаний customer_id в таблиці customer  
    IF NOT EXISTS (SELECT 1 FROM customers c WHERE c.customer_id =  
    NEW.customer_id) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка, pay: вказаний customer_id не існує в таблиці  
        customers.';  
    END IF;  
  
    -- перевіряємо, чи існує вказаний employee_id в таблиці employee  
    IF NOT EXISTS (SELECT 1 FROM employees e WHERE e.employee_id =  
    NEW.employee_id) THEN  
        SIGNAL SQLSTATE '45000'
```

```

        SET MESSAGE_TEXT = 'Помилка, роз: вказаний employee_id не існує
в таблиці employees.';

    END IF;

-- Перевірка, arrival_id, order_id
    IF (NEW.arrival_id IS NULL AND NEW.order_id IS NULL) OR
        (NEW.arrival_id IS NOT NULL AND NEW.order_id IS NOT NULL)
THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, роз: arrival_id або order_id має
бути заповнене';
    END IF;

-- перевіряємо, чи існує вказаний order_id для даного customer_id в таблиці orders
    IF NEW.order_id IS NOT NULL THEN
        IF NOT EXISTS (SELECT 1 FROM orders o WHERE o.customer_id =
NEW.customer_id and o.order_id = NEW.order_id) THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка, роз: не існує вказаний
order_id для даного customer_id в таблиці orders.';

        END IF;
    END IF;

-- перевіряємо, чи існує вказаний arrival_id для даного customer_id в таблиці arrivals
    IF NEW.arrival_id IS NOT NULL THEN
        IF NOT EXISTS (SELECT 1 FROM arrivals a WHERE a.customer_id =
NEW.customer_id and a.arrival_id = NEW.arrival_id) THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка, роз: не існує вказаний
order_id для даного customer_id в таблиці arrivals.';

        END IF;
    END IF;

-- Перевірка, обов'язкових полів
    IF NEW.date_pay IS NULL AND

```

```

        NEW.pay_sum IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Помилка, ру: треба заповнити обов'язкові
поля";
    END IF;
END//;
DELIMITER ;

```

До вставки даних

```

2 •  INSERT INTO pay
3      (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum)
4      VALUES(5392, 12, 1000, null, "2024-09-29", 50000),
5          (1, 12, null, 10019 , "2024-09-29", 50000);
6
7 •  SELECT * FROM pay ORDER BY pay_id DESC;

```

result Grid							
pay_id	customer_id	employee_id	arrival_id	order_id	date_pay	pay_sum	pay_form
11005	1	12	NULL	10019	2024-09-29	50000	оплата на р.р.
11004	4	12	1000	NULL	2024-09-29	50000	оплата на р.р.
11003	1	14	1004	NULL	2024-11-20	100000	оплата на р.р.

Після вставки даних

```

5 14:07:55 INSERT INTO pay (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum) VALUE... 1 row(s) affected
6 14:08:33 INSERT INTO pay (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum) VALUE... 2 row(s) affected Records: 2

```

```

2 •  INSERT INTO pay
3      (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum)
4      VALUES(5392, 12, 1000, null, "2024-09-29", 50000),
5          (5392, 12, 1000, null, "2024-09-29", 50000),
6          (1, 12, null, 10019 , "2024-09-29", 50000);
7
8 •  SELECT * FROM pay ORDER BY pay_id DESC;

```

result Grid							
pay_id	customer_id	employee_id	arrival_id	order_id	date_pay	pay_sum	pay_form
11008	1	12	NULL	10019	2024-09-29	50000	оплата на р.р.
11007	5392	12	1000	NULL	2024-09-29	50000	оплата на р.р.
11006	5392	12	1000	NULL	2024-09-29	50000	оплата на р.р.
11005	1	12	NULL	10019	2024-09-29	50000	оплата на р.р.
11004	4	12	1000	NULL	2024-09-29	50000	оплата на р.р.

Помилки при додаванні даних, що визвав тригер

8 14:38:39	INSERT INTO pay (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum) VALUES(5391, 12, 1000, null, "2024-09-29", 50000), (5392, 12, 1000, null, "2024-09-29", 50000), (...)	Error Code: 1644. Помилка, pay: не існує вказанний order_id для даного customer_id в таблиці arrivals.
9 14:39:06	INSERT INTO pay (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum) VALUES(5392, 32, 1000, null, "2024-09-29", 50000), (5392, 12, 1000, null, "2024-09-29", 50000), (...)	Error Code: 1644. Помилка, pay: вказаній employee_id не існує в таблиці employees.
10 14:39:42	INSERT INTO pay (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum) VALUES(53920, 12, 1000, null, "2024-09-29", 50000)	Error Code: 1644. Помилка, pay: вказаній customer_id не існує в таблиці customers.
11 14:40:13	INSERT INTO pay (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum) VALUES(53920, 12, 1000, null, "2024-09-29", 50000), (2, 12, null, 10019, "2024-09-29", 50000)	Error Code: 1644. Помилка, pay: вказаній customer_id не існує в таблиці customers.
12 14:40:27	INSERT INTO pay (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum) VALUES(5392, 12, 1000, null, "2024-09-29", 50000), (2, 12, null, 10019, "2024-09-29", 50000)	Error Code: 1644. Помилка, pay: не існує вказанний order_id для даного customer_id в таблиці orders.
13 14:41:04	INSERT INTO pay (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum) VALUES(5392, 12, 1000, 1, "2024-09-29", 50000), (2, 12, null, 10019, "2024-09-29", 50000)	Error Code: 1644. Помилка, pay: arrival_id або order_id має бути заповнене
16 14:43:24	INSERT INTO pay (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum) VALUES(5392, 12, 1000, null, "2024-09-29", 50000)	Error Code: 1644. Помилка, pay: треба заповнити обов'язкові поля
17 14:43:44	INSERT INTO pay (customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum) VALUES(5392, 12, 1000, null, 50000), (2, 12, null, 10019, "2024-09-29", 50000)	Error Code: 1644. Помилка, pay: треба заповнити обов'язкові поля

Рис. 3.1.7.1 Робота тригера Before Insert при додаванні даних в таблицю pay

3.1.7.2 Before Update (RESTRICT)

```
DELIMITER //  
CREATE TRIGGER pay_before_update  
BEFORE UPDATE ON pay  
FOR EACH ROW  
BEGIN  
    -- Перевірка, чи існує вказаний customer_id в таблиці customer  
    IF NEW.customer_id <> OLD.customer_id AND NOT EXISTS  
        (SELECT 1 FROM customers c WHERE c.customer_id =  
        NEW.customer_id) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка, pay: вказаний customer_id не існує в таблиці  
        customers.';  
    END IF;  
  
    -- перевіряємо, чи існує вказаний employee_id в таблиці employee  
    IF NEW.employee_id <> OLD.employee_id AND NOT EXISTS  
        (SELECT 1 FROM employees e WHERE e.employee_id =  
        NEW.employee_id) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка, pay: вказаний employee_id не існує  
        в таблиці employees.';  
    END IF;  
  
    -- Перевірка, arrival_id, order_id  
    IF (NEW.arrival_id IS NULL AND NEW.order_id IS NULL) OR
```

```

        (NEW.arrival_id IS NOT NULL AND NEW.order_id IS NOT NULL)

THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, рах: тільки одне з полів: arrival_id
або order_id має бути заповнене';

    END IF;

-- перевіряємо, чи існує вказаний order_id для даного customer_id в таблиці orders
IF NEW.order_id <> OLD.order_id THEN
    IF NOT EXISTS (SELECT 1 FROM orders o WHERE o.customer_id =
NEW.customer_id and o.order_id = NEW.order_id) THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка, рах: не існує вказаний
order_id для даного customer_id в таблиці orders.';

        END IF;
    END IF;

-- перевіряємо, чи існує вказаний arrival_id для даного customer_id в таблиці arrivals
IF NEW.arrival_id <> OLD.arrival_id THEN
    IF NOT EXISTS (SELECT 1 FROM arrivals a WHERE a.customer_id =
NEW.customer_id and a.arrival_id = NEW.arrival_id) THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка, рах: не існує вказаний
order_id для даного customer_id в таблиці arrivals.';

        END IF;
    END IF;

-- Перевірка, обов'язкових полів
IF NEW.customer_id IS NULL OR
    NEW.employee_id IS NULL OR
    NEW.date_pay IS NULL OR
    NEW.pay_sum IS NULL THEN
    SIGNAL SQLSTATE '45000'

```

```

SET MESSAGE_TEXT = "Помилка, рахунок треба заповнити обов'язкові
поля";
END IF;

```

```

-- Заборона оновлення зв'язків, якщо pay_id змінюється - Update RESTRICT
IF NEW.pay_id <> OLD.pay_id THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, рахунок pay_id не дозволена.';
END IF;
END//  

DELIMITER ;

```

До оновлення даних

```

6      -- оплата по замовленню №1987
7 •  UPDATE pay
8      SET customer_id = 771, employee_id = 12, arrival_id = NULL, order_id = 1987,
9          date_pay = '2024-07-08', pay_sum = 68200, pay_form = 'готівка'
L0      WHERE pay_id = 11008;
L1
L2      -- оплата по приходу №4
L3 •  UPDATE pay
L4      SET customer_id = 592, employee_id = 12, arrival_id = 4, order_id = NULL,
L5          date_pay = '2024-07-30', pay_sum = 682000
L6      WHERE pay_id = 11009;
L7
L8 •  SELECT pay_id, customer_id, employee_id, arrival_id, order_id, date_pay, pay_sum, pay_form
L9      FROM pay ORDER BY pay_id DESC;

```

result Grid							
pay_id	customer_id	employee_id	arrival_id	order_id	date_pay	pay_sum	pay_form
11009	5392	12	1000	NULL	2024-09-29	50000	оплата на р.р.
11008	1	12	NULL	10019	2024-09-29	50000	оплата на р.р.
----	----	----	----	NULL	-----	-----	-----

Після оновлення даних

```

✓ 44 21:06:03 UPDATE pay SET customer_id = 771, employee_id = 12, arrival_id = NULL, order_id = 1987, date... 1 row(s) affected Rows matched: 1 Changed: 1
✓ 45 21:06:20 UPDATE pay SET customer_id = 592, employee_id = 12, arrival_id = 4, order_id = NULL, date_pay... 1 row(s) affected Rows matched: 1 Changed: 1

```

pay_id	customer_id	employee_id	arrival_id	order_id	date_pay	pay_sum	pay_form
11009	592	12	4	NULL	2024-07-30	682000	оплата на р.р.
11008	771	12	NULL	1987	2024-07-08	68200	готівка

Помилки при оновленні даних, що визвав тригер

47 21:08:25	UPDATE pay SET customer_id = 33771, employee_id = 12, arrival_id = NULL, order_id = 1987, date_pay = '2024-07-08', pay_sum = 68200, pay_form = 'готівка' WHERE pay_id = 11008	Error Code: 1644. Помилка, ру: вказаний customer_id не існує в таблиці customers.
48 21:08:40	UPDATE pay SET customer_id = 771, employee_id = 52, arrival_id = NULL, order_id = 1987, date_pay = '2024-07-08', pay_sum = 68200, pay_form = 'готівка' WHERE pay_id = 11008	Error Code: 1644. Помилка, ру: вказаний employee_id не існує в таблиці employees.
49 21:09:02	UPDATE pay SET customer_id = 771, employee_id = 4, arrival_id = 4, order_id = 1987, date_pay = '2024-07-08', pay_sum = 68200, pay_form = 'готівка' WHERE pay_id = 11008	Error Code: 1644. Помилка, ру: тільки одне з полів: arrival_id або order_id має бути заповнене.
50 21:09:55	UPDATE pay SET customer_id = 771, employee_id = 12, arrival_id = NULL, order_id = 1988, date_pay = '2024-07-08', pay_sum = 68200, pay_form = 'готівка' WHERE pay_id = 11008	Error Code: 1644. Помилка, ру: тільки одне з полів: arrival_id або order_id має бути заповнене.
51 21:10:15	UPDATE pay SET customer_id = 592, employee_id = 12, arrival_id = 5, order_id = NULL, date_pay = '2024-07-30', pay_sum = 682000 WHERE pay_id = 11009	Error Code: 1644. Помилка, ру: не існує вказаний order_id для даного customer_id в таблиці orders.
52 21:12:29	UPDATE pay SET customer_id = 592, employee_id = 12, arrival_id = 4, order_id = NULL, date_pay = '2024-07-30', pay_sum = NULL WHERE pay_id = 11009	Error Code: 1644. Помилка, ру: не існує вказаний order_id для даного customer_id в таблиці orders.
53 21:12:48	UPDATE pay SET customer_id = 592, employee_id = 12, arrival_id = 4, order_id = NULL, date_pay = NULL, pay_sum = 682000 WHERE pay_id = 11009	Error Code: 1644. Помилка, ру: треба заповнити обов'язкове поле
54 21:13:58	UPDATE pay SET pay_id = 11011 WHERE pay_id = 11009	Error Code: 1644. Помилка, ру: зміна pay_id не дозволена.

Рис. 3.1.7.2 Робота тригера Before Update при оновленні даних в таблиці pay

3.1.7.3 Before Delete (RESTRICT)

DELIMITER //

CREATE TRIGGER pay_before_delete

BEFORE DELETE ON pay

FOR EACH ROW

BEGIN

-- Перевірка, чи read_only = 1

IF OLD.read_only = 1 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE_TEXT = 'Помилка, ру: цю оплату видалити не можна, тільки до читання';

END IF;

END//

DELIMITER ;

До видалення даних

```

23 •   SELECT pay_id, customer_id, employee_id, arrival_id, order_id,
24           date_pay, pay_sum, pay_form, read_only
25   FROM pay ORDER BY pay_id DESC;
26
27 •   DELETE FROM pay
28   WHERE pay_id = 11007;

```

Result Grid									Filter Rows:	Edit:	Export/Import:	Wrap Cell Content
pay_id	customer_id	employee_id	arrival_id	order_id	date_pay	pay_sum	pay_form	read_only				
11009	592	12	4	NULL	2024-07-30	682000	оплата на р.р.	1				
11008	771	12	NULL	1987	2024-07-08	68200	готівка	1				
11007	5392	12	1000	NULL	2024-09-29	50000	оплата на р.р.	0				
11006	5392	12	1000	12000	2024-09-29	50000	оплата на р.р.	0				

Після видалення даних

59 21:25:25 DELETE FROM pay WHERE pay_id = 11007 1 row(s) affected

pay_id	customer_id	employee_id	arrival_id	order_id	date_pay	pay_sum	pay_form	read_only
11009	592	12	4	NULL	2024-07-30	682000	оплата на р.р.	1
11008	771	12	NULL	1987	2024-07-08	68200	готівка	1
11006	5392	12	1000	NULL	2024-09-29	50000	оплата на р.р.	0

Помилки при видаленні даних, що визвав тригер

61 21:27:06 DELETE FROM pay WHERE pay_id = 11008 Error Code: 1644. Помилка, pay: що оплату видалити не можна, тільки до читання

Рис. 3.1.7.3 Робота тригеру Before Delete при видаленні даних з таблиці pay

3.1.8 Таблиця orders_goods

В тригерах для цієї таблиці ми стикаємося з появою бізнес-логіки, такою як перевірка наявності потрібної кількості запчастини, та встановлення нових значень stock, reserve.

3.1.8.1 Before Insert (RESTRICT)

DELIMITER //

```

CREATE TRIGGER orders_goods_before_insert
BEFORE INSERT ON orders_goods
FOR EACH ROW
BEGIN
    DECLARE new_price_out INT;
    DECLARE old_stock decimal(7,3);
    DECLARE old_reserve decimal(7,3);

    -- Перевірка, чи існує вказаний goods_id в таблиці goods
    IF NOT EXISTS (SELECT 1 FROM goods g WHERE g.goods_id = NEW.goods_id)
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_goods: вказаний goods_id не існує в
таблиці goods.';
```

```

END IF;

-- Перевірка, чи існує вказаний order_id в таблиці orders і чи він не відгружен
IF NOT EXISTS (SELECT 1 FROM orders o WHERE o.order_id = NEW.order_id
AND date_returned IS NULL) THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_goods: вказаний order_id не існує в
таблиці orders або замовлення вже відгружене.';

END IF;

-- Встановлення price_out, считування stock, reserve
SELECT stock INTO old_stock FROM goods WHERE goods_id = NEW.goods_id;
SELECT reserve INTO old_reserve FROM goods WHERE goods_id =
NEW.goods_id;
SELECT price INTO new_price_out FROM goods_dscr JOIN goods USING
(goods_dscr_id) WHERE goods_id = NEW.goods_id;
SET NEW.price_out = new_price_out;

-- Чи є потрібна кількість
IF NEW.quantity IS NULL OR NEW.quantity > old_stock - old_reserve THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_goods: кількість
запчастини недостатня.';

ELSE
    -- Якщо кількість є, оновлюємо склад
    IF NEW.is_shipped = 0 THEN
        UPDATE goods SET reserve = old_reserve + NEW.quantity
WHERE goods_id = NEW.goods_id;
    ELSE
        UPDATE goods SET stock = old_stock - NEW.quantity WHERE
goods_id = NEW.goods_id;
    END IF;
END IF;
END//
```

DELIMITER ;

До вставки даних

```
11 •  SELECT order_id, date_invoice, date_returned
12      FROM orders
13      WHERE order_id = 10012;
```

Result Grid		
order_id	date_invoice	date_returned
10012	2024-11-06	NULL
NULL	NULL	NULL
NULL	NULL	NULL

```
15 •  SELECT order_id, goods_id, quantity,
16          price_out, is_shipped
17      FROM orders_goods
```

Result Grid				
order_id	goods_id	quantity	price_out	is_shipped
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL

```
22 •  SELECT goods_id, stock, reserve
23      FROM goods
24      WHERE goods_id in (9999, 10000);
```

Result Grid		
goods_id	stock	reserve
9999	4.000	0.000
10000	1.000	0.000
NULL	NULL	NULL

```
17 •  SELECT goods_id, price as price_out
18      FROM goods_dscr
19      JOIN goods USING (goods_dscr_id)
20      WHERE goods_id in (9999, 10000);
```

Result Grid	
goods_id	price_out
9999	9376
10000	5956

Вставка даних з блокуванням таблиц

```
1 •  LOCK TABLES goods WRITE, orders_goods WRITE;
2 •  INSERT INTO orders_goods (order_id, goods_id, quantity, price_out, is_shipped)
3      VALUES (10012, 10000, 1, 15000, 0), (10012, 9999, 2, 25000, 1);
4 •  UNLOCK TABLES;
```

Після вставки даних

```
93 23:17:58 LOCK TABLES goods WRITE, orders_goods WRITE          0 row(s) affected
94 23:17:58 INSERT INTO orders_goods (order_id, goods_id, quantity, price_out, is_shipped) VALUES (10012, 10000, 1, 15000, 0), (10012, 9999, 2, 25000, 1) 2 row(s) affected Records: 2
95 23:17:58 UNLOCK TABLES                                     0 row(s) affected
```

```
15 •  SELECT order_id, goods_id, quantity,
16          price_out, is_shipped
17      FROM orders_goods
18      WHERE order_id = 10012;
```

Result Grid				
order_id	goods_id	quantity	price_out	is_shipped
10012	9999	2.000	9376	1
10012	10000	1.000	5956	0
NULL	NULL	NULL	NULL	NULL

```
22 •  SELECT goods_id, stock, reserve
23      FROM goods
24      WHERE goods_id in (9999, 10000);
```

Result Grid		
goods_id	stock	reserve
9999	2.000	0.000
10000	1.000	1.000

Помилки при додаванні даних, що визвав тригер

```
100 00:07:44 INSERT INTO orders_goods (order_id, goods_id, quantity, price_out, is_shipped) VALUES (10012, 10000, 1, 15000, 0), (10012, 9999, 2, 25000, 1) Error Code: 1644. Помилка, orders_goods: кількість запчастини недостатня.
101 00:08:48 INSERT INTO orders_goods (order_id, goods_id, quantity, price_out, is_shipped) VALUES (10012, 9999, 1, 25000, 1) Error Code: 1644. Помилка, orders_goods: вказаний order_id не існує в таблиці orders або замовлення вже відгружене.
102 00:09:31 INSERT INTO orders_goods (order_id, goods_id, quantity, price_out, is_shipped) VALUES (10019, 9999, 1, 25000, 1) Error Code: 1644. Помилка, orders_goods: вказаний order_id не існує в таблиці orders або замовлення вже відгружене.
103 00:10:06 INSERT INTO orders_goods (order_id, goods_id, quantity, price_out, is_shipped) VALUES (10012, 19999, 1, 25000, 1) Error Code: 1644. Помилка, orders_goods: вказаний goods_id не існує в таблиці goods.
```

order_id	date_invoice	date_returned
10019	NULL	NULL

Рис. 3.1.8.1 Робота тригера Before Insert при додаванні даних в таблицю orders_goods

3.1.8.2 Before Update (RESTRICT)

Я не став викладати код тригера в текстовому вигляді, бо він дуже об'ємний та зовсім не читабельний у простому текстовому форматі, привожу скриншот програми MySQL WorkBench на рис. 3.1.8.2.1

