Министерство образования Республики Беларусь Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

ОТЧЕТ

Лабораторная работа №4

по теме: «Интернет магазин продажи компьютерной техники различных производителей»

Выполнил: Соловьёв Д.Н. гр.753502

Проверил: Алексеев И.Г.

Минск

БГУИР 2020

**SQL Запросы**

1. **Триггеры и процедуры для работы с backup копиями**

* Создаём процедуру для сохранения даных в таблицу с backup копиями

DELIMITER $$

CREATE PROCEDURE save\_backup\_user(IN \_login VARCHAR(45), IN \_operation\_type enum('Update','Delete'))

BEGIN

INSERT INTO OnlineStoreDB.users\_backup

(operation\_type, user\_id, change\_date, login, password, salt, role, status, block\_begin, block\_end,

registration\_date, id\_name1, id\_name2, id\_name3, id\_personal\_info)

SELECT \_operation\_type, id, NOW(), login, password, salt, role, status, block\_begin, block\_end, registration\_date, id\_name1, id\_name2, id\_name3, id\_personal\_info

FROM users WHERE login = \_login FOR UPDATE;

END$$

* Создаём триггер для создания backup для операции типа `update`

DELIMITER $$

CREATE TRIGGER user\_backup\_update BEFORE UPDATE

ON users FOR EACH ROW

BEGIN

IF @disable\_backup\_trigger IS NULL OR @disable\_backup\_trigger = 0 THEN

CALL save\_backup\_user(OLD.login, 'Update');

END IF;

END$$

* Создаём триггер для создания backup для операции типа `delete`

DELIMITER $$

CREATE TRIGGER user\_backup\_delete BEFORE DELETE

ON users FOR EACH ROW

BEGIN

IF @disable\_backup\_trigger IS NULL OR @disable\_backup\_trigger = 0 THEN

CALL save\_backup\_user(OLD.login, 'Delete');

END IF;

END$$

* Процедура для определения типа последней операции из таблицы backup и отмены этой операции

DELIMITER $$

CREATE PROCEDURE undo\_user\_backup\_procedure()

BEGIN

DECLARE backup\_id smallint;

DECLARE backup\_type VARCHAR(20);

START TRANSACTION;

SET @disable\_backup\_trigger = 1;

SELECT id, operation\_type INTO backup\_id, backup\_type FROM users\_backup ORDER BY change\_date DESC limit 1;

IF backup\_type = 'Update' THEN

UPDATE users

INNER JOIN users\_backup

ON users.id = users\_backup.user\_id

SET users.password = users\_backup.password, users.salt = users\_backup.salt, users.role = users\_backup.role,

users.status = users\_backup.status, users.block\_begin = users\_backup.block\_begin, users.block\_end = users\_backup.block\_end,

users.id\_name1 = users\_backup.id\_name1, users.id\_name2 = users\_backup.id\_name2, users.id\_name3 = users\_backup.id\_name3,

users.id\_personal\_info = users\_backup.id\_personal\_info

WHERE users\_backup.id = backup\_id;

ELSE

INSERT INTO OnlineStoreDB.users

(id, login, password, salt, role, status, block\_begin, block\_end,

registration\_date, id\_name1, id\_name2, id\_name3, id\_personal\_info)

SELECT user\_id, login, password, salt, role, status, block\_begin, block\_end,

registration\_date, id\_name1, id\_name2, id\_name3, id\_personal\_info FROM users\_backup WHERE id = backup\_id;

END IF;

DELETE FROM users\_backup WHERE id = backup\_id;

SET @disable\_backup\_trigger = 0;

COMMIT;

END$$

1. **Процедуры для управления пользователями**

* Создание пользователя

DELIMITER $$

CREATE PROCEDURE insert\_user(IN \_n1 VARCHAR(64), IN \_n2 VARCHAR(64), IN \_n3 VARCHAR(64), IN \_log VARCHAR(45), IN \_pass VARCHAR(128), IN \_salt VARCHAR(64),

IN \_r VARCHAR(20))

BEGIN

START TRANSACTION;

INSERT IGNORE INTO names1(name1) VALUES (\_n1);

INSERT IGNORE INTO names2(name2) VALUES (\_n2);

INSERT IGNORE INTO names3(name3) VALUES (\_n3);

INSERT INTO users(login, password, salt, role, status, registration\_date, id\_name1, id\_name2, id\_name3)

VALUES (\_log, \_pass, \_salt, \_r, 'Active', NOW(),

(SELECT id FROM names1 WHERE name1=\_n1),

(SELECT id FROM names2 WHERE name2=\_n2),

(SELECT id FROM names3 WHERE name3=\_n3));

SELECT NOW()

COMMIT;

END$$

* Авторизация пользователя

DELIMITER $$

CREATE PROCEDURE login\_user(IN \_log VARCHAR(50), IN \_pass VARCHAR(50))

BEGIN

DECLARE user\_id smallint;

DECLARE \_status VARCHAR(20);

SELECT id INTO user\_id FROM users WHERE login = \_log AND password = \_pass;

SELECT user\_id;

IF user\_id > 0 THEN

SELECT status INTO \_status FROM users WHERE id = user\_id;

IF \_status = "Active" THEN

SELECT \* FROM users WHERE id = user\_id;

ELSE

SELECT "User is blocked";

END IF;

ELSE

SELECT "Wrong login or pass";

END IF;

END$$

* Удаление пользователя

DELIMITER $$

CREATE PROCEDURE delete\_user(IN \_log VARCHAR(45))

BEGIN

DELETE FROM users WHERE login=\_log;

END$$

* Поиск пользователя по различны критериям

DELIMITER $$

CREATE PROCEDURE search\_user(IN \_n1 VARCHAR(64), IN \_n2 VARCHAR(64), IN \_n3 VARCHAR(64), IN \_r VARCHAR(20), IN \_s VARCHAR(20))

BEGIN

SELECT users.id, login, status, role, block\_begin, block\_end, registration\_date,

name1, name2, name3, id\_personal\_info FROM users

INNER JOIN names1 ON users.id\_name1 = names1.id

INNER JOIN names2 ON users.id\_name2 = names2.id

INNER JOIN names3 ON users.id\_name3 = names3.id

WHERE

(name1 LIKE CONCAT('%', \_n1, '%')) AND

(name2 LIKE CONCAT('%', \_n2, '%')) AND

(name3 LIKE CONCAT('%', \_n3, '%')) AND

(role LIKE CONCAT('%', \_r, '%')) AND

(status LIKE CONCAT('%', \_s, '%'));

END$$

* Блокировка и разблокировка пользователя

DELIMITER $$

CREATE PROCEDURE set\_blocked(IN \_s CHAR(10), IN \_log CHAR(50), IN \_start DATETIME, IN \_end DATETIME)

BEGIN

UPDATE users SET

block\_begin = \_start, block\_end = \_end, status = \_s

WHERE login = \_log;

END$$