```

1  DELIMITER //
2  •  DROP TRIGGER orders_goods_before_update;
3
4  CREATE TRIGGER orders_goods_before_update
5  BEFORE UPDATE ON orders_goods
6  FOR EACH ROW
7  BEGIN
8      DECLARE new_stock_new_goods decimal(7,3);
9      DECLARE new_reserve_new_goods decimal(7,3);
10     DECLARE old_stock_old_goods decimal(7,3);
11     DECLARE old_reserve_old_goods decimal(7,3);
12     DECLARE message CHAR(110);
13
14     -- Перевірка, чи існує вказаний goods_id в таблиці goods
15     IF NEW.goods_id <> OLD.goods_id AND
16         NOT EXISTS (SELECT 1 FROM goods g WHERE g.goods_id = NEW.goods_id) THEN
17         SET message = CONCAT('Помилка, orders_goods: вказаний goods_id = ', NEW.goods_id, ' не існує в таблиці goods.');
18         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
19     END IF;
20
21     -- Перевірка, чи існує вказаний order_id в таблиці orders і чи він не відвантажен
22     IF NEW.order_id <> OLD.order_id AND
23         NOT EXISTS (SELECT 1 FROM orders o WHERE o.order_id = NEW.order_id AND date_returned IS NULL) THEN
24         SET message = CONCAT(
25             'Помилка, orders_goods: вказаний order_id = ', NEW.order_id, ' не існує в таблиці orders або замовлення вже відвантажено.');
26         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
27     END IF;
28
29     IF NEW.goods_id <> OLD.goods_id THEN
30         IF NEW.is_shipped = OLD.is_shipped THEN
31             IF NEW.is_shipped = 1 THEN      -- change goods on shipment
32                 -- повернення старого на склад
33                 SELECT stock INTO old_stock_old_goods FROM goods WHERE goods_id = OLD.goods_id;
34                 UPDATE goods SET stock = old_stock_old_goods + OLD.quantity WHERE goods_id = OLD.goods_id;
35
36                 -- списання нового зі складу
37                 SELECT stock, reserve INTO new_stock_new_goods, new_reserve_new_goods FROM goods WHERE goods_id = NEW.goods_id;
38                 IF new_stock_new_goods - new_reserve_new_goods - NEW.quantity >= 0 THEN
39                     UPDATE goods SET stock = new_stock_new_goods - NEW.quantity WHERE goods_id = NEW.goods_id;

```

```

40
41     ELSE
42         SET message = CONCAT('Помилка, orders_goods: кількість запчастини goods_id = ', NEW.goods_id, ' недостатня.');
43         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
44     END IF;
45
46     ELSE -- change goods on reserve
47         -- видалення старого з резерву
48         SELECT reserve INTO old_reserve_old_goods FROM goods WHERE goods_id = OLD.goods_id;
49         UPDATE goods SET reserve = old_reserve_old_goods - OLD.quantity WHERE goods_id = OLD.goods_id;
50
51         -- додавання нового до резерву
52         SELECT stock, reserve INTO new_stock_new_goods, new_reserve_new_goods FROM goods WHERE goods_id = NEW.goods_id;
53         IF new_stock_new_goods - new_reserve_new_goods - NEW.quantity >= 0 THEN
54             UPDATE goods SET reserve = new_reserve_new_goods + NEW.quantity WHERE goods_id = NEW.goods_id;
55         ELSE
56             SET message = CONCAT('Помилка, orders_goods: кількість запчастини goods_id = ', NEW.goods_id, ' недостатня.');
57             SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
58         END IF;
59     END IF;
60
61     ELSE -- NEW.is_shipped <> OLD.is_shipped -- change goods on shipment and reserve
62         IF NEW.is_shipped = 1 THEN -- and OLD.is_shipped = 0
63             -- видалення старого з резерву
64             SELECT reserve INTO old_reserve_old_goods FROM goods WHERE goods_id = OLD.goods_id;
65             UPDATE goods SET reserve = old_reserve_old_goods - OLD.quantity WHERE goods_id = OLD.goods_id;
66
67             -- списання нового зі складу
68             SELECT stock, reserve INTO new_stock_new_goods, new_reserve_new_goods FROM goods WHERE goods_id = NEW.goods_id;
69             IF new_stock_new_goods - new_reserve_new_goods - NEW.quantity >= 0 THEN
70                 UPDATE goods SET stock = new_stock_new_goods - NEW.quantity WHERE goods_id = NEW.goods_id;
71             ELSE
72                 SET message = CONCAT('Помилка, orders_goods: кількість запчастини goods_id = ', NEW.goods_id, ' недостатня.');
73                 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
74             END IF;
75
76             ELSE -- NEW.is_shipped = 0 and OLD.is_shipped = 1
77                 -- повернення старого на склад
78                 SELECT stock INTO old_stock_old_goods FROM goods WHERE goods_id = OLD.goods_id;
79                 UPDATE goods SET stock = old_stock_old_goods + OLD.quantity WHERE goods_id = OLD.goods_id;
80
81                 -- додавання нового до резерву
82                 SELECT stock, reserve INTO new_stock_new_goods, new_reserve_new_goods FROM goods WHERE goods_id = NEW.goods_id;
83                 IF new_stock_new_goods - new_reserve_new_goods - NEW.quantity >= 0 THEN
84                     UPDATE goods SET reserve = new_reserve_new_goods + NEW.quantity WHERE goods_id = NEW.goods_id;
85                 ELSE
86                     SET message = CONCAT('Помилка, orders_goods: кількість запчастини goods_id = ', NEW.goods_id, ' недостатня.');
87                     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
88                 END IF;
89             END IF;
90
91             ELSE -- NEW.goods_id = OLD.goods_id
92                 IF NEW.is_shipped <> OLD.is_shipped THEN
93                     IF NEW.is_shipped = 1 THEN -- OLD.is_shipped = 0
94                         -- видалення з резерву та списання зі складу, reserve -> stock
95                         SELECT stock, reserve INTO new_stock_new_goods, new_reserve_new_goods FROM goods WHERE goods_id = NEW.goods_id;
96                         IF new_stock_new_goods - new_reserve_new_goods + OLD.quantity - NEW.quantity >= 0 THEN
97                             UPDATE goods
98                             SET stock = new_stock_new_goods - NEW.quantity,
99                             reserve = new_reserve_new_goods - OLD.quantity
100                            WHERE goods_id = NEW.goods_id;
101
102                         ELSE
103                             SET message = CONCAT('Помилка, orders_goods: кількість запчастини goods_id = ', NEW.goods_id, ' недостатня.');
104                             SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;

```

```

104      END IF;
105      -- NEW.is_shipped = 0 and OLD.is_shipped = 1
106      -- повернення на склад, додавання до резерву,
107      SELECT stock, reserve INTO new_stock_new_goods, new_reserve_new_goods FROM goods WHERE goods_id = NEW.goods_id;
108
109      IF new_stock_new_goods - new_reserve_new_goods + OLD.quantity - NEW.quantity >= 0 THEN
110          UPDATE goods
111          SET stock = new_stock_new_goods + OLD.quantity,
112              reserve = new_reserve_new_goods + NEW.quantity
113              WHERE goods_id = NEW.goods_id;
114      ELSE
115          SET message = CONCAT('Помилка, orders_goods: кількість запчастини goods_id = ', NEW.goods_id, ' недостатня.');
116          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
117      END IF;
118  END IF;
119      -- NEW.is_shipped = OLD.is_shipped
120      IF NEW.quantity <> OLD.quantity AND NEW.is_shipped = 1 THEN
121          -- пересписання зі складу в іншій кількості
122          SELECT stock, reserve INTO new_stock_new_goods, new_reserve_new_goods FROM goods WHERE goods_id = NEW.goods_id;
123          IF new_stock_new_goods - new_reserve_new_goods + OLD.quantity - NEW.quantity >= 0 THEN
124              UPDATE goods
125              SET stock = new_stock_new_goods + OLD.quantity - NEW.quantity
126              WHERE goods_id = NEW.goods_id;
127          ELSE
128              SET message = CONCAT('Помилка, orders_goods: кількість запчастини goods_id = ', NEW.goods_id, ' недостатня.');
129              SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
130          END IF;
131      ELSEIF NEW.quantity <> OLD.quantity AND NEW.is_shipped = 0 THEN
132          -- перерахунок резерву
133          SELECT stock, reserve INTO new_stock_new_goods, new_reserve_new_goods FROM goods WHERE goods_id = NEW.goods_id;
134          IF new_stock_new_goods - new_reserve_new_goods + OLD.quantity - NEW.quantity >= 0 THEN
135              UPDATE goods
136              SET reserve = new_reserve_new_goods + NEW.quantity - OLD.quantity
137              WHERE goods_id = NEW.goods_id;
138          ELSE
139              SET message = CONCAT('Помилка, orders_goods: кількість запчастини goods_id = ', NEW.goods_id, ' недостатня.');
140              SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
141          END IF;
142      END IF;
143  END IF;
144 END IF;
145 END//  

146 DELIMITER ;

```

Рис. 3.1.8.2 скриншот тригеру Before Update

До оновлення даних

```

15 •  SELECT order_id, goods_id,
16      quantity, is_shipped
17  FROM orders_goods
18 WHERE order_id = 10009;

```

Result Grid			
order_id	goods_id	quantity	is_shipped
10009	1	1.000	0
10009	2	1.000	0
10009	3	1.000	1
NULL	NULL	NULL	NULL

```

22 •  SELECT goods_id, stock, reserve
23  FROM goods
24 WHERE goods_id in (1, 2, 3, 4, 5, 6);

```

Result Grid		
goods_id	stock	reserve
1	14.000	6.000
2	12.000	3.000
3	21.000	1.000
4	25.000	2.000
6	22.000	0.000
NULL	NULL	NULL

Після оновлення даних

7 19:13:24 UPDATE orders_goods SET quantity = 2, is_shipped = 1 WHERE order_id = 10009 AND goods_id = 1 1 row(s) affected Rows matched: 1 Changed: 1

```
UPDATE orders_goods
SET quantity = 2,
is_shipped = 1
WHERE order_id = 10009
AND goods_id = 1;
```

order_id	goods_id	quantity	is_shipped
10009	1	2.000	1
10009	2	1.000	0
10009	3	1.000	1

goods_id	stock	reserve
1	12.000	5.000
2	12.000	3.000
3	21.000	1.000
4	25.000	2.000
6	22.000	0.000
NULL	NULL	NULL

11 19:19:23 UPDATE orders_goods SET quantity = 10 WHERE order_id = 10009 AND goods_id = 2 1 row(s) affected Rows matched: 1 Changed: 1

```
UPDATE orders_goods
SET quantity = 10
WHERE order_id = 10009
AND goods_id = 2;
```

order_id	goods_id	quantity	is_shipped
10009	1	2.000	1
10009	2	10.000	0
10009	3	1.000	1
NULL	NULL	NULL	NULL

goods_id	stock	reserve
1	12.000	5.000
2	12.000	12.000
3	21.000	1.000
4	25.000	2.000
6	22.000	0.000
NULL	NULL	NULL

14 19:24:36 UPDATE orders_goods SET quantity = 2, is_shipped = 0, goods_id = 4 WHERE order_id = 10009 AND goods_id = 3 1 row(s) affected Rows matched: 1 Changed: 1

```
UPDATE orders_goods
SET quantity = 2,
is_shipped = 0,
goods_id = 4
WHERE order_id = 10009
AND goods_id = 3;
```

order_id	goods_id	quantity	is_shipped
10009	1	2.000	1
10009	2	10.000	0
10009	4	2.000	0
NULL	NULL	NULL	NULL

goods_id	stock	reserve
1	12.000	5.000
2	12.000	12.000
3	22.000	1.000
4	25.000	4.000
6	22.000	0.000
NULL	NULL	NULL

17 19:28:37 UPDATE orders_goods SET quantity = 1, is_shipped = 0, goods_id = 5 WHERE order_id = 10009 AND goods_id = 4 Error Code: 1644. Помилка, orders_goods: вказаній goods_id = 5 не існує в таблиці goods.

```
UPDATE orders_goods
SET quantity = 1,
is_shipped = 0,
goods_id = 5
WHERE order_id = 10009
AND goods_id = 4;
```

order_id	goods_id	quantity	is_shipped
10009	1	2.000	1
10009	2	10.000	0
10009	4	2.000	0

goods_id	stock	reserve
1	12.000	5.000
2	12.000	12.000
3	22.000	1.000
4	25.000	4.000
6	22.000	0.000
NULL	NULL	NULL

UPDATE orders_goods SET order_id = 10019 WHERE order_id = 10009;

20 19:33:05 UPDATE orders_goods SET order_id = 10019 WHERE order_id = 10009 Error Code: 1644. Помилка, orders_goods: вказаній order_id = 10019 не існує в таблиці orders або замовлення вже відвантажено.

21 19:37:04 UPDATE orders_goods SET quantity = 25, is_shipped = 0, goods_id = 6 WHERE order_id = 10009 AND goods_id = 4 Error Code: 1644. Помилка, orders_goods: кількість запчастини goods_id = 6 недостатнія.

```
UPDATE orders_goods
SET quantity = 25,
is_shipped = 0,
goods_id = 6
WHERE order_id = 10009
AND goods_id = 4;
```

order_id	goods_id	quantity	is_shipped	goods_id	stock	reserve
10009	1	2.000	1	1	12.000	5.000
10009	2	10.000	0	2	12.000	12.000
10009	4	2.000	0	3	22.000	1.000
				4	25.000	2.000
				6	22.000	0.000

24 19:40:22 UPDATE orders_goods SET quantity = 25, is_shipped = 1 WHERE order_id = 10009 AND goods_id = 4 1 row(s) affected Rows matched: 1 Changed: 1

```
UPDATE orders_goods
SET quantity = 25,
is_shipped = 1
WHERE order_id = 10009
AND goods_id = 4;
```

order_id	goods_id	quantity	is_shipped	goods_id	stock	reserve
10009	1	2.000	1	1	12.000	5.000
10009	2	10.000	0	2	12.000	12.000
10009	4	25.000	1	3	22.000	1.000
				4	0.000	0.000
				6	22.000	0.000

27 19:44:03 UPDATE orders_goods SET quantity = 10 WHERE order_id = 10009 AND goods_id = 1 Error Code: 1644. Помилка, orders_goods: кількість запчастини goods_id = 1 недостатня.

```
UPDATE orders_goods
SET quantity = 10
WHERE order_id = 10009
AND goods_id = 1;
```

order_id	goods_id	quantity	is_shipped	goods_id	stock	reserve
10009	1	2.000	1	1	12.000	5.000
10009	2	10.000	0	2	12.000	12.000
10009	4	25.000	1	3	22.000	1.000
				4	0.000	0.000

Рис. 3.1.8.4 Робота тригера Before Update при оновленні даних в таблиці orders_goods

Тригер на оновлення даних працює як слід, перевіряє наявну кількість запчастин, та запчастин у резерві, якщо кількості не вистачає генерує помилку. Також генерує помилку, якщо вказаної запчастини немає у таблиці goods та якщо змінюється номер замовлення на не існуюче або вже відвантажене Замовнику.

В подальшему ми створимо тригер After Update, який буде підсумовувати загальну суму замовлення і зберігати її в поле total таблиці orders (це поле буде додано в наступних роботах).

Також, при використанні тригера у високонавантажених системах на таблицях MyISAM, рекомендується виконувати оновлення с блокуванням

таблиц на запис, як ми це робили при вставці даних (дивись приклади використання попереднього тригера Before Insert)

3.1.8.2 Before Delete (RESTRICT)

```
DELIMITER //  
CREATE TRIGGER orders_goods_before_delete  
BEFORE DELETE ON orders_goods  
FOR EACH ROW  
BEGIN  
  
    DECLARE old_reserve_old_goods decimal(7,3);  
    DECLARE message CHAR(110);  
  
    -- Перевірка, чи запчастина відгруженна  
    IF OLD.is_shipped = 1 THEN  
        SET message = CONCAT(  
            'Помилка, orders_goods: вказаний goods_id = ', OLD.goods_id, ' вже відвантажено, не  
            можливо видалити.');?>;  
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;  
    END IF;  
  
    -- Перевірка, чи вказаний order_id в таблиці orders відвантажен  
    IF NOT EXISTS (SELECT 1 FROM orders o WHERE o.order_id = OLD.order_id AND  
        o.date_returned IS NULL) THEN  
        SET message = CONCAT(  
            'Помилка, orders_goods: вказаний order_id = ', OLD.order_id, ' вже відвантажено, не  
            можливо видалити.');?>;  
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;  
    END IF;  
  
    -- Якщо запис у таблиці orders_goods, має is_shipped = 0, а також orders.date_returned IS NULL  
    -- можемо видалити цей запис з таблиці orders_goods, перерахуємо резерв  
    SELECT reserve INTO old_reserve_old_goods FROM goods WHERE goods_id  
        = OLD.goods_id;
```

```
UPDATE goods
      SET reserve = old_reserve - OLD.quantity
      WHERE goods_id = OLD.goods_id;

END//  
DELIMITER ;
```

До видалення даних

```
15 •  SELECT order_id, goods_id,
16      quantity, is_shipped
17  FROM orders_goods
18 WHERE order_id = 10009;
19
20
21
22 •  SELECT goods_id, stock, reserve
23  FROM goods
24 WHERE goods_id in (1, 2, 3, 4);
```

Після видалення даних

40 21:41:10	DELETE FROM orders_goods WHERE order_id = 10009 AND goods_id = 2	1 row(s) affected	
<pre>DELETE FROM orders_goods WHERE order_id = 10009 AND goods_id = 2;</pre>			
49 23:21:21	DELETE FROM orders_goods WHERE order_id = 10009 AND goods_id = 1	Error Code: 1644. Помилка, orders_goods: вказаний goods_id = 1 вже відвантажено, не можливо видалити.	
<pre>DELETE FROM orders_goods WHERE order_id = 10009 AND goods_id = 1;</pre>			
77 23:46:53	DELETE FROM orders_goods WHERE order_id = 10000 AND goods_id = 3736	Error Code: 1644. Помилка, orders_goods: вказаний order_id = 10000 вже відвантажено, не можливо видалити.	
<pre>DELETE FROM orders_goods WHERE order_id = 10000;</pre>			

Рис. 3.1.8.4 Робота триггеру Before Delete при видаленні даних з таблиці
orders goods

Таблиця arrivals_goods

3.1.9.1 Before Insert (RESTRICT)

```
DELIMITER //  
CREATE TRIGGER arrivals_goods_before_insert  
BEFORE INSERT ON arrivals_goods  
FOR EACH ROW  
BEGIN  
  
    DECLARE old_stock decimal(7,3);  
    DECLARE message CHAR(110);  
  
    -- Перевірка, чи існує вказаний goods_id в таблиці goods  
    IF NOT EXISTS (SELECT 1 FROM goods g WHERE g.goods_id = NEW.goods_id)  
    THEN  
        SET message = CONCAT(  
            'Помилка, arrivals_goods: вказаний goods_id = ', NEW.goods_id, ' не існує в таблиці  
            goods.');        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;  
    END IF;  
  
    -- Перевірка, чи існує вказаний arrival_id в таблиці arrivals і чи arrivals.date_arrival is null  
    -- є заборона на встановлення arrivals.date_arrival якщо total не співпадає  
    IF NOT EXISTS (SELECT 1 FROM arrivals a WHERE a.arrival_id = NEW.arrival_id  
    AND a.date_arrival IS NULL) THEN  
        SET message = CONCAT(  
            'Помилка, arrivals_goods: вказаний arrival_id = ', NEW.arrival_id, ' не існує в таблиці  
            arrivals або встановлена date_arrival.');        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;  
    END IF;  
  
    -- Считування склада  
    SELECT stock INTO old_stock FROM goods WHERE goods_id = NEW.goods_id;
```

```
-- Оновлюємо склад
UPDATE goods SET stock = old_stock + NEW.quantity WHERE goods_id =
NEW.goods_id;

END///
DELIMITER ;
```

До вставки даних

8 • <code>SELECT *</code>	22 • <code>SELECT goods_id,</code>
9 <code>FROM arrivals_goods</code>	23 <code>stock</code>
10 <code>WHERE arrival_id >= 1000;</code>	24 <code>FROM goods</code>
11	25 <code>WHERE goods_id = 1;</code>
12 • <code>INSERT INTO arrivals_goods</code>	
13 <code>VALUES (1004, 1, 10);</code>	

arrival_id	date_arrival
1000	2024-02-07
1002	NULL
1004	NULL
NULL	NULL

arrival_id	goods_id	quantity
1004	2	20.000
1004	12	10.000
1000	3697	21.000

goods_id	stock
1	12.000
NULL	NULL

Після вставки даних

14 12:07:10 `INSERT INTO arrivals_goods VALUES (1004, 1, 10)` 1 row(s) affected

arrival_id	goods_id	quantity
1004	1	10.000
1004	2	20.000
1004	12	10.000
1000	3697	21.000

goods_id	stock
1	22.000
NULL	NULL

Помилки при додавані даних, що визвав тригер

- ✖ 17 12:10:56 `INSERT INTO arrivals_goods VALUES (1000, 1, 10)` Error Code: 1644. Помилка, arrivals_goods: вказаний arrival_id = 1000 не існує в таблиці arrivals або встановлена date_arrival.
- ✖ 18 12:11:23 `INSERT INTO arrivals_goods VALUES (1004, 5, 10)` Error Code: 1644. Помилка, arrivals_goods: вказаний goods_id = 5 не існує в таблиці goods.
- ✖ 19 12:11:56 `INSERT INTO arrivals_goods VALUES (1005, 1, 10)` Error Code: 1644. Помилка, arrivals_goods: вказаний arrival_id = 1005 не існує в таблиці arrivals або встановлена date_arrival.

Рис. 3.1.9.1 Робота тригера Before Insert при додавані даних в таблицю `arrivals_goods`

3.1.9.2 Before Update (RESTRICT)

DELIMITER //

```

CREATE TRIGGER arrivals_goods_before_update
BEFORE UPDATE ON arrivals_goods
FOR EACH ROW
BEGIN

    DECLARE old_stock_old_goods decimal(7,3);
    DECLARE old_stock_new_goods decimal(7,3);
    DECLARE message CHAR(110);

    -- Перевірка, чи існує вказаний goods_id в таблиці goods
    IF NOT EXISTS (SELECT 1 FROM goods g WHERE g.goods_id = NEW.goods_id)
    THEN
        SET message = CONCAT(
            'Помилка, arrivals_goods: вказаний goods_id = ', NEW.goods_id, ' не існує в таблиці
            goods.');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
    END IF;

    -- Перевірка, чи існує вказаний arrival_id в таблиці arrivals і чи arrivals.date_arrival is null
    -- є заборона на встановлення arrivals.date_arrival якщо total не співпадає
    IF NOT EXISTS (SELECT 1 FROM arrivals a WHERE a.arrival_id = NEW.arrival_id
    AND a.date_arrival IS NULL) THEN
        SET message = CONCAT(
            'Помилка, arrivals_goods: вказаний arrival_id = ', NEW.arrival_id, ' не існує в таблиці
            arrivals або встановлена date_arrival.');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
    END IF;

    -- Считування та оновлення stock
    IF NEW.goods_id <> OLD.goods_id THEN
        SELECT stock INTO old_stock_old_goods FROM goods WHERE
        goods_id = OLD.goods_id;
        SELECT stock INTO old_stock_new_goods FROM goods WHERE
        goods_id = NEW.goods_id;
    END IF;

```

```

        UPDATE goods SET stock = old_stock_old_goods - OLD.quantity
        WHERE goods_id = OLD.goods_id;

        UPDATE goods SET stock = old_stock_new_goods + NEW.quantity WHERE
        goods_id = NEW.goods_id;

        ELSE          -- NEW.goods_id = OLD.goods_id

            IF NEW.quantity <> OLD.quantity THEN

                SELECT stock INTO old_stock_new_goods FROM goods WHERE
                goods_id = NEW.goods_id;

                UPDATE goods SET stock = old_stock_new_goods + NEW.quantity -
                OLD.quantity WHERE goods_id = NEW.goods_id;

            END IF;

        END IF;

    END//
```

DELIMITER ;

До оновлення даних

15 • <code>SELECT arrival_id,</code>	8 • <code>SELECT *</code>	22 • <code>SELECT goods_id,</code>
16 date_arrival	9 FROM arrivals_goods	23 stock, reserve
17 FROM arrivals	10 WHERE arrival_id >= 1000;	24 FROM goods
18 WHERE arrival_id >= 1000;		25 WHERE goods_id IN (1,2,3);

result Grid		result Grid			result Grid		
arrival_id	date_arrival	arrival_id	goods_id	quantity	goods_id	stock	reserve
1000	2024-02-07	1004	1	10.000	1	22.000	5.000
1002	NULL	1004	2	20.000	2	12.000	2.000
1004	NULL	1004	12	10.000	3	22.000	1.000
NULL	NULL	1000	3697	21.000	NULL	NULL	NULL

Після оновлення даних

26 12:45:01	<code>UPDATE arrivals_goods SET quantity = 15 WHERE arrival_id = 1004 AND goods_id = 1</code>	1 row(s) affected Rows matched: 1 Changed: 1
	<code>UPDATE arrivals_goods SET quantity = 15 WHERE arrival_id = 1004 AND goods_id = 1;</code>	

result Grid			result Grid		
arrival_id	goods_id	quantity	goods_id	stock	reserve
1004	1	15.000	1	27.000	5.000
1004	2	20.000	2	12.000	2.000
1004	12	10.000	3	22.000	1.000
1000	3697	21.000	NULL	NULL	NULL

3 19:42:22 `UPDATE arrivals_goods SET goods_id = 3, quantity = 10 WHERE arrival_id = 1004 AND goods_id = 1` 1 row(s) affected Rows matched: 1 Changed: 1

6 19:47:51 UPDATE arrivals_goods SET goods_id = 3, quantity = 10 WHERE arrival_id = 1004 AND goods_id = 1; Error Code: 1644. Помилка, arrivals_goods: вказаний goods_id = 3 не існує в таблиці goods.

UPDATE arrivals_goods
SET goods_id = 5,
quantity = 10
WHERE arrival_id = 1004
AND goods_id = 3;

arrival_id	goods_id	quantity
1004	2	20.000
1004	3	10.000
1004	12	10.000
1000	3697	21.000

goods_id	stock	reserve
1	12.000	5.000
2	12.000	2.000
3	32.000	1.000
3697	NULL	NULL

12 19:55:13 UPDATE arrivals_goods SET quantity = 25 WHERE arrival_id = 1000; Error Code: 1644. Помилка, arrivals_goods: вказаний arrival_id = 1000 не існує в таблиці arrivals або встановлена date_arrival.

UPDATE arrivals_goods
SET quantity = 25
WHERE arrival_id = 1000
AND goods_id = 3697;

arrival_id	goods_id	quantity
1004	2	20.000
1004	3	10.000
1004	12	10.000
1000	3697	21.000

goods_id	stock	reserve
1	12.000	5.000
2	12.000	2.000
3	32.000	1.000
3697	1.000	0.000
NULL	NULL	NULL

Рис. 3.1.9.2 Робота тригеру Before Update при додавані даних в таблицю arrivals_goods

3.1.9.3 Before Delete (RESTRICT)

```

DELIMITER //

CREATE TRIGGER arrivals_goods_before_delete
BEFORE DELETE ON arrivals_goods
FOR EACH ROW
BEGIN

DECLARE old_stock_old_goods decimal(7,3);
DECLARE message CHAR(110);

-- Перевірка, чи існує вказаний goods_id в таблиці goods
IF NOT EXISTS (SELECT 1 FROM goods g WHERE g.goods_id = OLD.goods_id)
THEN
    SET message = CONCAT(

```

```

'Помилка, arrivals_goods: вказаний goods_id = ', OLD.goods_id,' не існує в таблиці
goods.);

    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
END IF;

-- Перевірка, чи існує вказаний arrival_id в таблиці arrivals і чи arrivals.date_arrival is null
IF NOT EXISTS (SELECT 1 FROM arrivals a WHERE a.arrival_id = OLD.arrival_id
AND a.date_arrival IS NULL) THEN
    SET message = CONCAT(
        'Помилка, arrivals_goods: вказаний arrival_id = ', OLD.arrival_id,' не існує в таблиці
arrivals або встановлена date_arrival.');

    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
END IF;

-- Считування та оновлення stock
SELECT stock INTO old_stock_old_goods FROM goods WHERE goods_id =
OLD.goods_id;

UPDATE goods SET stock = old_stock_old_goods - OLD.quantity WHERE
goods_id = OLD.goods_id;

END//;
DELIMITER ;

```

До видалення даних

15 • <code>SELECT arrival_id,</code>	8 • <code>SELECT *</code>	23 • <code>SELECT goods_id,</code>
16 date_arrival	9 <code>FROM arrivals_goods</code>	24 stock, reserve
17 <code>FROM arrivals</code>	10 <code>WHERE arrival_id >= 1000;</code>	25 <code>FROM goods</code>
18 <code>WHERE arrival_id >= 1000;</code>		26 <code>WHERE goods_id IN (1,2,3,3697);</code>
result Grid Filter Rows: <input type="text"/>	result Grid Filter Rows: <input type="text"/>	result Grid Filter Rows: <input type="text"/>
arrival_id date_arrival	arrival_id goods_id quantity	goods_id stock reserve
1000 2024-02-07	1004 2 20.000	1 12.000 5.000
1002 NULL	1004 3 10.000	2 12.000 2.000
1004 NULL	1004 12 10.000	3 32.000 1.000
NULL NULL	1000 3697 21.000	3697 1.000 0.000

Після видалення даних

22 20:31:04 DELETE FROM arrivals_goods WHERE arrival_id = 1004 AND goods_id = 3 1 row(s) affected

```
DELETE FROM arrivals_goods
WHERE arrival_id = 1004
AND goods_id = 3;
```

arrival_id	goods_id	quantity
1004	2	20.000
1004	12	10.000
1000	3697	21.000

goods_id	stock	reserve
1	12.000	5.000
2	12.000	2.000
3	22.000	1.000
3697	1.000	0.000

Помилки при видаленні даних, що визвав тригер

26 20:35:01 DELETE FROM arrivals_goods WHERE arrival_id = 1000 AND goods_id = 3697 Error Code: 1644. Помилка, arrivals_goods: вказаний arrival_id = 1000 не існує в таблиці arrivals або в...

Рис. 3.1.9.3 Робота тригеру Before Delete при видаленні даних з таблиці `arrivals_goods`

Таблиця `orders_received`

3.1.10.1 Before Insert (RESTRICT)

```
DELIMITER //
CREATE TRIGGER orders_received_before_insert
BEFORE INSERT ON orders_received
FOR EACH ROW
BEGIN
    -- Перевірка, чи вже існує вказаний order_id в таблиці orders_received
    IF EXISTS (SELECT 1 FROM orders_received WHERE order_id = NEW.order_id)
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_received: вказаний order_id вже існує в
таблиці orders_received.';

    END IF;

    -- Перевірка, чи існує вказаний order_id в таблиці orders
    IF NOT EXISTS (SELECT 1 FROM orders o WHERE o.order_id = NEW.order_id
    AND o.date_returned IS NULL) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_received: вказаний order_id не існує в
таблиці orders, або вже відгружен Замовнику.';
```

```
END IF;
```

```
-- чи існує вказаний employee_id в таблиці employee
IF NOT EXISTS (SELECT 1 FROM employees e WHERE e.employee_id =
NEW.employee_id) THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_received: вказаний
employee_id не існує в таблиці employees.';
END IF;
```

```
-- Перевірка, чи існує вказаний brand_id в таблиці brand
IF NOT EXISTS (SELECT 1 FROM brand b WHERE b.brand_id = NEW.brand_id)
THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_received: вказаний brand_id не існує в
таблиці brand_id.';
END IF;
```

```
-- Перевірка, чи всі обов'язкові поля заповнені
IF NEW.model_name IS NULL OR
    NEW.sn IS NULL OR
    NEW.equipment IS NULL OR
    NEW.fault_description IS NULL THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_received: всі обов'язкові поля мають
бути заповнені.';

END IF;
```

```
END//
```

```
DELIMITER ;
```

До вставки даних

<pre>14 • SELECT order_id, 15 date_returned 16 FROM orders 17 WHERE order_id > 10015;</pre>	<pre>3 • SELECT order_id, brand_id 4 FROM orders_received 5 ORDER BY order_id DESC;</pre>																				
<p>Result Grid Filter Rows: []</p> <table border="1"> <thead> <tr> <th>order_id</th> <th>date_returned</th> </tr> </thead> <tbody> <tr> <td>10016</td> <td>NULL</td> </tr> <tr> <td>10017</td> <td>2024-11-25</td> </tr> <tr> <td>10018</td> <td>NULL</td> </tr> <tr> <td>10019</td> <td>2024-11-21</td> </tr> <tr> <td>10020</td> <td>NULL</td> </tr> </tbody> </table>	order_id	date_returned	10016	NULL	10017	2024-11-25	10018	NULL	10019	2024-11-21	10020	NULL	<p>Result Grid Filter Rows: []</p> <table border="1"> <thead> <tr> <th>order_id</th> <th>brand_id</th> </tr> </thead> <tbody> <tr> <td>10019</td> <td>3</td> </tr> <tr> <td>10011</td> <td>1</td> </tr> <tr> <td>10005</td> <td>1</td> </tr> </tbody> </table>	order_id	brand_id	10019	3	10011	1	10005	1
order_id	date_returned																				
10016	NULL																				
10017	2024-11-25																				
10018	NULL																				
10019	2024-11-21																				
10020	NULL																				
order_id	brand_id																				
10019	3																				
10011	1																				
10005	1																				

Після вставки даних

<p>34 23:30:00 INSERT INTO orders_received VALUES (10018, 3, 5, "1610", "SN1610-126", NULL) 1 row(s) affected</p> <pre>INSERT INTO orders_received VALUES (10018, 3, 5, "1610", "SN1610-126", NULL, "без кабелю живлення", "торохтиль під час увімкнення");</pre>	<p>Result Grid Filter Rows: []</p> <table border="1"> <thead> <tr> <th>order_id</th> <th>brand_id</th> </tr> </thead> <tbody> <tr> <td>10019</td> <td>3</td> </tr> <tr> <td>10018</td> <td>3</td> </tr> <tr> <td>10011</td> <td>1</td> </tr> <tr> <td>10005</td> <td>1</td> </tr> </tbody> </table>	order_id	brand_id	10019	3	10018	3	10011	1	10005	1
order_id	brand_id										
10019	3										
10018	3										
10011	1										
10005	1										

Помилки при додаванні даних, що взвав тригер

38 23:37:48 INSERT INTO orders_received VALUES (10018, 3, 5, "1610", "SN1610-126", NULL, "без кабелю живлення", "торохтиль під час увімкнення")	Error Code: 1644. Помилка, orders_received: вказаний order_id вже існує в таблиці orders_received.
39 23:38:05 INSERT INTO orders_received VALUES (10017, 3, 5, "1610", "SN1610-126", NULL, "без кабелю живлення", "торохтиль під час увімкнення")	Error Code: 1644. Помилка, orders_received: вказаний order_id не існує в таблиці orders, або вже відгружен Замовнику.
40 23:38:37 INSERT INTO orders_received VALUES (10020, 3, 5, "1610", "SN1610-126", NULL, "без кабелю живлення", "торохтиль під час увімкнення")	Error Code: 1644. Помилка, orders_received: вказаний order_id не існує в таблиці orders, або вже відгружен Замовнику.
41 23:38:58 INSERT INTO orders_received VALUES (10016, 33, 5, "1610", "SN1610-126", NULL, "без кабелю живлення", "торохтиль під час увімкнення")	Error Code: 1644. Помилка, orders_received: вказаний brand_id не існує в таблиці brand_id.
42 23:39:06 INSERT INTO orders_received VALUES (10016, 3, 35, "1610", "SN1610-126", NULL, "без кабелю живлення", "торохтиль під час увімкнення")	Error Code: 1644. Помилка, orders_received: вказаний employee_id не існує в таблиці employees.
43 23:39:22 INSERT INTO orders_received VALUES (10016, 3, 5, "1610", "SN1610-126", NULL, "без кабелю живлення", "торохтиль під час увімкнення")	Error Code: 1644. Помилка, orders_received: всі обов'язкові поля мають бути заповнені.
44 23:39:39 INSERT INTO orders_received VALUES (10016, 3, 5, "1610", "SN1610-126", NULL, "без кабелю живлення", "торохтиль під час увімкнення")	Error Code: 1644. Помилка, orders_received: всі обов'язкові поля мають бути заповнені.
45 23:40:02 INSERT INTO orders_received VALUES (10016, 3, 5, "1610", "SN1610-126", NULL, NULL, "торохтиль під час увімкнення")	Error Code: 1644. Помилка, orders_received: всі обов'язкові поля мають бути заповнені.
46 23:40:13 INSERT INTO orders_received VALUES (10016, 3, 5, "1610", "SN1610-126", NULL, "без кабелю живлення", NULL)	Error Code: 1644. Помилка, orders_received: всі обов'язкові поля мають бути заповнені.

Рис. 3.1.10.1 Робота тригеру Before Delete при додаванні даних в таблицю orders_received

3.1.10.2 Before Update (RESTRICT)

DELIMITER //

CREATE TRIGGER orders_received_before_update

BEFORE UPDATE ON orders_received

FOR EACH ROW

BEGIN

-- Перевірка, чи існує вказаний order_id в таблиці orders

IF NOT EXISTS (SELECT 1 FROM orders o WHERE o.order_id = NEW.order_id

AND o.date_returned IS NULL) THEN

```

        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_received: вказаний order_id не існує в
таблиці orders, або вже відгружен Замовнику.';

    END IF;

    -- чи існує вказаний employee_id в таблиці employee
    IF NOT EXISTS (SELECT 1 FROM employees e WHERE e.employee_id =
NEW.employee_id) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_received: вказаний
employee_id не існує в таблиці employees.';

    END IF;

    -- Перевірка, чи існує вказаний brand_id в таблиці brand
    IF NOT EXISTS (SELECT 1 FROM brand b WHERE b.brand_id = NEW.brand_id)
THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_received: вказаний brand_id не існує в
таблиці brand_id.';

    END IF;

    -- Перевірка, чи всі обов'язкові поля заповнені
    IF NEW.model_name IS NULL OR
        NEW.sn IS NULL OR
        NEW.equipment IS NULL OR
        NEW.fault_description IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_received: всі обов'язкові поля мають бути
заповнені.';

    END IF;

END//
```

DELIMITER ;

До оновлення даних

```

3 •  SELECT *
4   FROM orders_received
5   WHERE order_id = 10018;

```

order_id	brand_id	employee_id	model_name	sn	barcode	equipment	fault_description
10018	3	5	1610	SN1610-126	NULL	без кабелю живлення	торохтить під час увімкнення

Після оновлення даних

```

50 00:00:26 UPDATE orders_received SET brand_id = 2, employee_id = 4, model_name = '1710', sn = "SN1710-126", equipment = "з кабелю живлення", fault_description = 'погано друкує' WHERE order_id = 10018 1 row(s) affected Rows matched: 1 Changed: 1

15 •  UPDATE orders_received
16   SET brand_id = 2, employee_id = 4, model_name = '1710', sn = "SN1710-126",
17   equipment = "з кабелю живлення", fault_description = 'погано друкує'
18   WHERE order_id = 10018;

```

order_id	brand_id	employee_id	model_name	sn	barcode	equipment	fault_description
10018	2	4	1710	SN1710-126	NULL	з кабелю живлення	погано друкує

Помилки при оновленні даних, що визвав тригер

- ✖ 9 17:38:35 UPDATE orders_received SET brand_id = 2 WHERE order_id = 10019 Error Code: 1644. Помилка, orders_received: вказаний order_id не існує в таблиці orders, або вже відгружен Замовнику.
- ✖ 10 17:39:06 UPDATE orders_received SET brand_id = 22 WHERE order_id = 10018 Error Code: 1644. Помилка, orders_received: вказаний brand_id не існує в таблиці brand_id.
- ✖ 11 17:39:38 UPDATE orders_received SET employee_id = 44 WHERE order_id = 10018 Error Code: 1644. Помилка, orders_received: вказаний employee_id не існує в таблиці employees.
- ✖ 12 17:40:07 UPDATE orders_received SET employee_id = NULL WHERE order_id = 10018 Error Code: 1644. Помилка, orders_received: вказаний employee_id не існує в таблиці employees.
- ✖ 13 17:40:37 UPDATE orders_received SET model_name = NULL WHERE order_id = 10018 Error Code: 1644. Помилка, orders_received: всі обов'язкові поля мають бути заповнені.
- ✖ 14 17:40:50 UPDATE orders_received SET sn = NULL WHERE order_id = 10018 Error Code: 1644. Помилка, orders_received: всі обов'язкові поля мають бути заповнені.
- ✖ 15 17:40:57 UPDATE orders_received SET equipment = NULL WHERE order_id = 10018 Error Code: 1644. Помилка, orders_received: всі обов'язкові поля мають бути заповнені.
- ✖ 16 17:41:09 UPDATE orders_received SET fault_description = NULL WHERE order_id = 10018 Error Code: 1644. Помилка, orders_received: всі обов'язкові поля мають бути заповнені.

3.1.10.3 Before Delete (RESTRICT)

DELIMITER //

```
CREATE TRIGGER orders_received_before_delete
```

```
BEFORE DELETE ON orders_received
```

```
FOR EACH ROW
```

```
BEGIN
```

```
-- Перевірка, чи існує вказаний order_id в таблиці orders
```

```
IF NOT EXISTS (SELECT 1 FROM orders o WHERE o.order_id = OLD.order_id AND
o.date_returned IS NULL) THEN
```

```
  SIGNAL SQLSTATE '45000'
```

```

SET MESSAGE_TEXT = 'Помилка, orders_received: вказаний order_id не існує в
таблиці orders, або вже відгружен Замовнику.';

END IF;

END//;

DELIMITER ;

```

До видалення даних

order_id	date_returned
10016	NULL
10017	2024-11-25
10018	NULL
10019	2024-11-21
NULL	NULL

order_id	brand_id	employee_id	model_name
10018	2	4	1710
10019	3	5	1610
NULL	NULL	NULL	NULL

Після видалення даних

```

31 18:06:09 DELETE FROM orders_received WHERE order_id = 10018 - brand_id = 22, employee_id = 4, model_name = '1710', sn = "SN1710-126", equipment = "з кабелю живлення", fault_description = "ногано друкує" 1 row(s) affected

```

order_id	brand_id	employee_id	model_name
10019	3	5	1610
NULL	NULL	NULL	NULL

`DELETE FROM orders_received WHERE order_id = 10018`

Помилки при оновленні даних, що визвав триггер

```

42 18:14:19 DELETE FROM orders_received WHERE order_id = 10019 Error Code: 1644. Помилка, orders_received: вказаний order_id не існує в таблиці orders, або вже відгружен Замовнику.

```

Рис. 3.1.10.3 Робота триггеру Before Delete при видалені даних з таблиці `orders_received`

Таблиця `orders_repair`

3.1.11.1 Before Insert (RESTRICT)

DELIMITER //

```

CREATE TRIGGER orders_repair_before_insert
BEFORE INSERT ON orders_repair
FOR EACH ROW
BEGIN

    -- Перевірка, чи існує вказаний order_id в таблиці orders
    IF NOT EXISTS (SELECT 1 FROM orders o WHERE o.order_id = NEW.order_id
    AND o.date_returned IS NULL) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_received: вказаний order_id не існує в
таблиці orders, або вже відгружен Замовнику。';
    END IF;

    -- чи існує вказаний employee_id в таблиці employee
    IF NOT EXISTS (SELECT 1 FROM employees e WHERE e.employee_id =
    NEW.employee_id) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_received: вказаний
employee_id не існує в таблиці employees。';
    END IF;

END//
```

DELIMITER ;

До вставки даних

```

14 •  SELECT order_id,
15      date_returned
16  FROM orders
17 WHERE order_id > 10015;
```

order_id	date_returned
10016	NULL
10017	2024-11-25
10018	NULL
10019	2024-11-21
NULL	NULL

order_id	employee_id	date_ready	price_repair
10019	17	NULL	0
10005	8	2024-09-08	51000
10005	9	2024-09-06	30500

Після вставки даних

56 22:13:00 INSERT INTO orders_repair (order_id, employee_id) VALUES (10018, 15) 1 row(s) affected

```
INSERT INTO orders_repair
(order_id, employee_id)
VALUES (10018, 15);
```

order_id	employee_id	date_ready	price_repair
10019	17	NULL	0
10018	15	NULL	0
10005	8	2024-09-08	51000
10005	9	2024-09-06	30500

Помилки при додаванні даних, що визвав тригер

```
58 22:16:53 INSERT INTO orders_repair (order_id, employee_id) VALUES (10018, 45) Error Code: 1644. Помилка, orders_received: вказаний employee_id не існує в таблиці employees.
59 22:17:34 INSERT INTO orders_repair (order_id, employee_id) VALUES (10017, 15) Error Code: 1644. Помилка, orders_received: вказаний order_id не існує в таблиці orders, або вже відгружен Замовнику.
```

Рис. 3.1.11.1 Робота тригера Before Insert при додаванні даних в таблицю orders_repair

3.1.11.2 Before Update (RESTRICT)

DELIMITER //

DROP TRIGGER orders_repair_before_update;

```
CREATE TRIGGER orders_repair_before_update
BEFORE UPDATE ON orders_repair
FOR EACH ROW
BEGIN
```

```
-- Перевірка, чи існує вказаний order_id в таблиці orders
IF NOT EXISTS (SELECT 1 FROM orders o WHERE o.order_id = NEW.order_id
AND o.date_returned IS NULL) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, orders_repair: вказаний order_id не існує в
таблиці orders, або вже відгружен Замовнику.';
END IF;
```

```
-- чи існує вказаний employee_id в таблиці employee
IF NOT EXISTS (SELECT 1 FROM employees e WHERE e.employee_id =
NEW.employee_id) THEN
    SIGNAL SQLSTATE '45000'
```

```

SET MESSAGE_TEXT = 'Помилка, orders_repair: вказаний
employee_id не існує в таблиці employees.';

END IF;

END//
```

До оновлення даних

14 •	<code>SELECT order_id,</code>	2 •	<code>SELECT order_id, employee_id,</code>																						
15	<code> date_returned</code>	3	<code> date_ready, price_repair</code>																						
16	<code>FROM orders</code>	4	<code>FROM orders_repair</code>																						
17	<code>WHERE order_id > 10016;</code>	5	<code>order by order_id DESC;</code>																						
	Result Grid Filter Rows: <input type="text"/>		Result Grid Filter Rows: <input type="text"/>																						
	<table border="1"> <thead> <tr> <th>order_id</th><th>date_returned</th></tr> </thead> <tbody> <tr> <td>10017</td><td>2024-11-25</td></tr> <tr> <td>10018</td><td>NULL</td></tr> <tr> <td>10019</td><td>2024-11-21</td></tr> <tr> <td>NULL</td><td>NULL</td></tr> </tbody> </table>	order_id	date_returned	10017	2024-11-25	10018	NULL	10019	2024-11-21	NULL	NULL		<table border="1"> <thead> <tr> <th>order_id</th><th>employee_id</th><th>date_ready</th><th>price_repair</th></tr> </thead> <tbody> <tr> <td>10019</td><td>17</td><td>2024-11-30</td><td>73500</td></tr> <tr> <td>10018</td><td>15</td><td>NULL</td><td>0</td></tr> </tbody> </table>	order_id	employee_id	date_ready	price_repair	10019	17	2024-11-30	73500	10018	15	NULL	0
order_id	date_returned																								
10017	2024-11-25																								
10018	NULL																								
10019	2024-11-21																								
NULL	NULL																								
order_id	employee_id	date_ready	price_repair																						
10019	17	2024-11-30	73500																						
10018	15	NULL	0																						

Після оновлення даних

66 22:38:24	<code>UPDATE orders_repair SET date_ready = '2024-11-30', price_repair = 37500, description = 'ремонт картриджу' WHERE order_id = 10018 AND employee_id = 15</code>	1 row(s) affected Rows matched: 1 Changed: 1												
<code>UPDATE orders_repair SET date_ready = '2024-11-30', price_repair = 37500, description = 'ремонт картриджу' WHERE order_id = 10018 AND employee_id = 15;</code>														
		<table border="1"> <thead> <tr> <th>order_id</th><th>employee_id</th><th>date_ready</th><th>price_repair</th></tr> </thead> <tbody> <tr> <td>10019</td><td>17</td><td>2024-11-30</td><td>73500</td></tr> <tr> <td>10018</td><td>15</td><td>2024-11-30</td><td>37500</td></tr> </tbody> </table>	order_id	employee_id	date_ready	price_repair	10019	17	2024-11-30	73500	10018	15	2024-11-30	37500
order_id	employee_id	date_ready	price_repair											
10019	17	2024-11-30	73500											
10018	15	2024-11-30	37500											

Помилки при оновленні даних, які визвав тригер

74 22:47:58	<code>UPDATE orders_repair SET employee_id = 47 WHERE order_id = 10018 AND employee_id = 15</code>	Error Code: 1644. Помилка, orders_repair: вказаний employee_id не існує в таблиці employees.
75 22:48:20	<code>UPDATE orders_repair SET employee_id = 47 WHERE order_id = 10019 AND employee_id = 17</code>	Error Code: 1644. Помилка, orders_repair: вказаний order_id не існує в таблиці orders, або вже відгружен Замовник.

Рис. 3.1.11.2 Робота тригеру Before Update при оновленні даних в таблиці orders_repair

3.1.11.3 Before Delete (RESTRICT)

```

DELIMITER //

DROP TRIGGER orders_repair_before_delete;
```

```

CREATE TRIGGER orders_repair_before_delete
BEFORE DELETE ON orders_repair
FOR EACH ROW
BEGIN

    -- якщо date_ready є, видалити не можна
    IF OLD.date_ready IS NOT NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_repair: роботи завершени, видалити не
можна';
    END IF;

    -- Перевірка, чи існує вказаний order_id в таблиці orders
    IF NOT EXISTS (SELECT 1 FROM orders o WHERE o.order_id = OLD.order_id AND
o.date_returned IS NULL) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, orders_repair: вказаний order_id не існує в
таблиці orders, або вже відгружен Замовнику。';
    END IF;

END//
```

DELIMITER ;

До оновлення даних

<pre> 14 • SELECT order_id, 15 date_returned 16 FROM orders 17 WHERE order_id > 10016; </pre>	<pre> 2 • SELECT order_id, employee_id, 3 date_ready, price_repair 4 FROM orders_repair 5 ORDER BY order_id DESC; </pre>																										
<p>Result Grid Filter Rows: <input type="text"/></p> <table border="1"> <thead> <tr> <th>order_id</th> <th>date_returned</th> </tr> </thead> <tbody> <tr> <td>10017</td> <td>NULL</td> </tr> <tr> <td>10018</td> <td>NULL</td> </tr> <tr> <td>10019</td> <td>2024-11-21</td> </tr> <tr> <td>NULL</td> <td>NULL</td> </tr> </tbody> </table>	order_id	date_returned	10017	NULL	10018	NULL	10019	2024-11-21	NULL	NULL	<p>Result Grid Filter Rows: <input type="text"/> E</p> <table border="1"> <thead> <tr> <th>order_id</th> <th>employee_id</th> <th>date_ready</th> <th>price_repair</th> </tr> </thead> <tbody> <tr> <td>10019</td> <td>17</td> <td>2024-11-30</td> <td>73500</td> </tr> <tr> <td>10018</td> <td>15</td> <td>2024-11-30</td> <td>37500</td> </tr> <tr> <td>10017</td> <td>15</td> <td>NULL</td> <td>0</td> </tr> </tbody> </table>	order_id	employee_id	date_ready	price_repair	10019	17	2024-11-30	73500	10018	15	2024-11-30	37500	10017	15	NULL	0
order_id	date_returned																										
10017	NULL																										
10018	NULL																										
10019	2024-11-21																										
NULL	NULL																										
order_id	employee_id	date_ready	price_repair																								
10019	17	2024-11-30	73500																								
10018	15	2024-11-30	37500																								
10017	15	NULL	0																								

Після оновлення даних

```

| ✓ | 83 23:00:35 DELETE FROM orders_repair WHERE order_id = 10017 AND employee_id = 15      1 row(s) affected

```

Помилки при оновленні даних, що визвав тригер

```

| ✗ | 84 23:01:44 DELETE FROM orders_repair WHERE order_id = 10018 AND employee_id = 15      Error Code: 1644. Помилка, orders_repair: роботи завершени, видалити не можна

```

Рис. 3.1.11.3 Робота тригера Before Delete при видалені даних з таблиці `orders_repair`

Отже, написано 35 тригерів для 11 таблиц, 33 before + 2 after.

Відстежили усі основні зв'язки та перевірки валідності на найбільш цікавих таблицях. Залишилось написати тригери для 7 однотипних та не цікавих таблиц, в яких немає бізнес логіки. Для економії часу я просто наведу код та скриншоти виконання тригерів без скриншотів таблиц.

Таблиця brand

3.1.12.1 Before Insert (RESTRICT)

```

DELIMITER //

CREATE TRIGGER brand_before_insert
BEFORE INSERT ON brand
FOR EACH ROW
BEGIN
    -- Перевірка унікальності name
    IF EXISTS (SELECT 1 FROM brand b WHERE b.name = NEW.name) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Помилка, brand: таке ім'я вже є, ім'я бренду має бути
        унікальним.";
    END IF;
    -- Перевірка чи є description
    IF NEW.description IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, brand: додайте опис бренду';
    END IF;
END//
```

DELIMITER ;

3	11:41:50	INSERT INTO brand VALUES (NULL, 'Panasonic', '')	1 row(s) affected
4	11:42:25	INSERT INTO brand VALUES (NULL, 'Panasonic')	Error Code: 1136. Column count doesn't match value count at row 1
5	11:42:49	INSERT INTO brand VALUES (NULL, 'Panasonic', 'NULL')	Error Code: 1644. Помилка, brand: таке ім'я вже є, ім'я бренду має бути унікальним.
6	11:43:16	INSERT INTO brand VALUES (NULL, 'Kodak', 'NULL')	1 row(s) affected
7	11:43:38	INSERT INTO brand VALUES (NULL, 'Konica', 'NULL')	Error Code: 1644. Помилка, brand: додайте опис бренду

Рис. 3.1.12.1 Робота тригеру Before Insert при додаванні даних в таблицю brand

3.1.12.2 Before Update (RESTRICT)

```
DELIMITER //  
CREATE TRIGGER brand_before_update  
BEFORE UPDATE ON brand  
FOR EACH ROW  
BEGIN  
    -- Перевірка унікальності name  
    IF NEW.name <> OLD.name THEN  
        IF EXISTS (SELECT 1 FROM brand b WHERE b.name = NEW.name)  
        THEN  
            SIGNAL SQLSTATE '45000'  
            SET MESSAGE_TEXT = "Помилка, brand: таке ім'я вже є, ім'я  
брэнду має бути унікальним.";  
        END IF;  
    END IF;  
    -- Перевірка чи є description  
    IF NEW.description IS NULL THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка, brand: додайте опис бренду';  
    END IF;  
  
    -- зміна brand_id не дозволена  
    IF NEW.brand_id <> OLD.brand_id THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Помилка, brand: зміна brand_id не  
дозволена';
```

```

END IF;
END//;
DELIMITER ;

```

✓	12	14:25:43	UPDATE brand SET description = 'photo' WHERE brand_id = 7	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✓	13	14:26:13	UPDATE brand SET name = 'Konica' WHERE brand_id = 7	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✗	14	14:26:27	UPDATE brand SET name = 'Konica' WHERE brand_id = 6	Error Code: 1644. Помилка, brand: таке ім'я вже є, ім'я бренду має бути унікальним.
✓	15	14:27:02	SET SQL_SAFE_UPDATES = 0	0 row(s) affected
✗	16	14:27:09	UPDATE brand SET brand_id = 8 WHERE brand_id = 6	Error Code: 1644. Помилка, brand: зміна brand_id не дозволена
✗	17	14:28:39	UPDATE brand SET description = NULL WHERE brand_id = 6	Error Code: 1644. Помилка, brand: додайте опис бренду

Рис. 3.1.12.2 Робота тригеру Before Update при оновленні даних в таблиці

brand

3.1.11.3 Before Delete (RESTRICT)

```

DELIMITER //
CREATE TRIGGER brand_before_delete
BEFORE DELETE ON brand
FOR EACH ROW
BEGIN
    -- Перевірка наявності пов'язаних записів у таблиці orders_received
    IF (SELECT COUNT(*) FROM orders_received WHERE brand_id = OLD.brand_id) > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, brand: Неможливо видалити запис, оскільки існують пов'язані записи в таблиці orders_received.';
    END IF;
END//;
DELIMITER ;

```

✓	20	14:46:40	DELETE FROM brand WHERE brand_id = 7	1 row(s) affected
✗	21	14:46:48	DELETE FROM brand WHERE brand_id = 1	Error Code: 1644. Помилка, brand: Неможливо видалити запис, оскільки існують пов'язані записи в таблиці orders_received.

Рис. 3.1.12.3 Робота тригеру Before Delete при видалені даних з таблиці brand

Таблиця city

3.1.13.1 Before Insert (RESTRICT)

```
DELIMITER //
CREATE TRIGGER city_before_insert
BEFORE INSERT ON city
FOR EACH ROW
BEGIN
    -- Перевірка унікальності name
    IF EXISTS (SELECT 1 FROM city b WHERE b.name = NEW.name) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Помилка, city: таке ім'я вже є, ім'я міста має бути
унікальним.";
    END IF;
END//
DELIMITER ;
```

✗	28	15:05:11	INSERT INTO city(name) VALUES ('Харків')	Error Code: 1644. Помилка, city: таке ім'я вже є, ім'я міста має бути унікальним.
✓	29	15:05:24	INSERT INTO city(name) VALUES ('Суми')	1 row(s) affected

Рис. 3.1.13.1 Робота тригера Before Insert при додаванні даних в таблицю city

3.1.13.2 Before Update (RESTRICT)

```
DELIMITER //
CREATE TRIGGER city_before_update
BEFORE UPDATE ON city
FOR EACH ROW
BEGIN
    -- Перевірка унікальності name
    IF NEW.name <> OLD.name THEN
        IF EXISTS (SELECT 1 FROM city b WHERE b.name = NEW.name) THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = "Помилка, city: таке ім'я вже є, ім'я
міста має бути унікальним.";
        END IF;
    END IF;
END IF;
```

```

-- зміна city_id не дозволена
IF NEW.city_id <> OLD.city_id THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, city: зміна city_id не дозволена';
END IF;
END//;
DELIMITER ;

```

✖ 40	15:17:21	UPDATE city SET city_id = 10 WHERE city_id = 2	Error Code: 1644. Помилка, city: зміна city_id не дозволена
✖ 41	15:17:42	UPDATE city SET name = 'Суми' WHERE city_id = 2	Error Code: 1644. Помилка, city: таке ім'я вже є, ім'я міста має бути унікальним.
✓ 42	15:17:55	UPDATE city SET name = 'Ровно' WHERE city_id = 2	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

Рис. 3.1.13.2 Робота тригера Before Update при оновленні даних в таблиці city

3.1.13.3 Before Delete (RESTRICT)

```

DELIMITER //
CREATE TRIGGER city_before_delete
BEFORE DELETE ON city
FOR EACH ROW
BEGIN
    -- Перевірка наявності пов'язаних записів у таблиці customers
    IF (SELECT COUNT(*) FROM customers WHERE city_id = OLD.city_id) > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, city: Неможливо видалити запис, оскільки
існують пов'язані записи в таблиці customers.';
    END IF;
END//;
DELIMITER ;

```

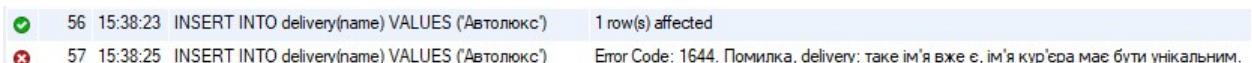
✖ 45	15:24:41	DELETE FROM city WHERE city_id = 1	Error Code: 1644. Помилка, city: Неможливо видалити запис, оскільки існують пов'язані записи в таблиці customers.
✓ 46	15:24:52	DELETE FROM city WHERE city_id = 12	1 row(s) affected

Рис. 3.1.13.3 Робота тригера Before Delete при видалені даних з таблиці city

Таблиця delivery

3.1.14.1 Before Insert (RESTRICT)

```
DELIMITER //
CREATE TRIGGER delivery_before_insert
BEFORE INSERT ON delivery
FOR EACH ROW
BEGIN
    -- Перевірка унікальності name
    IF EXISTS (SELECT 1 FROM delivery b WHERE b.name = NEW.name) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Помилка, delivery: таке ім'я вже є, ім'я
кур'єра має бути унікальним.";
    END IF;
END//;
DELIMITER ;
```



56	15:38:23	INSERT INTO delivery(name) VALUES ('Автолюкс')	1 row(s) affected
57	15:38:25	INSERT INTO delivery(name) VALUES ('Автолюкс')	Error Code: 1644. Помилка, delivery: таке ім'я вже є, ім'я кур'єра має бути унікальним.

Рис. 3.1.14.1 Робота тригера Before Insert при додаванні даних в таблицю delivery

3.1.14.2 Before Update (RESTRICT)

```
DELIMITER //
CREATE TRIGGER delivery_before_update
BEFORE UPDATE ON delivery
FOR EACH ROW
BEGIN
    -- Перевірка унікальності name
    IF NEW.name <> OLD.name THEN
        IF EXISTS (SELECT 1 FROM delivery b WHERE b.name = NEW.name)
        THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = "Помилка, delivery: таке ім'я вже є,
ім'я міста має бути унікальним.";
        END IF;
    END IF;
```

```

END IF;

-- зміна delivery_id не дозволена

IF NEW.delivery_id <> OLD.delivery_id THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, delivery: зміна delivery_id не
дозволена';

END IF;

END//;

DELIMITER ;

```

60	15:42:28	UPDATE delivery SET name = 'Нічний експрес' WHERE delivery_id = 4	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
61	15:42:35	UPDATE delivery SET name = 'Нічний експрес' WHERE delivery_id = 4	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
62	15:42:53	UPDATE delivery SET name = 'Нічний експрес' WHERE delivery_id = 3	Error Code: 1644. Помилка, delivery: таке ім'я вже є, ім'я міста має бути унікальним.
63	15:43:35	UPDATE delivery SET delivery_id = 5 WHERE delivery_id = 3	Error Code: 1644. Помилка, delivery: зміна delivery_id не дозволена

Рис. 3.1.14.2 Робота триггеру Before Update при оновленні даних в таблиці delivery

3.1.14.3 Before Delete (RESTRICT)

```

DELIMITER //

CREATE TRIGGER delivery_before_delete
BEFORE DELETE ON delivery
FOR EACH ROW
BEGIN
    -- Перевірка наявності пов'язаних записів у таблиці orders
    IF (SELECT COUNT(*) FROM orders WHERE delivery_id = OLD.delivery_id) > 0
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, delivery: Неможливо видалити запис, оскільки
існують пов'язані записи в таблиці orders.';

    END IF;
END//;

DELIMITER ;

```

66	15:48:14	DELETE FROM delivery WHERE delivery_id = 4	1 row(s) affected
67	15:48:23	DELETE FROM delivery WHERE delivery_id = 1	Error Code: 1644. Помилка, delivery: Неможливо видалити запис, оскільки існують пов'язані записи в таблиці orders.

Рис. 3.1.14.3 Робота тригеру Before Delete при видалені даних з таблиці delivery

Таблиця group

3.1.15.1 Before Insert (RESTRICT)

```
DELIMITER //
CREATE TRIGGER group_before_insert
BEFORE INSERT ON `group`
FOR EACH ROW
BEGIN
    -- Перевірка унікальності name
    IF EXISTS (SELECT 1 FROM `group` b WHERE b.name = NEW.name) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Помилка, group: таке ім'я вже є, ім'я групи
має бути унікальним.";
    END IF;
END//
DELIMITER ;
```

71	16:22:24	INSERT INTO `group`(`name`) VALUES ('supplies Panasonic')	1 row(s) affected
72	16:22:29	INSERT INTO `group`(`name`) VALUES ('supplies Panasonic')	Error Code: 1644. Помилка, group: таке ім'я вже є, ім'я групи має бути унікальним.

Рис. 3.1.15.1 Робота тригеру Before Insert при додаванні даних в таблицю group

3.1.15.2 Before Update (RESTRICT)

```
DELIMITER //
CREATE TRIGGER group_before_update
BEFORE UPDATE ON `group`
FOR EACH ROW
BEGIN
    -- Перевірка унікальності name
    IF NEW.name <> OLD.name THEN
```

```

        IF EXISTS (SELECT 1 FROM `group` b WHERE b.name = NEW.name)
THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Помилка, group: таке ім'я вже є, ім'я
группи має бути унікальним.";
    END IF;
END IF;

-- зміна delivery_id не дозволена
IF NEW.group_id <> OLD.group_id THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, group: зміна group_id не
дозволена';
END IF;

END//  

DELIMITER ;

```

77	16:27:31	UPDATE `group` SET name = 'spares Panasonic' WHERE group_id = 15	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
78	16:28:00	UPDATE `group` SET name = 'spares Panasonic' WHERE group_id = 14	Error Code: 1644. Помилка, group: таке ім'я вже є, ім'я групи має бути унікальним.
79	16:28:20	UPDATE `group` SET group_id = 16 WHERE group_id = 14	Error Code: 1644. Помилка, group: зміна group_id не дозволена

Рис. 3.1.15.2 Робота тригеру Before Update при оновленні даних в таблиці
group

3.1.15.3 Before Delete (RESTRICT)

```

DELIMITER //
CREATE TRIGGER group_before_delete
BEFORE DELETE ON `group`
FOR EACH ROW
BEGIN
    -- Перевірка наявності пов'язаних записів у таблиці goods_dscr
    IF (SELECT COUNT(*) FROM goods_dscr WHERE group_id = OLD.group_id) > 0
THEN
    SIGNAL SQLSTATE '45000'

```

```

        SET MESSAGE_TEXT = 'Помилка, group: Неможливо видалити запис, оскільки
існують пов'язані записи в таблиці goods_dscr.';

    END IF;

END//
```

DELIMITER ;

82	16:52:38	DELETE FROM `group` WHERE group_id = 15	1 row(s) affected
83	16:52:44	DELETE FROM `group` WHERE group_id = 10	Error Code: 1644. Помилка, group: Неможливо видалити запис, оскільки існують пов'язані записи в таблиці goods_dscr.

Рис. 3.1.15.3 Робота тригера Before Delete при видалені даних з таблиці group

Таблиця position

3.1.16.1 Before Insert (RESTRICT)

```

DELIMITER //

CREATE TRIGGER position_before_insert
BEFORE INSERT ON position
FOR EACH ROW
BEGIN
    -- Перевірка унікальності name
    IF EXISTS (SELECT 1 FROM position b WHERE b.name = NEW.name) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Помилка, position: таке ім'я вже є, ім'я групи
має бути унікальним.";
    END IF;
END//
```

DELIMITER ;

87	17:02:09	INSERT INTO position (name) VALUES ('заправщик')	1 row(s) affected
88	17:02:14	INSERT INTO position (name) VALUES ('заправщик')	Error Code: 1644. Помилка, position: таке ім'я вже є, ім'я групи має бути унікальним.

Рис. 3.1.16.1 Робота тригера Before Insert при додаванні даних в таблицю position

3.1.16.2 Before Update (RESTRICT)

```

DELIMITER //
CREATE TRIGGER position_before_update
BEFORE UPDATE ON position
FOR EACH ROW
BEGIN
    -- Перевірка унікальності name
    IF NEW.name <> OLD.name THEN
        IF EXISTS (SELECT 1 FROM position b WHERE b.name = NEW.name)
        THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = "Помилка, position: таке ім'я вже є,
ім'я посади має бути унікальним.";
        END IF;
    END IF;
    -- зміна delivery_id не дозволена
    IF NEW.position_id <> OLD.position_id THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, position: зміна position_id не
дозволена';
    END IF;
END//
```

DELIMITER ;

✓	91	17:04:56	UPDATE position SET name = 'заправщик' WHERE position_id = 7	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✗	92	17:05:12	UPDATE position SET name = 'заправщик' WHERE position_id = 6	Error Code: 1644. Помилка, position: таке ім'я вже є, ім'я посади має бути унікальним.
✗	93	17:05:26	UPDATE position SET position_id = 8 WHERE position_id = 6	Error Code: 1644. Помилка, position: зміна position_id не дозволена

Рис. 3.1.16.2 Робота тригеру Before Update при оновленні даних в таблиці
position

3.1.16.3 Before Delete (RESTRICT)

```

DELIMITER //
CREATE TRIGGER position_before_delete
BEFORE DELETE ON position
FOR EACH ROW
```

```

BEGIN
    -- Перевірка наявності пов'язаних записів у таблиці employees
    IF (SELECT COUNT(*) FROM employees WHERE position_id = OLD.position_id) >
0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, position: Неможливо видалити запис, оскільки
існують пов'язані записи в таблиці employees.';
    END IF;
END//  

DELIMITER ;

```

 97 17:10:36 DELETE FROM position WHERE position_id = 7 1 row(s) affected
 98 17:10:42 DELETE FROM position WHERE position_id = 5 Error Code: 1644. Помилка, position: Неможливо видалити запис, оскільки існують пов'язані записи в таблиці employees.

Рис. 3.1.16.3 Робота тригера Before Delete при видалені даних з таблиці position

Таблиця salary

3.1.17.1 Before Insert (RESTRICT)

```

DELIMITER //
CREATE TRIGGER salary_before_insert
BEFORE INSERT ON salary
FOR EACH ROW
BEGIN
    -- Перевірка, чи всі обов'язкові поля заповнені
    IF NEW.employee_id IS NULL OR
        NEW.accrued_year IS NULL OR
        NEW.accrued_month IS NULL OR
        NEW.accrued_sum IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, salary: всі обов'язкові поля мають бути заповнені。';
    END IF;
    -- Перевіряємо, чи існує вже запис з такими самими ключовими значеннями
    IF EXISTS (

```

```

SELECT 1
FROM salary
WHERE employee_id = NEW.employee_id
AND accrued_year = NEW.accrued_year
AND accrued_month = NEW.accrued_month
) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, salary: запис із такими значеннями вже існує。';
END IF;
-- Встановлення початкових значень
SET NEW.ESV = NEW.accrued_sum * 0.22;
SET NEW.PDFO = NEW.accrued_sum * 0.18;
SET NEW.VS = NEW.accrued_sum * 0.05;
SET NEW.payable_in_cash = NEW.accrued_sum - NEW.PDFO - NEW.VS;
END///
DELIMITER ;

```

```

✓ 101 20:20:21 INSERT INTO salary (employee_id, accrued_year, accrued_month, accrued_sum) VALUES (1, 2024, 11, 17500) 1 row(s) affected
✗ 102 20:24:02 INSERT INTO salary (employee_id, accrued_year, accrued_month) VALUES (1, 2024, 11) Error Code: 1644. Помилка, salary: запис із такими значеннями вже існує。
✗ 109 20:47:19 INSERT INTO salary (employee_id, accrued_year, accrued_month) VALUES (3, 2024, 11) Error Code: 1644. Помилка, salary: всі обов'язкові поля мають бути заповнені.

```

Рис. 3.1.17.1 Робота тригера Before Insert при додаванні даних в таблицю

salary

3.1.17.2 Before Update (RESTRICT)

```

DELIMITER //
CREATE TRIGGER salary_before_update
BEFORE UPDATE ON salary
FOR EACH ROW
BEGIN
    -- Встановлення початкових значень
    SET NEW.ESV = NEW.accrued_sum * 0.22;
    SET NEW.PDFO = NEW.accrued_sum * 0.18;
    SET NEW.VS = NEW.accrued_sum * 0.05;
    SET NEW.payable_in_cash = NEW.accrued_sum - NEW.PDFO - NEW.VS;

```

```
END//  
DELIMITER ;
```

The screenshot shows the MySQL Workbench interface. At the top, there is a log of three events:

- 107 20:45:40 UPDATE salary SET accrued_sum = 20000 WHERE employee_id = 2 AND accrued_year = 2024 AND accrued_month = 11 1 row(s) affected Rows matched: 1 Changed: 1
- 108 20:46:16 UPDATE salary SET accrued_sum = 20000 WHERE employee_id = 3 AND accrued_year = 2024 AND accrued_month = 11 0 row(s) affected Rows matched: 0 Changed: 0
- 110 20:50:02 UPDATE salary SET ESV = 1 WHERE employee_id = 2 AND accrued_year = 2024 AND accrued_month = 11 1 row(s) affected Rows matched: 1 Changed: 1

Below the log, a query is shown:

```
5 •  SELECT * FROM salary WHERE accrued_month = 11;
```

The result grid shows the following data:

employee_id	accrued_year	accrued_month	accrued_sum	ESV	PDFO	VS	payable_in_cash
1	2024	11	17500	3850	3150	875	13475
2	2024	11	20000	4400	3600	1000	15400
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рис. 3.1.17.2 Робота тригера Before Update при оновленні даних в таблиці salary

3.1.17.3 Before Delete (RESTRICT)

```
DELIMITER //
```

```
DROP TRIGGER salary_before_delete;
```

```
CREATE TRIGGER salary_before_delete
BEFORE DELETE ON salary
FOR EACH ROW
BEGIN
    -- Перевіряємо, чи існує вже запис з такими самими ключовими значеннями
    IF EXISTS (
        SELECT 1
        FROM salary
        WHERE employee_id = OLD.employee_id
        AND accrued_year = OLD.accrued_year
        AND accrued_month = OLD.accrued_month
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, salary: видалення заборонено。';
    END IF;
```

```
END//  
DELIMITER ;
```

118 21:14:41	DELETE FROM salary WHERE employee_id = 2 AND accrued_year = 2024 AND accrued_month = 11	Error Code: 1644. Помилка, salary: видалення заборонено.
119 21:14:50	DELETE FROM salary WHERE employee_id = 3 AND accrued_year = 2024 AND accrued_month = 11	0 row(s) affected

Рис. 3.1.17.3 Робота тригеру Before Delete при видалені даних з таблиці salary

Таблиця status

3.1.18.1 Before Insert (RESTRICT)

```
DELIMITER //  
CREATE TRIGGER status_before_insert  
BEFORE INSERT ON status  
FOR EACH ROW  
BEGIN  
    -- Перевірка унікальності name  
    IF EXISTS (SELECT 1 FROM status b WHERE b.name = NEW.name) THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = "Помилка, status: таке ім'я вже є, ім'я статусу має бути  
унікальним."  
    END IF;  
END//  
DELIMITER ;
```

123 21:23:59	INSERT INTO status (name) VALUES ('other')	1 row(s) affected
124 21:24:13	INSERT INTO status (name) VALUES ('dir')	1 row(s) affected
125 21:25:06	SELECT * FROM status LIMIT 0, 10	10 row(s) returned
126 21:25:31	INSERT INTO status (name) VALUES ('buh')	Error Code: 1644. Помилка, status: таке ім'я вже є, ім'я статусу має бути унікальним.

Рис. 3.1.18.1 Робота тригеру Before Insert при додаванні даних в таблицю
status

3.1.18.2 Before Update (RESTRICT)

```
DELIMITER //  
CREATE TRIGGER status_before_update
```

```

BEFORE UPDATE ON status
FOR EACH ROW
BEGIN
    -- Перевірка унікальності name
    IF NEW.name <> OLD.name THEN
        IF EXISTS (SELECT 1 FROM status WHERE name = NEW.name) THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = "Помилка, status: таке ім'я вже є, ім'я статусу
має бути унікальним.";
        END IF;
    END IF;
    -- зміна status_id не дозволена
    IF NEW.status_id <> OLD.status_id THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, status: зміна status_id не дозволена';
    END IF;
END///
DELIMITER ;

```

✖ 129 21:30:34 UPDATE status SET name = 'director' WHERE status_id = 10	Error Code: 1644. Помилка, status: таке ім'я вже є, ім'я статусу має бути унікальним.
✓ 130 21:31:50 UPDATE status SET name = 'secretary' WHERE status_id = 10	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✖ 131 21:32:15 UPDATE status SET status_id = 11 WHERE status_id = 10	Error Code: 1644. Помилка, status: зміна status_id не дозволена

Рис. 3.1.18.2 Робота триггеру Before Update при оновленні даних в таблиці

status

3.1.18.3 Before Delete (RESTRICT)

```

DELIMITER //
CREATE TRIGGER status_before_delete
BEFORE DELETE ON status
FOR EACH ROW
BEGIN
    -- Перевірка наявності пов'язаних записів у таблиці employees
    IF (SELECT COUNT(*) FROM employees WHERE status_id = OLD.status_id) > 0
    THEN

```

```

    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, status: Неможливо видалити запис, оскільки існують
пов'язані записи в таблиці employees.';

END IF;
-- Перевірка наявності пов'язаних записів у таблиці customers
IF (SELECT COUNT(*) FROM customers WHERE status_id = OLD.status_id) > 0
THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Помилка, status: Неможливо видалити запис, оскільки існують
пов'язані записи в таблиці customers.';

END IF;
END//  

DELIMITER ;

```

✖ 141 21:40:01 DELETE FROM status WHERE status_id = 10 Error Code: 1644. Помилка, status: Неможливо видалити запис, оскільки існують пов'язані записи в таблиці customers.

✓ 142 21:40:10 DELETE FROM status WHERE status_id = 11 1 row(s) affected

Рис. 3.1.18.3 Робота тригера Before Delete при видалені даних з таблиці status

Таблиця unit

3.1.19.1 Before Insert (RESTRICT)

```

DELIMITER //
CREATE TRIGGER unit_before_insert
BEFORE INSERT ON unit
FOR EACH ROW
BEGIN
    -- Перевірка унікальності name
    IF EXISTS (SELECT 1 FROM unit WHERE name = NEW.name) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Помилка, unit: таке ім'я вже є, ім'я одиниці
        вимірювання має бути унікальним.";
    END IF;
END//
```

```
DELIMITER ;
```

145 22:16:35	INSERT INTO unit (name) VALUES ('р')	1 row(s) affected
146 22:16:45	INSERT INTO unit (name) VALUES ('шт')	Error Code: 1644. Помилка, unit: таке ім'я вже є, ім'я одиниці вимірювання має бути унікальним.

Рис. 3.1.19.1 Робота тригера Before Insert при додаванні даних в таблицю unit

3.1.19.2 Before Update (RESTRICT)

```
DELIMITER //
```

```
CREATE TRIGGER unit_before_update
```

```
BEFORE UPDATE ON unit
```

```
FOR EACH ROW
```

```
BEGIN
```

```
-- Перевірка унікальності name
```

```
IF NEW.name <> OLD.name THEN
```

```
    IF EXISTS (SELECT 1 FROM unit WHERE name = NEW.name) THEN
```

```
        SIGNAL SQLSTATE '45000'
```

```
        SET MESSAGE_TEXT = "Помилка, unit: таке ім'я вже є, ім'я одиниці вимірювання має бути унікальним.";
```

```
    END IF;
```

```
END IF;
```

```
-- зміна unit_id не дозволена
```

```
IF NEW.unit_id <> OLD.unit_id THEN
```

```
    SIGNAL SQLSTATE '45000'
```

```
    SET MESSAGE_TEXT = 'Помилка, unit: зміна unit_id не дозволена';
```

```
END IF;
```

```
END//
```

```
DELIMITER ;
```

149 22:20:28	UPDATE unit SET name = "аркуш" WHERE unit_id = 7	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
150 22:20:56	UPDATE unit SET name = "шт" WHERE unit_id = 7	Error Code: 1644. Помилка, unit: таке ім'я вже є, ім'я одиниці вимірювання має бути унікальним.
151 22:21:11	UPDATE unit SET unit_id = 11 WHERE unit_id = 7	Error Code: 1644. Помилка, unit: зміна unit_id не дозволена

Рис. 3.1.19.2 Робота тригера Before Update при оновленні даних в таблиці unit

3.1.19.3 Before Delete (RESTRICT)

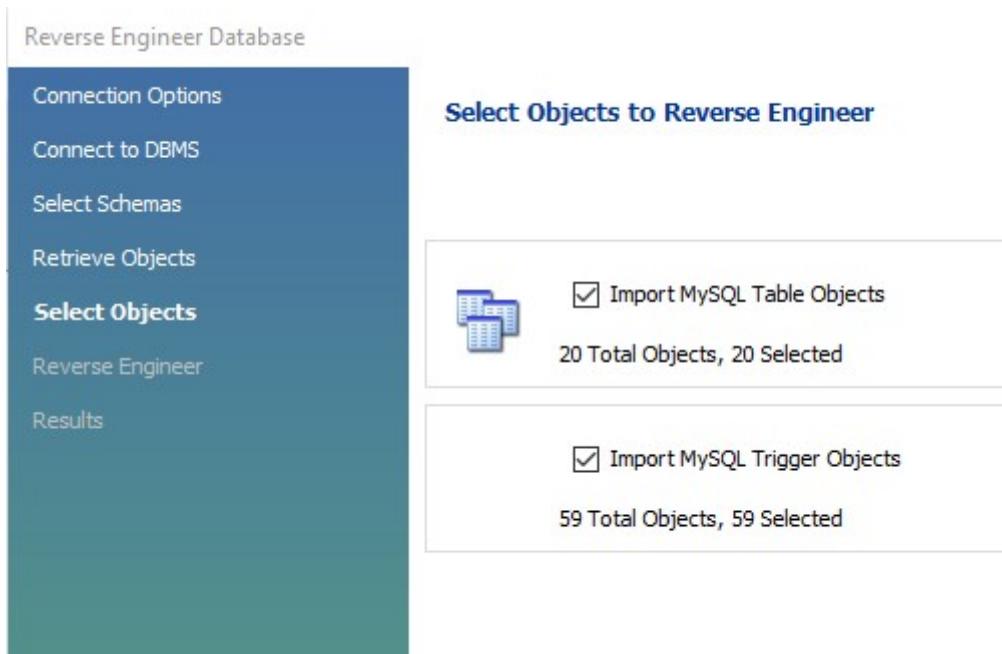
```
DELIMITER //
CREATE TRIGGER unit_before_delete
BEFORE DELETE ON unit
FOR EACH ROW
BEGIN
    -- Перевірка наявності пов'язаних записів у таблиці goods_dscr
    IF (SELECT COUNT(*) FROM goods_dscr WHERE unit_id = OLD.unit_id) > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, unit: Неможливо видалити запис, оскільки існують пов'язані
записи в таблиці goods_dscr.';
    END IF;
END//;
DELIMITER ;
```

```
154 22:24:51 DELETE FROM unit WHERE unit_id = 8 1 row(s) affected
155 22:25:01 DELETE FROM unit WHERE unit_id = 5 Error Code: 1644. Помилка, unit: Неможливо видалити запис, оскільки існують пов'язані записи в таблиці goods_dscr.
```

Рис. 3.1.19.3 Робота тригера Before Delete при видаленні даних з таблиці unit

УСЬОГО зроблено 59 тригерів для 19 таблиц

Схема Reverse Engineer БД з таблицями MyISAM



Reverse Engineer Database

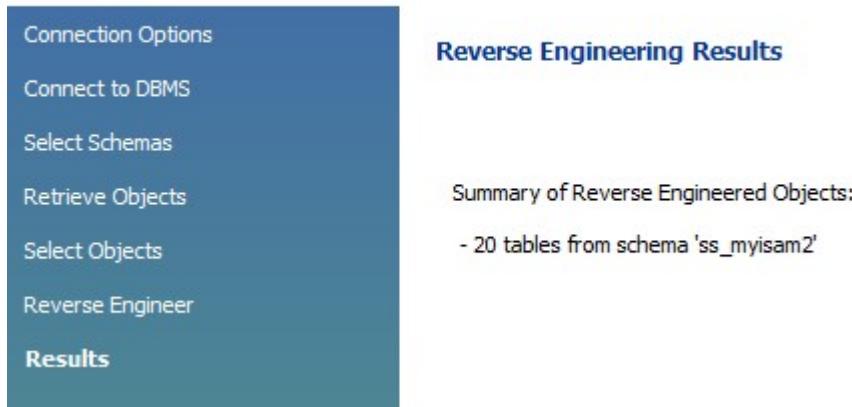


Рис. 3.1.20.1 Скриншоти отримання схеми БД

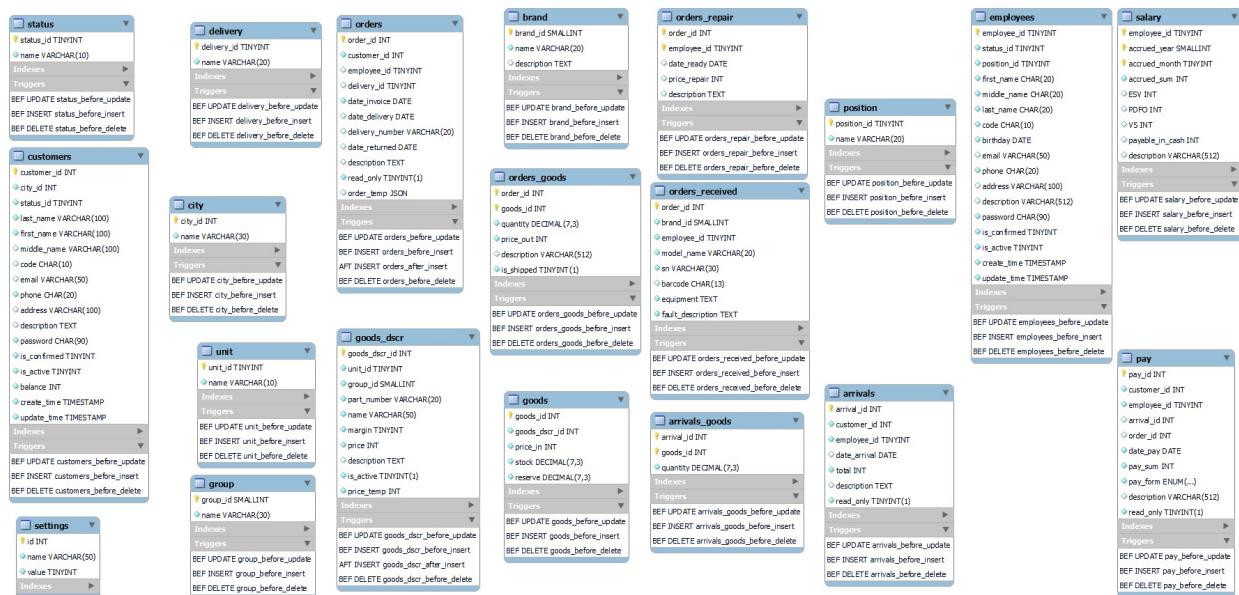


Рис. 3.1.20.2 Схема БД

Завдання 3.2. Для бази даних з таблицями типу InnoDB, створеної відповідно до завдання 1.3 практичного заняття № 1, розробити тригери, що забезпечують цілісність зв'язків, для операцій, не підтримуваних механізмом посилальної цілісності СУБД MySQL. Для всіх тригерів реалізувати скасування операції у випадку невиконання умов цілісності зв'язків, з генерацією повідомлення за допомогою речення SIGNAL.

Оскільки ми порівнюємо дві БД з різним движком (engine = InnoDB/MyISAM) правильно буде сконфігурувати зв'язки однаково. Тому перепишемо таблицю 3.3 для БД InnoDB з урахуванням того, що частина зв'язків реалізована за допомогою SQL- інструкцій, а частина - за допомогою тригерів.

Таблиця 3.4 – Перелік типів посилальної цілісності БД InnoDB

№	Ім'я таблиці 1, зовнішній ключ	Ім'я таблиці 2, первинний ключ	SQL інструкція для таблиці 1	Тип посилальної цілісності	Тригер або SQL-інструкція
1	customers, status_id	status, status_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
2	customers, city_id	city, city_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
3	goods_dscr, unit_id	unit, unit_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
4	goods_dscr, group_id	group, group_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
5	goods, goods_dscr_id	goods_dscr, goods_dscr_id	Update	CASCADE	SQL
			Delete	CASCADE	SQL
			Before Insert	CASCADE	Тригер
			After Insert		
6	orders_goods, goods_id	goods, goods_id	Update	CASCADE	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
7	orders_goods, order_id	orders, order_id	Update	CASCADE	SQL
			Delete	CASCADE	Тригер
			Before Insert	RESTRICT	Тригер

Таблиця 3.4, продовження

8	orders_repair, order_id	orders, order_id	Update	CASCADE	SQL
			Delete	RESTRICT	SQL
			Before Insert	CASCADE	Тригер
			After Insert		
9	orders_repair, employee_id	employees, employee_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
10	orders_received, order_id	orders, order_id	Update	CASCADE	SQL
			Delete	RESTRICT	SQL
			Before Insert	CASCADE	Тригер
11	orders_received, brand_id	brand, brand_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
12	orders_received, employee_id	employees, employee_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
13	orders, employee_id	employees, employee_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
14	orders, delivery_id	delivery, delivery_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
15	orders, customer_id	customers, customer_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
16	salary, employee_id	employees, employee_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
17	pay, employee_id	employees, employee_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер

Таблиця 3.4, продовження

18	pay, customer_id	customers, customer_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
19	pay, arrival_id	arrivals, arrival_id	Update	CASCADE	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
20	pay, order_id	orders, order_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
21	arrivals, employee_id	employees, employee_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
22	arrivals, customer_id	customers, customer_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
23	arrival_goods, arrival_id	arrivals, arrival_id	Update	CASCADE	SQL
			Delete	CASCADE	SQL
			Before Insert	RESTRICT	Тригер
24	arrival_goods, good_id	goods, good_id	Update	CASCADE	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
25	employees, status_id	status, status_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер
26	employees, position_id	position, position_id	Update	RESTRICT	SQL
			Delete	RESTRICT	SQL
			Before Insert	RESTRICT	Тригер

Я не бачу потреби повторно розглядати вже розглянуті тригери, краще зосередимось на реалізації бізнес-логики, деякі тригери модернізуємо. До того ж мені не зрозуміло навіщо для таблиць InnoDB взагалі використовувати

тригери, окрім як для бізнес-логіки. Зв'язки і цілісність даних успішно реалізуються SQL інструкціями, навіть для операцій вставки Insert Restrict. Для ілюстрації вищесказаного наводжу скріншоти вставки даних в таблиці salary і arrivals_goods. Якщо дані не задовольняють вимоги цілісності даних, виникає помилка. Тоді навіщо тут тригер?

```
3 16:54:36 INSERT INTO salary(employee_id, accrued_year, accrued_month, accrued_sum) values (1, 2024, 11, 15000) 1 row(s) affected
4 16:54:41 INSERT INTO salary(employee_id, accrued_year, accrued_month, accrued_sum) values (1, 2024, 11, 15000) Error Code: 1062. Duplicate entry '1-2024-11' for key 'salary.PRIMARY'
5 16:57:10 INSERT INTO salary(employee_id, accrued_year, accrued_month) values (2, 2024, 11) Error Code: 1364. Field 'accrued_sum' doesn't have a default value
12 18:06:30 INSERT INTO arrivals_goods VALUES (1100, 2, 10) Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('ss_innodb`.`arrivals_goods', CONSTRAINT `fk_arrivals_goods_arrivals` FOREIGN KEY (`arrival_id`) REFERENCES `arrivals` (`arrival_id`))
13 18:06:56 INSERT INTO arrivals_goods VALUES (1000, 22000, 10) Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('ss_innodb`.`arrivals_goods`, CONSTRAINT `fk_arrivals_goods_goods` FOREIGN KEY (goods_id) REFERENCES `goods` (goods_id))
```

Рис. 3.2.1 Скриншоти вставки даних у таблиці salary і arrivals_goods

Завдання 3.3. Для баз даних з таблицями MyISAM і InnoDB розробити або модернізувати тригери, що реалізують програмну логіку інтерфейсу бази даних (бізнес-функцій), на які впливає специфіка високонавантажених систем. Під час прийняття рішень на розробку врахувати результати виконання п.2.3.9 завдання 2.2.

Розглянемо кілька бізнес функцій і їх реалізацію в тригерах.

3.3.1. Приход запчастин, задіяні таблиці arrivals, arrivals_goods та тригера до них. При приході запчастин потрібни такі дані:

customer_id, employee_id, total – таблиця arrivals,
arrival_id, goods_id, quantity – таблиця arrivals_goods,
а також name, price_in для обрання потрібного товара, або додавання нового.

Бізнес функція, що реалізується в тригері перевіряє суму заприходованих запчастин з загальною сумою(total) накладної і якщо вони співпадають, то тригер дозволяє проставити дату накладної. Якщо не співпадають – видає помилку і не дозволяє проставити дату. Ця функція потрібна щоб уникнути помилок ручного введення даних.

Код тригера BEFORE UPDATE для таблиці arrivals

```

DELIMITER //

CREATE TRIGGER arrivals_before_update
BEFORE UPDATE ON arrivals
FOR EACH ROW
BEGIN

    DECLARE message VARCHAR(200);
    DECLARE total_arrivals_goods INT;

    -- Перевірка, чи read_only = 1
    IF OLD.read_only = 1 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка, arrivals: цей приход тільки до читання, правити не
можна';
    END IF;

    -- Перевірка, чи date_arrival is not null
    IF NEW.date_arrival IS NOT NULL THEN
        IF NOT EXISTS (SELECT 1 FROM arrivals_goods ag WHERE
ag.arrival_id = NEW.arrival_id) THEN
            SET message = CONCAT('Помилка, arrivals: приход не введен, виправьте, а
потім введіть дату накладної');
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
        ELSE
            SELECT ROUND(SUM(price_in*quantity)) INTO
total_arrivals_goods
                FROM arrivals a
                JOIN arrivals_goods ag USING(arrival_id)
                JOIN goods g USING(goods_id)
                WHERE a.arrival_id = NEW.arrival_id
                GROUP BY arrival_id;
            IF total_arrivals_goods <> NEW.total THEN
                SET message = CONCAT('Помилка, arrivals: сума накладної не
співпадає з сумою запчастин, виправьте, а потім введіть дату накладної');
            END IF;
        END IF;
    END IF;
END

```

```
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
message;

ELSE -- дата приходу не старше як 2 тижня

    IF NEW.date_arrival <> OLD.date_arrival AND (
        DATEDIFF(CURDATE(), NEW.date_arrival) > 15 OR
        DATEDIFF(NEW.date_arrival, CURDATE()) > 0)

THEN

    SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Помилка, arrivals: вказана
date_arrival не допустима.';

    END IF;

    END IF;

END IF;

ELSE

    IF NEW.read_only = 1 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Помилка, arrivals: цей приход ще не отриман, не можна ставити
тільки читання';

    END IF;

    END IF;

END//
```

До оновлення дати приходу

```
14 •   SELECT arrival_id,
15           date_arrival, total
16   FROM arrivals
17  WHERE arrival_id = 1006;
--
```

Result Grid | Filter Rows:

arrival_id	date_arrival	total
1006	NULL	46280
NULL	NULL	NULL


```
2 •   SELECT arrival_id, goods_id,
3           quantity, price_in
4   FROM arrivals_goods
5   JOIN goods USING(goods_id)
6  WHERE arrival_id = 1006;
```

Result Grid | Filter Rows:

arrival_id	goods_id	quantity	price_in
1006	2	10.000	1446


```
24 •  SELECT goods_id,
25           stock, price_in
26   FROM goods
27  WHERE goods_id IN (2,3);
--
```

Result Grid | Filter Rows:

goods_id	stock	price_in
2	22.000	1446
3	21.000	3182

Після оновлення дати виникає помилка – суми не співпадають

10 00:19:58 UPDATE arrivals SET date_arrival = '2024-11-24' WHERE arrival_id = 1006 Error Code: 1644. Помилка, arrivals: сума накладної не співпадає з сумою запчастин, виправте, а

Оновлення дати після додавання запчастини в приход - суми співпадають

arrival_id	date_arrival	total
1006	2024-11-24	46280
NULL	NULL	NULL

arrival_id	goods_id	quantity	price_in
1006	2	10.000	1446
1006	3	10.000	3182

goods_id	stock	price_in
2	22.000	1446
3	31.000	3182
NULL	NULL	NULL

Рис. 3.3.1 Робота тригера Before Update при оновленні даних в таблиці arrivals

Ще одна бізнес функція при приході запчастин – оновлення складу, яка реалізується тригерами BEFORE Insert/Update для таблиці arrivals_goods. Робота цього тригера також продемонстрована вище на скриншоті. В цих тригерах в SELECT запиті для запобігання конкурентного оновлення даних будем використовувати додаткову інструкцію FOR UPDATE.

Код тригера BEFORE UPDATE для таблиці arrivals_goods

DELIMITER //

CREATE TRIGGER arrivals_goods_before_update

BEFORE UPDATE ON arrivals_goods

FOR EACH ROW

BEGIN

DECLARE old_stock_old_goods decimal(7,3);

DECLARE old_stock_new_goods decimal(7,3);

DECLARE message VARCHAR(200);

-- Перевірка, чи arrivals.date_arrival is null

IF NOT EXISTS (SELECT 1 FROM arrivals a WHERE a.arrival_id = NEW.arrival_id
AND a.date_arrival IS NULL) THEN

SET message = CONCAT(

'Помилка, arrivals_goods: вказаний arrival_id = ', NEW.arrival_id, ' не існує в таблиці arrivals або приход
вже зроблений, оновити не можливо.');

SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;

END IF;

-- Считування та оновлення stock, FOR UPDATE - блокування рядка для запобігання конкурентного оновлення

```

IF NEW.goods_id <> OLD.goods_id THEN
    SELECT stock INTO old_stock_old_goods FROM goods WHERE
    goods_id = OLD.goods_id FOR UPDATE;
    SELECT stock INTO old_stock_new_goods FROM goods WHERE
    goods_id = NEW.goods_id FOR UPDATE;
    UPDATE goods SET stock = old_stock_old_goods - OLD.quantity
    WHERE goods_id = OLD.goods_id;
    UPDATE goods SET stock = old_stock_new_goods + NEW.quantity WHERE
    goods_id = NEW.goods_id;
ELSE
    -- NEW.goods_id = OLD.goods_id
    IF NEW.quantity <> OLD.quantity THEN
        SELECT stock INTO old_stock_new_goods FROM goods WHERE
        goods_id = NEW.goods_id FOR UPDATE;
        UPDATE goods SET stock = old_stock_new_goods + NEW.quantity -
        OLD.quantity WHERE goods_id = NEW.goods_id;
    END IF;
END IF;
END//  

DELIMITER ;

```

Ще одна бізнес-функція, що є в приходе, це видалення застарілих даних. Ця функція охоплює також і Замовлення, і Оплати. Ця функція з однієї сторони є небеспечною з точки зору збереження даних, а з другої - бажано мати просте рішення цього питання. Тому я виніс дозвіл на видалення в окрему таблицю settings. Він виглядає як деактивація тригеру на видалення. Якщо в звичайних умовах тригер не дозволяє видалення, то встановив потрібний прапорець в означеній вище таблиці можемо тимчасово деактивувати тригер, тим самим дозволить видалення неактуальних даних. Доступ на редагування таблиці settings обмежен, щоб уберечити дані.

Таблиці arrivals, arrivals_goods між собою мають тип посилальної цілісності On Delete Cascade. Тому при видаленні даних з таблиці arrivals

також будуть видалені і зв'язані записи з таблиці arrivals_goods. При такому каскадному видаленні тригер таблиці arrivals_goods не задіюється та це видалення складу не торкнеться. Це треба враховувати.

Код тригера BEFORE DELETE для таблиці arrivals

```
DROP DELIMITER //  
CREATE TRIGGER arrivals_before_delete  
BEFORE DELETE ON arrivals  
FOR EACH ROW  
BEGIN  
  
    DECLARE triggers_enabled TINYINT;  
  
    -- Перевірка пропорція triggers_enable в таблиці налаштувань  
    SELECT `value` INTO triggers_enabled  
    FROM settings  
    WHERE name = 'triggers_enable';  
  
    -- Якщо тригери увімкнені, продовжуємо, якщо вимкнені - пропускаємо  
    IF triggers_enabled = 1 THEN  
  
        -- Перевірка, чи приход отриман  
        IF OLD.date_arrival IS NOT NULL THEN  
            SIGNAL SQLSTATE '45000'  
            SET MESSAGE_TEXT = 'Помилка: приход отриман, неможливо  
            видалити.';  
        END IF;  
  
        -- Перевірка, чи є пов'язаний запис у таблиці pay  
        IF EXISTS (SELECT 1 FROM pay WHERE arrival_id = OLD.arrival_id)  
        THEN  
            SIGNAL SQLSTATE '45000'  
            SET MESSAGE_TEXT = "Помилка: неможливо видалити, оскільки  
            існує пов'язаний запис у таблиці pay.";  
        ELSE
```

```

        DELETE FROM arrivals_goods WHERE arrival_id =
OLD.arrival_id;
        END IF;
    END IF;
END//  

DELIMITER ;

```

Проведемо тест цієї функції пізніше, одночасно з Замовленнями та Оплатами.

3.3.2. Замовлення

На 36-й сторінке ми вже розглянули одну бізнес-функцію по одночасній вставці даних в таблиці `orders`, `orders_received` та `orders_repair`, а також роботу тригера `Insert Cascade`.

Розглянемо ще декілька бізнес-функцій.

Підрахунок загальної суми замовлення. Додамо поле `total` до таблиці `orders`, в якому будемо зберігати суму замовлення. Код для підрахунку суми додамо до тригерів таблиць `orders_repair` та `orders_goods`

Код тригера AFTER UPDATE ON orders_repair (тригер таблиці `orders_goods` аналогічний)

```

DELIMITER //
CREATE TRIGGER orders_repair_after_update
AFTER UPDATE ON orders_repair
FOR EACH ROW
BEGIN
    DECLARE order_total INT;
    -- сума замовлення
    SELECT ROUND(COALESCE(SUM(og.price_out * og.quantity), 0) +
COALESCE(SUM(price_repair), 0)) INTO order_total
    FROM orders o
    LEFT JOIN orders_goods og ON o.order_id = og.order_id

```

```

    LEFT JOIN orders_repair orp ON o.order_id = orp.order_id
    WHERE o.order_id = NEW.order_id;

    UPDATE orders SET total = order_total WHERE order_id = NEW.order_id;
END//  

DELIMITER ;

```

До оновлення даних

```

1 •   SELECT order_id, employee_id,
2                  price_repair
3 •   SELECT order_id, total
4      FROM orders
5      WHERE order_id in (9999, 10005);
6      WHERE order_id in (9999, 10005);
```

order_id	employee_id	price_repair
9999	12	135343
10005	8	51000
10005	9	30500
NULL	NULL	NULL

```

9 •   SELECT order_id, goods_id,
10                 quantity, price_out
11      FROM orders_goods
12      WHERE order_id in (9999, 10005);
```

order_id	goods_id	quantity	price_out
9999	2794	1.009	5204
NULL	NULL	NULL	NULL

```

UPDATE orders_repair
SET price_repair = 140000
WHERE order_id = 9999
AND employee_id = 12;
```

```

UPDATE orders_repair
SET price_repair = 37000
WHERE order_id = 10005
AND employee_id = 9;
```

Після оновлення

- 21 22:35:31 UPDATE orders_repair SET price_repair = 37000 WHERE order_id = 10005 AND employee_id = 9 1 row(s) affected Rows matched: 1
- 22 22:37:17 UPDATE orders_repair SET price_repair = 140000 WHERE order_id = 9999 AND employee_id = 12 1 row(s) affected Rows matched: 1
- 28 23:06:09 UPDATE orders_goods SET quantity = 1, price_out = 5000 WHERE order_id = 9999 AND goods_id = 2794 1 row(s) affected Rows matched: 1 Changed: 1

order_id	employee_id	price_repair
9999	12	140000
10005	8	51000
10005	9	37000
NULL	NULL	NULL

order_id	goods_id	quantity	price_out
9999	2794	1.000	5000
NULL	NULL	NULL	NULL

Рис. 3.3.2 Робота тригера AFTER UPDATE при оновленні даних в таблиці
orders_repair

Наступна бізнес функція, Замовлення – оновлення складу

Код тригера BEFORE INSERT ON orders_goods

```

DELIMITER //

CREATE TRIGGER orders_goods_before_insert
BEFORE INSERT ON orders_goods
FOR EACH ROW
BEGIN

    DECLARE new_price_out INT;
    DECLARE old_stock decimal(7,3);
    DECLARE old_reserve decimal(7,3);
    DECLARE message VARCHAR(200);

    -- Встановлення price_out, считування stock, reserve
    SELECT price INTO new_price_out FROM goods_dscr JOIN goods USING
(goods_dscr_id) WHERE goods_id = NEW.goods_id;
    SET NEW.price_out = new_price_out;

    SELECT stock INTO old_stock FROM goods WHERE goods_id =
NEW.goods_id FOR UPDATE;
    SELECT reserve INTO old_reserve FROM goods WHERE goods_id =
NEW.goods_id FOR UPDATE;

    -- Чи є потрібна кількість
    IF NEW.quantity IS NULL OR NEW.quantity > old_stock - old_reserve THEN
        SET message = CONCAT('Помилка, orders_goods: запчастина з goods_id = ',
NEW.goods_id, ': такої кількості немає в наявності');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
    ELSE
        -- Якщо кількість є, оновлюємо склад
        IF NEW.is_shipped = 0 THEN
            UPDATE goods SET reserve = old_reserve + NEW.quantity
WHERE goods_id = NEW.goods_id;
        ELSE
            UPDATE goods SET stock = old_stock - NEW.quantity WHERE
goods_id = NEW.goods_id;
        END IF;
    END IF;
END;

```

```

END IF;
END IF;
END//;
DELIMITER ;

```

До вставки

```

.9 •  SELECT order_id, goods_id,
.10   quantity, price_out, is_shipped
.11   FROM orders_goods
.12   WHERE order_id in (9999);

4 •  SELECT goods_id,
5   stock, price_in
6   FROM goods
7   WHERE goods_id IN (2,3);

sult Grid | Filter Rows: [ ] | Edit: [ ]
order_id goods_id quantity price_out is_shipped
9999 2794 1.000 5000 1
NULL NULL NULL NULL NULL

sult Grid | Filter Rows: [ ] | Edit: [ ]
goods_id stock price_in
2 22.000 1446
3 31.000 3182
NULL NULL NULL

4 •  SELECT order_id, total
5   FROM orders
6   WHERE order_id in (9999);

sult Grid | Filter Rows: [ ] | Edit: [ ]
order_id total
9999 145000
NULL NULL

INSERT INTO orders_goods
(order_id, goods_id, quantity,
is_shipped)
VALUES (9999, 2, 1, 1),
(9999, 3, 2, 0);

```

Після вставки

```

38 23:51:43 INSERT INTO orders_goods (order_id, goods_id, quantity, is_shipped) VALUES (9999, 2, 1, 1), (9999, 3, 2, 0) 2 row(s) affected Records: 2

| order_id | goods_id | quantity | price_out | is_shipped | goods_id | stock | reserve | price_in | order_id | total |
| 9999 | 2 | 1.000 | 1880 | 1 | 2 | 21.000 | 3.000 | 1446 | 9999 | 435154 |
| 9999 | 3 | 2.000 | 4137 | 0 | 3 | 31.000 | 3.000 | 3182 | NULL | NULL |
| 9999 | 2794 | 1.000 | 5000 | 1 | NULL | NULL | NULL | NULL | NULL | NULL |

```

Рис. 3.3.3 Робота тригера Before Insert при додаванні даних в таблицю orders_goods

Отримали цікаві результати, тобто склад триггер перераховує вірно, а вот триггер after insert видав НЕ вірний результат та в таблицю orders pole total записана не вірна сума, котра равна $3 * \text{repair_total} + \text{orders_goods_total}$. Так вийшло з-за того, що у замовленні 9999 списано 3 запчастини. Щоб виправити цю помилку треба підраховувати суми в окремих запросах, а не в одному як зараз.

Виправлений код тригера AFTER INSERT ON orders_goods

```
DELIMITER //

CREATE TRIGGER orders_goods_after_insert
AFTER INSERT ON orders_goods
FOR EACH ROW
BEGIN

    DECLARE order_total INT;

    -- сума замовлення
    SELECT ROUND(COALESCE(SUM(og.price_out * og.quantity), 0) +
    COALESCE(MAX(repairs.total_repair), 0)) INTO order_total

    FROM orders o
    LEFT JOIN orders_goods og ON o.order_id = og.order_id
    LEFT JOIN (
        SELECT order_id, SUM(price_repair) AS total_repair
        FROM orders_repair
        WHERE order_id = NEW.order_id
        GROUP BY order_id
    ) AS repairs ON o.order_id = repairs.order_id
    WHERE o.order_id = NEW.order_id;

    UPDATE orders SET total = order_total WHERE order_id = NEW.order_id;


```

```
UPDATE orders SET total = order_total WHERE order_id = NEW.order_id;
```

```
END//
```

```
DELIMITER ;
```

Повторимо експеримент - до вставки

order_id	goods_id	quantity	price_out	is_shipped
9999	2	1.000	1880	1
9999	3	2.000	4137	0
9999	2794	1.000	5000	1
NULL	NULL	NULL	NULL	NULL

goods_id	stock	reserve	price_in
4	25.000	2.000	3604
6	22.000	0.000	7404
NULL	NULL	NULL	NULL

```
INSERT INTO orders_goods
(order_id, goods_id,
quantity, is_shipped)
VALUES (9999, 4, 3, 0),
(9999, 6, 3, 1);
```

Після вставки

```
16 18:51:45 INSERT INTO orders_goods (order_id, goods_id, quantity, is_shipped) VALUES (9999, 4, 3, 0), (9999, 6, 3, 1) 2 row(s) affected Records: 2
```

order_id	goods_id	quantity	price_out	is_shipped	goods_id	stock	reserve	price_in	order_id	total
9999	2	1.000	1880	1	4	25.000	5.000	3604	9999	198090
9999	3	2.000	4137	0	6	19.000	0.000	7404	NULL	NULL
9999	4	3.000	4686	0						
9999	6	3.000	9626	1						
9999	2794	1.000	5000	1						
NULL	NULL	NULL	NULL	NULL						

Рис. 3.3.4. Робота тригеру Before Insert при додаванні даних в таблицю orders_goods. Тепер усе чудово працює!

Замовлення на продаж запчастин, що виписано, але не відвантажено. Бізнес-функція каскадного видалення. Тут видалення потрібно організувати не SQL інструкцієй, а тригером, щоб запустився процес корегування резерву запчастин. Крім того, використаємо код відключення тригеру BEFORE DELETE ON orders, який ми уже розглядали раніше, щоб в подальшему ми змогли скористатися функцією видалення застарілих даних.

```

CREATE TRIGGER orders_before_delete
BEFORE DELETE ON orders
FOR EACH ROW
BEGIN
    DECLARE triggers_enabled TINYINT;
    -- Перевірка праворядка triggers_enabled в таблиці налаштувань
    SELECT `value` INTO triggers_enabled
    FROM settings
    WHERE name = 'triggers_enable';
    -- Якщо тригери увімкнені, продовжуємо, якщо вимкнені - пропускаємо
    IF triggers_enabled = 1 THEN

        -- Перевірка, чи замовлення відгружене
        IF OLD.date_returned IS NOT NULL THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка: замовлення відгружене, неможливо видалити。';
        END IF;

        -- Перевірка, чи є пов'язана запис у таблиці orders_received
        IF EXISTS (SELECT 1 FROM orders_received WHERE order_id = OLD.order_id) THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = "Помилка: неможливо видалити, оскільки існує пов'язана запис у таблиці orders_received。";
        END IF;

        -- Перевірка, чи є пов'язаний запис у таблиці pay
        IF EXISTS (SELECT 1 FROM pay WHERE order_id = OLD.order_id) THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = "Помилка: неможливо видалити, оскільки існує пов'язана запис у таблиці pay。";
        END IF;

        -- Якщо немає записів з OLD.order_id у таблиці orders_goods, де is_shipped = 1, то
        -- видаляємо записи з таблиці orders_goods, пов'язані з orders по order_id
        IF (SELECT COUNT(*) FROM orders_goods WHERE order_id = OLD.order_id AND is_shipped = 1) = 0 THEN
            DELETE FROM orders_goods WHERE order_id = OLD.order_id;
        ELSE
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка: неможливо видалити, оскільки існують відвантажені запчастини у таблиці orders_goods。';
        END IF;
    END IF;
END//
```

DELIMITER ;

Рис. 3.3.5 Скриншот коду тригеру BEFORE DELETE ON orders

```

DELIMITER //
DROP TRIGGER orders_goods_before_delete;

CREATE TRIGGER orders_goods_before_delete
BEFORE DELETE ON orders_goods
FOR EACH ROW
BEGIN

DECLARE old_reserve_old_goods decimal(7,3);
DECLARE message VARCHAR(200);

-- Перевірка, чи запчастина відгруженна
IF OLD.is_shipped = 1 THEN
    SET message = CONCAT(
        'Помилка, orders_goods: вказаний goods_id = ', OLD.goods_id, ' вже відвантажено, не можливо видалити.');
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
END IF;

-- Перевірка, чи вказаний order_id в таблиці orders відвантажен
IF NOT EXISTS (SELECT 1 FROM orders o WHERE o.order_id = OLD.order_id AND o.date_returned IS NULL) THEN
    SET message = CONCAT(
        'Помилка, orders_goods: вказаний order_id = ', OLD.order_id, ' вже відвантажено, не можливо видалити.');
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
END IF;

-- Якщо запис у таблиці orders_goods, має is_shipped = 0, а також orders.date_returned IS NULL
-- можемо видалити цю запис з таблиці orders_goods, перерахуємо резерв
SELECT reserve INTO old_reserve_old_goods FROM goods WHERE goods_id = OLD.goods_id FOR UPDATE;
UPDATE goods
SET reserve = old_reserve_old_goods - OLD.quantity
WHERE goods_id = OLD.goods_id;

END//
```

Рис. 3.3.6 Скриншот коду тригера BEFORE DELETE ON orders_goods

До видалення

```
4 • SELECT order_id, date_returned
5   FROM orders
6 WHERE order_id = 10014;
```

Result Grid			
order_id	goods_id	quantity	is_shipped
10014	1	1.000	0
10014	2	1.000	0
10014	4	2.000	0
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL

```
19 • SELECT order_id, goods_id,
20   quantity, is_shipped
21   FROM orders_goods
22 WHERE order_id = 10014;
```

Result Grid		
goods_id	stock	reserve
1	14.000	6.000
2	21.000	3.000
4	25.000	5.000
NULL	NULL	NULL

```
4 • SELECT goods_id, stock,
5   reserve
6   FROM goods
7 WHERE goods_id IN (1,2,4);
```

Після видалення

41 21:16:29 DELETE FROM orders WHERE order_id = 10014 1 row(s) affected

order_id	date_returned	order_id	goods_id	quantity	is_shipped	goods_id	stock	reserve
NULL	NULL	NULL	NULL	NULL	NULL	1	14.000	5.000
						2	21.000	2.000
						4	25.000	3.000

Рис. 3.3.7. Робота тригеру Before Delete при видалені даних з таблиць orders, orders_goods - Все працює як треба!

Перевіримо роботу тригерів, що забезпечують бізнес-функцію каскадного видалення приходу запчастин, що не оплачено та не заприходовано повністю (date_arrival is null).

```

DELIMITER //
DROP TRIGGER arrivals_before_delete;

CREATE TRIGGER arrivals_before_delete
BEFORE DELETE ON arrivals
FOR EACH ROW
BEGIN

DECLARE triggers_enabled TINYINT;

-- Перевірка прапорця triggers_enable в таблиці налаштувань
SELECT `value` INTO triggers_enabled
FROM settings
WHERE name = 'triggers_enable';

-- Якщо тригери увімкнені, продовжуємо, якщо вимкнені - пропускаємо
IF triggers_enabled = 1 THEN

    -- Перевірка, чи приход отриман
    IF OLD.date_arrival IS NOT NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: приход отриман, неможливо видалити.';
    END IF;

    -- Перевірка, чи є пов'язаний запис у таблиці pay
    IF EXISTS (SELECT 1 FROM pay WHERE arrival_id = OLD.arrival_id) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Помилка: неможливо видалити, оскільки існує пов'язаний запис у таблиці pay.";
    ELSE
        -- видаляємо записи з таблиці arrivals_goods каскадно
        DELETE FROM arrivals_goods WHERE arrival_id = OLD.arrival_id;
    END IF;
END IF;
END//
```

DELIMITER ;

Рис. 3.3.8. Скриншот коду тригеру BEFORE DELETE ON arrivals

```

DELIMITER //
DROP TRIGGER arrivals_goods_before_delete;

CREATE TRIGGER arrivals_goods_before_delete
BEFORE DELETE ON arrivals_goods
FOR EACH ROW
BEGIN

    DECLARE old_stock_old_goods decimal(7,3);
    DECLARE message VARCHAR(200);

    -- Перевірка, чи проставлена дата приходу
    IF NOT EXISTS (SELECT 1 FROM arrivals a WHERE a.arrival_id = OLD.arrival_id AND a.date_arrival IS NULL) THEN
        SET message = CONCAT(
            'Помилка, arrivals_goods: вказаний arrival_id = ', OLD.arrival_id, ' не існує або оприходован, видалити не можна.');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
    END IF;

    -- Считування та оновлення stock
    SELECT stock INTO old_stock_old_goods FROM goods WHERE goods_id = OLD.goods_id FOR UPDATE;
    UPDATE goods SET stock = old_stock_old_goods - OLD.quantity WHERE goods_id = OLD.goods_id;

END//
DELIMITER ;

```

Рис. 3.3.9. Скриншот коду тригера BEFORE DELETE ON arrivals_goods

До видалення

```

4 •  SELECT arrival_id,
5      date_arrival
6  FROM arrivals
7 WHERE arrival_id = 1007;

```

```

4 •  SELECT *
5  FROM arrivals_goods
6 WHERE arrival_id = 1007;

```

```

4 •  SELECT goods_id, stock
5  FROM goods
6 WHERE goods_id IN (2,3);

```

arrival_id	date_arrival
1007	NULL
NULL	NULL

arrival_id	goods_id	quantity
1007	2	10.000
1007	3	10.000
NULL	NULL	NULL

goods_id	stock
2	31.000
3	41.000
NULL	NULL

Після каскадного видалення

54 22:52:10 DELETE FROM arrivals WHERE arrival_id = 1007 1 row(s) affected

arrival_id	date_arrival
NULL	NULL

arrival_id	goods_id	quantity
NULL	NULL	NULL

goods_id	stock
2	21.000
3	31.000
NULL	NULL

Рис. 3.3.10 Робота тригера Before Delete при каскадному видалені даних з таблиць arrivals, arrivals_goods - Все добре працює!

Розглянемо таку бизнес-функція, як каскадне видалення запчастини з таблиць goods_dscr та goods за допомогою SQL інструкції - зв'язка ON DELETE CASCADE

```

DELIMITER //
DROP TRIGGER goods_dscr_before_delete;

CREATE TRIGGER goods_dscr_before_delete
BEFORE DELETE ON goods_dscr
FOR EACH ROW
BEGIN

    -- Перевірка, чи запчастина є активною
    IF OLD.is_active <> 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Помилка: запчастина є активної, неможливо видалити.';
    END IF;

    -- Якщо є пов'язані записи у таблиці goods, продовжуємо перевірку
    IF (SELECT COUNT(*) FROM goods g WHERE g.goods_dscr_id = OLD.goods_dscr_id) > 0 THEN

        -- Перевірка наявності пов'язаних записів у таблицях orders_goods та arrivals_goods за goods_id
        IF (SELECT COUNT(*) FROM orders_goods WHERE goods_id IN (SELECT goods_id FROM goods WHERE goods_dscr_id = OLD.goods_dscr_id)) > 0 THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка: неможливо видалити, оскільки існують пов'язані записи в таблиці orders_goods.';
        END IF;

        IF (SELECT COUNT(*) FROM arrivals_goods WHERE goods_id IN (SELECT goods_id FROM goods WHERE goods_dscr_id = OLD.goods_dscr_id)) > 0 THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Помилка: неможливо видалити, оскільки існують пов'язані записи в таблиці arrivals_goods.';
        END IF;

        -- Якщо пов'язаних записів немає, видаляємо записи з таблиці goods каскадно
    END IF;
END//
```

Рис. 3.3.11. Скриншот коду тригера BEFORE DELETE ON goods_dscr

До видалення

```

5 •  SELECT goods_dscr_id, name,
6          is_active
7  FROM goods_dscr
8  WHERE goods_dscr_id = 10001

```

goods_dscr_id	name	is_active
10001	Товар 4XAWK	0
NULL	NULL	NULL

```

4 •  SELECT goods_id, goods_dscr_id,
5          stock, reserve
6  FROM goods
7  WHERE goods_dscr_id = 10001;

```

goods_id	goods_dscr_id	stock	reserve
10001	10001	0.000	0.000
NULL	NULL	NULL	NULL

Після видалення

65 23:24:17 DELETE FROM goods_dscr WHERE goods_dscr_id = 10001
1 row(s) affected

goods_dscr_id	name	is_active
NULL	NULL	NULL

goods_id	goods_dscr_id	stock	reserve
NULL	NULL	NULL	NULL

Помилки, що визвав тригер

```
✖ 74 23:32:11 DELETE FROM goods_dscr WHERE goods_dscr_id = 998      Error Code: 1644. Помилка: запчастина є активної, неможливо видалити.  
✖ 75 23:32:18 DELETE FROM goods_dscr WHERE goods_dscr_id = 997      Error Code: 1644. Помилка: неможливо видалити, оскільки існують пов'язані записи в таблиці orders_goods.
```

Рис. 3.3.12. Робота тригера Before Delete при видалені даних з таблиць
goods, goods_dscr

Остання бизнес-функція це підрахунок зарплати робитникам та
утриманих податків з неї.

```
DELIMITER //  
DROP TRIGGER salary_before_insert;  
  
CREATE TRIGGER salary_before_insert  
BEFORE INSERT ON salary  
FOR EACH ROW  
BEGIN  
  
    -- Перевірка року та місяця  
    IF NEW.accrued_year IS NULL THEN  
        SET NEW.accrued_year = YEAR(CURDATE());  
    END IF;  
  
    IF NEW.accrued_month IS NULL THEN  
        SET NEW.accrued_month = MONTH(CURDATE());  
    END IF;  
  
    -- Обчислення податків  
    SET NEW.ESV = NEW.accrued_sum * 0.22;  
    SET NEW.PDFO = NEW.accrued_sum * 0.18;  
    SET NEW.VS = NEW.accrued_sum * 0.05;  
  
    -- чиста зарплата "на руки"  
    SET NEW.payable_in_cash = NEW.accrued_sum - NEW.PDFO - NEW.VS;  
  
END//  
DELIMITER ;
```

Рис. 3.3.13. Скриншот коду тригера BEFORE INSERT ON salary

Демонстрація роботи тригера

```
✓ 9 17:13:56 INSERT INTO salary(employee_id, accrued_sum) values (1, 15000) 1 row(s) affected
```

5 • `SELECT * FROM salary WHERE accrued_month = 12;`

Result Grid							
employee_id	accrued_year	accrued_month	accrued_sum	ESV	PDFO	VS	payable_in_cash
1 NULL	2024 NULL	12 NULL	15000 NULL	3300 NULL	2700 NULL	750 NULL	11550 NULL

Рис. 3.3.14. Робота триггеру Before Insert при додаванні даних в таблицю salary

На останок, подивимось один з варіантів як можливо організувати видалення застарілих даних з використанням конструкцій, що були розглянуті раніше.

Функція передбачає видалення даних старіших за 3 роки з таблиць замовлень, приходов та оплат. Всього 7 таблиць.

```

1      -- Видалення застарілих даних (більш як 3 роки).
2 •  START TRANSACTION;
3 •  SET SQL_SAFE_UPDATES = 0;  -- Вимкнути безпечний режим
4 •  UPDATE settings SET `value` = 0 WHERE name = 'triggers_enable'; -- Вимкнути тригери
5
6 •  DELETE FROM orders
7      WHERE date_returned between '2020-01-01' and '2020-12-31';
8
9 •  DELETE FROM arrivals
10     WHERE date_arrival between '2020-01-01' and '2020-12-31';
11
12 •  UPDATE settings SET `value` = 1 WHERE name = 'triggers_enable'; -- Увімкнути тригери
13 •  SET SQL_SAFE_UPDATES = 1;  -- Включити безпечний режим назад
14 •  COMMIT;

```

Рис. 3.3.15. Скриншот коду, що видаляє застарілі дані

До видалення

```

4 •  SELECT order_id, customer_id,
5          date_returned, total
6      FROM orders
7      WHERE date_returned
8      between '2020-01-01' and '2020-12-01';

```

order_id	customer_id	date_returned	total
10017	1	2020-12-06	62000
10018	1	2020-12-06	14312
NULL	NULL	NULL	NULL

```

6 •  SELECT *
7      FROM orders_received
8      WHERE order_id IN (10017, 10018);

```

order_id	brand_id	employee_id	model_name	sn
10017	3	5	1610	SN1
NULL	NULL	NULL	NULL	NULL

```

4 •  SELECT arrival_id, customer_id,
5          date_arrival, total
6      FROM arrivals
7      WHERE arrival_id = 1008;

```

arrival_id	customer_id	date_arrival	total
1008	1	2020-12-05	4628
NULL	NULL	NULL	NULL

```

0 •  SELECT pay_id, customer_id,
1          order_id, arrival_id,
2          date_pay, pay_sum
3      FROM pay
4      WHERE pay_id > 11002;

```

pay_id	customer_id	order_id	arrival_id	date_pay	pay_sum
11005	1	NULL	1008	2020-12-06	4628
11006	1	10017	NULL	2020-12-06	62000
11007	1	10018	NULL	2020-12-06	14312

```

20 •  SELECT order_id, goods_id,
21          quantity, is_shipped
22      FROM orders_goods
23      WHERE order_id IN (10017, 10018);

```

order_id	goods_id	quantity	is_shipped
10018	4	1.000	1
10018	6	1.000	1
NULL	NULL	NULL	NULL

```

1 •  SELECT order_id, employee_id,
2          price_repair, date_ready
3      FROM orders_repair
4      WHERE order_id IN (10017, 10018);

```

order_id	employee_id	price_repair	date_ready
10017	17	62000	2020-12-06
NULL	NULL	NULL	NULL

```

0 •  SELECT *
1      FROM arrivals_goods
2      WHERE arrival_id = 1008;

```

arrival_id	goods_id	quantity
1008	2	1.000
1008	3	1.000

```

25 •  SELECT goods_id, stock, reserve
26      FROM goods
27      WHERE goods_id IN (1,2,4,6);

```

goods_id	stock	reserve
1	14.000	5.000
2	22.000	2.000
4	24.000	3.000
6	18.000	0.000

Після видалення

34 20:03:54 START TRANSACTION	0 row(s) affected
35 20:03:54 SET SQL_SAFE_UPDATES = 0	0 row(s) affected
36 20:03:54 UPDATE settings SET 'value' = 0 WHERE name = 'triggers_enable'	1 row(s) affected Rows matched: 1 Changed: 1
37 20:03:54 DELETE FROM orders WHERE date_returned between '2020-01-01' and '2020-12-31'	2 row(s) affected
38 20:03:54 DELETE FROM arrivals WHERE date_arrival between '2020-01-01' and '2020-12-31'	2 row(s) affected
39 20:03:54 UPDATE settings SET 'value' = 1 WHERE name = 'triggers_enable'	1 row(s) affected Rows matched: 1 Changed: 1
40 20:03:54 SET SQL_SAFE_UPDATES = 1	0 row(s) affected
41 20:03:54 COMMIT	0 row(s) affected

order_id	customer_id	date_returned	total
HULL	NULL	NULL	NULL

order_id	goods_id	quantity	is_shipped
NULL	NULL	NULL	NULL

order_id	brand_id	employee_id	model_name	sn
NULL	NULL	NULL	NULL	NULL

order_id	employee_id	price_repair	date_ready
NULL	NULL	NULL	NULL

arrival_id	customer_id	date_arrival	total
NULL	NULL	NULL	NULL

arrival_id	goods_id	quantity
NULL	NULL	NULL

goods_id	stock	reserve
1	14.000	5.000
2	22.000	2.000
4	24.000	3.000
6	18.000	0.000
HULL	NULL	NULL

pay_id	custc	order_id	arrival	date_pay	pay_su
NULL	NULL	NULL	NULL	NULL	NULL

Рис. 3.3.16 Робота функція що видаляє застарілі дані з таблиць замовлень, приходів та оплат.

Наш код отработав коректно, застарілі дані 7-ми таблиць були видалені, стан складу не змінився. Чудово!

Завдання 3.4. Провести аналіз розроблених SQL-запитів, що використовуються в тригерах, за допомогою оператора EXPLAIN. Оцінити план виконання кожного SQL-запиту з висновком «неможливо оптимізувати» або «вимагає оптимізації». Підготувати короткі пропозиції з коректування коду SQL-запитів, схеми зв'язків, типів даних для зменшення часу запиту.

3.4.1. Проаналізуємо запит, що підраховує загальну суму замовлення

```

47 • explain
48 SELECT sql_no_cache ROUND(COALESCE(SUM(og.price_out * og.quantity), 0) + COALESCE(MAX(repairs.total_repair), 0)) AS order_total
49 FROM orders o
50 LEFT JOIN orders_goods og ON o.order_id = og.order_id
51 LEFT JOIN (
52     SELECT order_id, SUM(price_repair) AS total_repair
53     FROM orders_repair
54     WHERE order_id = 9999
55     GROUP BY order_id
56 ) AS repairs ON o.order_id = repairs.order_id
57 WHERE o.order_id = 9999;
58
59

```

Result Grid | Filter Rows: [] | Export: [] | Wrap Cell Content: []

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	o	NULL	const	PRIMARY	PRIMARY	4	const	1	100.00	Using index
1	PRIMARY	og	NULL	ref	PRIMARY	PRIMARY	4	const	5	100.00	NULL
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	1	100.00	Using where; Using join buffer (hash join)
2	DERIVED	orders_repair	NULL	ref	PRIMARY,fk_orders_repair_orders1_idx,fk_orders_r...	PRIMARY	4	const	1	100.00	NULL

Timing (as measured at client side):
Execution time: 0:00:0.00000000

Timing (as measured by the server):
Execution time: 0:00:0.00307360
Table lock wait time: 0:00:0.00000400

Errors:
Had Errors: NO
Warnings: 2

Rows Processed:
Rows affected: 0
Rows sent to client: 4
Rows examined: 1

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 1

Joins per Type:
Full table scans (Select_scan): 0
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event Id: 230
Thread Id: 51

Рис. 3.4.1. Скриншот Explain аналізу запита, що підраховує загальну суму замовлення

Маємо оптимальний запит, считується стільки строк, скільки потрібно для підрахунку загальної суми. Поле TYPE – CONST, REF , а також ALL для підзапиту, який вміщує тільки один рядок (загальну суму) та використовує join buffer. Я думаю, оптимізація не потрібна.

3.4.2. Проаналізуємо запит, що підраховує загальну суму приходу

```

52 -- підрахунок суми приходу (для тригеру BEFORE UPDATE ON arrivals)
53 • EXPLAIN SELECT sql_no_cache ROUND(SUM(price_in*quantity)) as total_arrivals_goods
54 FROM arrivals a
55 JOIN arrivals_goods ag USING(arrival_id)
56 JOIN goods g USING(goods_id)
57 WHERE a.arrival_id = 1000;

```

Result Grid | Filter Rows: [] | Export: [] | Wrap Cell Content: []

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	a	NULL	const	PRIMARY	PRIMARY	4	const	1	100.00	Using index
1	SIMPLE	ag	NULL	ref	PRIMARY,fk_arrival_goods_arrivals1_idx,fk_arrival...	PRIMARY	4	const	5	100.00	NULL
1	SIMPLE	g	NULL	eq_ref	PRIMARY	PRIMARY	4	ss_innodb.ag.goods_id	1	100.00	NULL

Timing (as measured at client side): Execution time: 0:00:0.00000000	Joins per Type: Full table scans (Select_scan): 0 Joins using table scans (Select_full_join): 0 Joins using range search (Select_full_range_join): 0 Joins with range checks (Select_range_check): 0 Joins using range (Select_range): 0
Timing (as measured by the server): Execution time: 0:00:0.00071820 Table lock wait time: 0:00:0.00000500	
Errors: Had Errors: NO Warnings: 2	Sorting: Sorted rows (Sort_rows): 0 Sort merge passes (Sort_merge_passes): 0 Sorts with ranges (Sort_range): 0 Sorts with table scans (Sort_scan): 0
Rows Processed: Rows affected: 0 Rows sent to client: 3 Rows examined: 0	Index Usage: At least one Index was used
Temporary Tables: Temporary disk tables created: 0 Temporary tables created: 0	Other Info: Event Id: 349 Thread Id: 51

Рис. 3.4.2. Скриншот Explain аналізу запита, що підраховує загальну суму приходу

Маємо оптимальний запит, считається стільки строк, скільки потрібно для підрахунку загальної суми. Поле TYPE – CONST, REF, EQ_REF. Я думаю, оптимізація не потрібна.

3.4.3. Запит, що отримує дані складу.

31 • EXPLAIN SELECT sql_no_cache stock, reserve FROM goods WHERE goods_id = 1;																									
Result Grid Filter Rows: <input type="text"/> Export: Wrap Cell Content:																									
<table border="1"> <thead> <tr> <th>id</th><th>select_type</th><th>table</th><th>partitions</th><th>type</th><th>possible_keys</th><th>key</th><th>key_len</th><th>ref</th><th>rows</th><th>filtered</th><th>Extra</th></tr> </thead> <tbody> <tr> <td>1</td><td>SIMPLE</td><td>goods</td><td>HULL</td><td>const</td><td>PRIMARY</td><td>PRIMARY</td><td>4</td><td>const</td><td>1</td><td>100.00</td><td>NULL</td></tr> </tbody> </table>	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra	1	SIMPLE	goods	HULL	const	PRIMARY	PRIMARY	4	const	1	100.00	NULL	
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra														
1	SIMPLE	goods	HULL	const	PRIMARY	PRIMARY	4	const	1	100.00	NULL														
Timing (as measured at client side): Execution time: 0:00:0.00000000	Joins per Type: Full table scans (Select_scan): 0 Joins using table scans (Select_full_join): 0 Joins using range search (Select_full_range_join): 0 Joins with range checks (Select_range_check): 0 Joins using range (Select_range): 0																								
Timing (as measured by the server): Execution time: 0:00:0.00038200 Table lock wait time: 0:00:0.00000400																									
Errors: Had Errors: NO Warnings: 2	Sorting: Sorted rows (Sort_rows): 0 Sort merge passes (Sort_merge_passes): 0 Sorts with ranges (Sort_range): 0 Sorts with table scans (Sort_scan): 0																								
Rows Processed: Rows affected: 0 Rows sent to client: 1 Rows examined: 0	Index Usage: At least one Index was used																								
Temporary Tables: Temporary disk tables created: 0 Temporary tables created: 0	Other Info: Event Id: 360 Thread Id: 51																								

Рис. 3.4.3. Скриншот Explain аналізу запита, що отримує дані складу

Тут одна строка в EXPLAIN з типом CONST з пошуком по PK – оптимальний запит.

3.4.4. Ну і останній тест. Дуже кортить перевірити запит Insert Cascade, що вставляє дані одразу в три таблиці і порівняти з трима (навіть чотирма) окремими запитами, що об'єднані в одну транзакцію. Що бистріше?

```

1  DELIMITER //
2  •  DROP TRIGGER orders_after_insert;
3
4  CREATE TRIGGER orders_after_insert
5  AFTER INSERT ON orders
6  FOR EACH ROW
7  BEGIN
8      DECLARE rc_brand_id SMALLINT;
9      DECLARE rc_employee_id TINYINT;
10     DECLARE rc_model_name VARCHAR(20);
11     DECLARE rc_sn VARCHAR(30);
12     DECLARE rc_equipment TEXT;
13     DECLARE rc_fault_description TEXT;
14     DECLARE rp_employee_id TINYINT;
15
16     -- Якщо orders_temp не NULL, зберігаємо дані в orders_received та orders_repair
17     IF NEW.order_temp IS NOT NULL THEN
18
19         SET rc_brand_id = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp, '$.brand_id'));
20         SET rc_employee_id = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp, '$.rc_employee_id'));
21         SET rc_model_name = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp, '$.model_name'));
22         SET rc_sn = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp, '$.sn'));
23         SET rc_equipment = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp, '$.equipment'));
24         SET rc_fault_description = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp, '$.fault_description'));
25         SET rp_employee_id = JSON_UNQUOTE(JSON_EXTRACT(NEW.order_temp, '$.rp_employee_id'));
26
27         INSERT INTO orders_received (order_id, brand_id, employee_id, model_name, sn, equipment, fault_description)
28         VALUES (NEW.order_id, rc_brand_id, rc_employee_id, rc_model_name, rc_sn, rc_equipment, rc_fault_description);
29
30         INSERT INTO orders_repair (order_id, employee_id)
31         VALUES (NEW.order_id, rp_employee_id);
32
33     END IF;
34  END//
35  DELIMITER ;

```

Рис. 3.4.4. Скриншот коду тригера AFTER INSERT ON orders

Перевіримо роботу тригера

До вставки

```

14 •  SELECT order_id, customer_id,
15                 date_returned, total
16     FROM orders
17     WHERE date_returned
18     ORDER BY order_id DESC;

```

order_id	customer_id	date_returned	total
10005	1	2024-09-09	88000
10000	1680	2024-06-30	NULL
9999	2817	2024-05-15	198090

```

6 •  SELECT *
7     FROM orders_received
8     ORDER BY order_id DESC;

```

order_id	brand_id	employee_id	model_name
10012	1	1	2015
10005	1	8	3115
9999	5	12	7108

```

11 •  SELECT order_id, employee_id,
12          price_repair, date_ready
13      FROM orders_repair
14  ORDER BY order_id DESC;

```

Result Grid			
order_id	employee_id	price_repair	date_ready
10005	9	37000	2024-09-06
10005	8	51000	2024-09-08
9999	12	140000	2024-05-15

```

INSERT INTO orders
(customer_id, employee_id, order_temp)
VALUES (4, 4, '{"brand_id":4,
"rc_employee_id":5,"model_name":"1610",
"sn":"SN123-126",
"equipment":"без кабелю живлення",
"fault_description": "поганий друк",
"rp_employee_id": 4}');


```

Після вставки Insert Cascade

```

5 11:12:48 INSERT INTO orders (customer_id, employee_id, order_temp) VALUES (1, 2, '{"brand_id":3, "rc_empl... 1 row(s) affected
14 •  SELECT order_id, customer_id,
15      date_invoice, date_returned, total
16  FROM orders
17  ORDER BY order_id DESC;
6 •  SELECT *
7  FROM orders_received
8  ORDER BY order_id DESC;

```

Result Grid					Result Grid			
order_id	customer_id	date_invoice	date_returned	total	order_id	brand_id	employee_id	model_
10019	1	2024-12-07	NULL	NULL	10019	3	5	1610
10012	1	2024-11-06	NULL	NULL	10012	1	1	2015
10011	6638	2024-09-11	NULL	NULL	10005	1	8	3115

```

11 •  SELECT order_id, employee_id,
12          price_repair, date_ready
13      FROM orders_repair
14  ORDER BY order_id DESC;

```

Result Grid			
order_id	employee_id	price_repair	date_ready
10019	17	0	NULL
10005	9	37000	2024-09-06
10005	8	51000	2024-09-08

Рис. 3.4.5. Робота тригера AFTER INSERT при додаванні даних в таблиці orders, orders_received, orders_repair

Тригер працює як треба, перевіримо його швидкість

```

73 • EXPLAIN
74     INSERT INTO orders (customer_id, employee_id, order_temp)
75     VALUES (1, 2, '{"brand_id":3, "rc_employee_id":5,"model_name":"1610","sn":"SN123-126",
76     "equipment":"без кабелю живлення","fault_description": "торохтить під час увімкнення","rp_employee_id":17}');


```

Query Statistics

Timing (as measured at client side):

Execution time: 0:00:0.0000000

Timing (as measured by the server):

Execution time: 0:00:0.00032850

Table lock wait time: 0:00:0.00000500

Errors:

Had Errors: NO

Warnings: 1

Rows Processed:

Rows affected: 0

Rows sent to client: 1

Rows examined: 0

Temporary Tables:

Temporary disk tables created: 0

Temporary tables created: 0

Joins per Type:

Full table scans (Select_scan): 0

Joins using table scans (Select_full_join): 0

Joins using range search (Select_full_range_join): 0

Joins with range checks (Select_range_check): 0

Joins using range (Select_range): 0

Sorting:

Sorted rows (Sort_rows): 0

Sort merge passes (Sort_merge_passes): 0

Sorts with ranges (Sort_range): 0

Sorts with table scans (Sort_scan): 0

Index Usage:

At least one Index was used

Other Info:

Event Id: 130

Thread Id: 51

Рис. 3.4.6. Скриншот Explain аналізу запита при додаванні даних в таблиці

orders, orders_received, orders_repair - Результат 328,5 мкс

Альтернативний варіант

```

3 • EXPLAIN
4     INSERT INTO orders (customer_id, employee_id)
5     VALUES (5, 5);


```

Query Statistics

Timing (as measured at client side):

Execution time: 0:00:0.0000000

Timing (as measured by the server):

Execution time: 0:00:0.00029250

Table lock wait time: 0:00:0.00000400

Joins per Type:

Full table scans (Select_scan): 0

Joins using table scans (Select_full_join): 0

Joins using range search (Select_full_range_join): 0

Joins with range checks (Select_range_check): 0

Joins using range (Select_range): 0

```

7 • EXPLAIN
8     SELECT MAX(order_id) FROM orders;


```

Query Statistics

Timing (as measured at client side):

Execution time: 0:00:0.0000000

Timing (as measured by the server):

Execution time: 0:00:0.00040200

Table lock wait time: 0:00:0.00000500

Joins per Type:

Full table scans (Select_scan): 0

Joins using table scans (Select_full_join): 0

Joins using range search (Select_full_range_join): 0

Joins with range checks (Select_range_check): 0

Joins using range (Select_range): 0

```

14 • EXPLAIN
15   INSERT INTO orders_received
16     VALUES (10022, 5, 5, "1710", "SN1710-1126", NULL, "з кабелем живлення", "поганий друк");

```

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.00000000

Timing (as measured by the server):
Execution time: 0:00:0.00027160
Table lock wait time: 0:00:0.00000400

Joins per Type:
Full table scans (Select_scan): 0
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0


```

18 • EXPLAIN
19   INSERT INTO orders_repair (order_id, employee_id)
20     VALUES (10022, 5);

```

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.00000000

Timing (as measured by the server):
Execution time: 0:00:0.00026510
Table lock wait time: 0:00:0.00000300

Joins per Type:
Full table scans (Select_scan): 0
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Рис. 3.4.7. Скриншоти Explain аналізу запитів при додаванні даних в таблиці orders, orders_received, orders_repair - Загальний результат 1231.2 мкс

Ще один варіант теста, з використанням Performance Schema

```

1 • START TRANSACTION;
2
3 •   INSERT INTO orders (customer_id, employee_id)
4     VALUES (1, 2);
5
6 •   SET @new_order_id = LAST_INSERT_ID();
7
8 •   INSERT INTO orders_received
9     VALUES (@new_order_id, 3, 5, "1710", "SN1710-1126", NULL, "з кабелем живлення", "поганий друк");
10
11 •  INSERT INTO orders_repair (order_id, employee_id)
12     VALUES (@new_order_id, 15);
13
14 •  COMMIT;
15
16 •  SELECT EVENT_NAME, TIMER_START, TIMER_END, TIMER_WAIT
17   FROM performance_schema.events_transactions_history
18  ORDER BY TIMER_START DESC LIMIT 1;
19

```

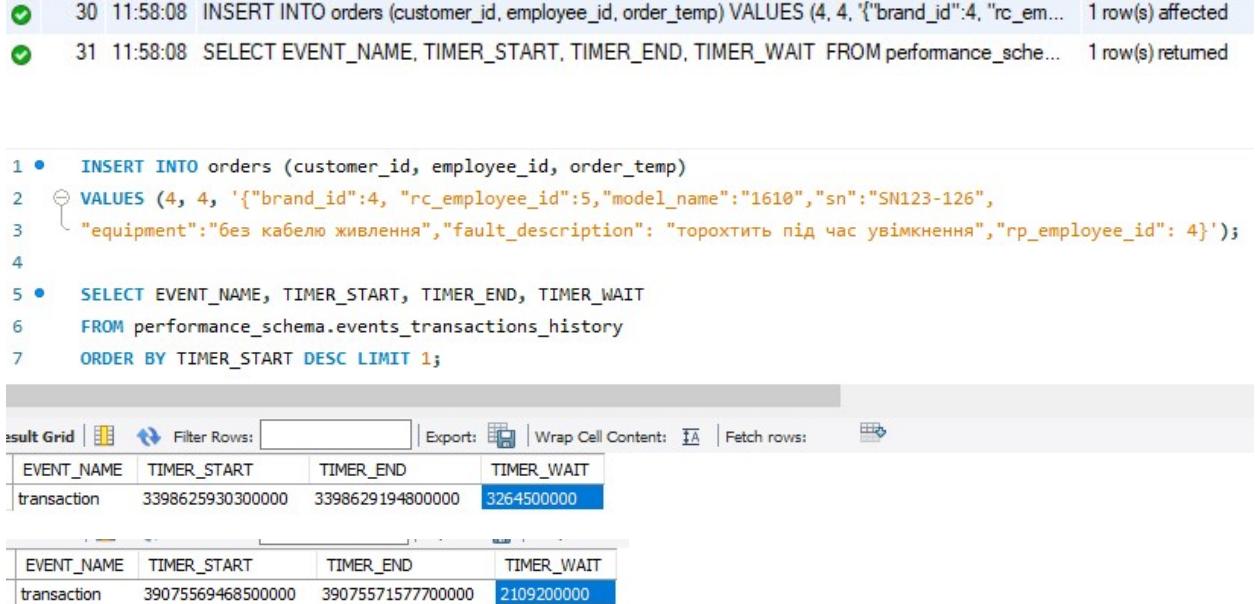
result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

EVENT_NAME	TIMER_START	TIMER_END	TIMER_WAIT
transaction	2432469196800000	2432477101500000	7904700000

EVENT_NAME	TIMER_START	TIMER_END	TIMER_WAIT
transaction	38859020787400000	38859028222600000	7435200000

Рис. 3.4.8. Скриншоти Explain аналізу запитів при додаванні даних в таблиці orders, orders_received, orders_repair - Результат виконання запиту 7904.7 і 7435.2 МКС

Альтернативна вставка



```

1 •   INSERT INTO orders (customer_id, employee_id, order_temp)
2   VALUES (4, 4, {"brand_id":4, "rc_employee_id":5,"model_name":"1610","sn":"SN123-126",
3   "equipment":"без кабелю живлення","fault_description": "торохтить під час увімкнення","rp_employee_id": 4});
4
5 •   SELECT EVENT_NAME, TIMER_START, TIMER_END, TIMER_WAIT
6   FROM performance_schema.events_transactions_history
7   ORDER BY TIMER_START DESC LIMIT 1;

```

EVENT_NAME	TIMER_START	TIMER_END	TIMER_WAIT
transaction	3398625930300000	3398629194800000	3264500000
transaction	39075569468500000	39075571577700000	2109200000

Рис. 3.4.9. Скриншот Explain аналізу запита при додаванні даних в таблиці orders, orders_received, orders_repair - Результат виконання запиту 3264.5 і 2109.2 МКС

Результати тестов демонструють, що вставка даних за допомогою тригера Insert Cascade мінімум в два раза швидше ніж традиційне послідовна вставка в три таблиці з виконанням чотирьох запитів.

Висновки:

- написано та перевірена робота більше 80 тригерів для таблиц MyISAM, InnoDB;
- для MyISAM таблиц customers, employees відмовився від Update CASCADE тригерів в пользу Update RESTRICT, через можливу втрату цілісності даних;
- найбільш цікаві та функціональні тригери: підраховують суму замовлення по кількості та ціні списаних запчастин, а також по сумі всіх ремонтів,

підраховують суму приходу по ціні та кількості запчастин і порівнюють її з сумою в накладній, підраховують зарплату та налоги сотрудника за поточний місяц та інші. В цих тригерах для запобігання конкурентного оновлення даних використав додаткову інструкцію FOR UPDATE

- придумав декілько цікавих тригерів Insert CASCADE, коли дані одним питанням вставляються в декілько таблиць одразу;
- найбільш цікавий варіант тригеру Insert CASCADE протестував двома способами на швидкість у порівнянні з традиційною вставкою;
- перевірив оптимальність трьох питань, що використовуються у тригерах за допомогою Explain, на мою думку їх оптимізувати не треба;
- виконав каскадне видалення застарілих даних одразу з 7-ми таблиць з програмним відключенням тригерів таблиць orders, arrivals;
- робота вишла об'ємна, думаю вона того стояла, дізнався для себе багато нового.