|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**Московский техникум космического приборостроения**

Отделение «Информационные системы и программирование»

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

по профессиональному модулю\_ПМ.07 Соадминистрирование баз данных и серверов

Код, Специальность 09.02.07 Информационные системы и программирование Квалификация – Администратор баз данных

Место прохождения практики Московский техникум космического приборостроения

Московского государственного технического университета им. Н.Э. Баумана

(полное название организации)

Выполнил студент\_\_\_\_Миронова Дарья Романовна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Курс\_\_\_\_\_4\_\_\_\_\_\_ Группа\_\_\_\_\_ТБД-81\_\_\_\_\_\_\_\_\_

Подпись студента\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата приема отчета \_\_\_ \_\_\_\_\_\_\_\_\_\_\_202 г.

Руководитель практики от техникума \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_­­­­­­­\_\_\_\_\_\_\_\_

(подпись) (фамилия, имя, отчество)

*2024г.*

Цель: ознакомление с MySQL Server.

Оснащение: ПК, учебная и справочная литература.

На рисунке 1 изображена ER-диаграмма созданной базы данных torg.

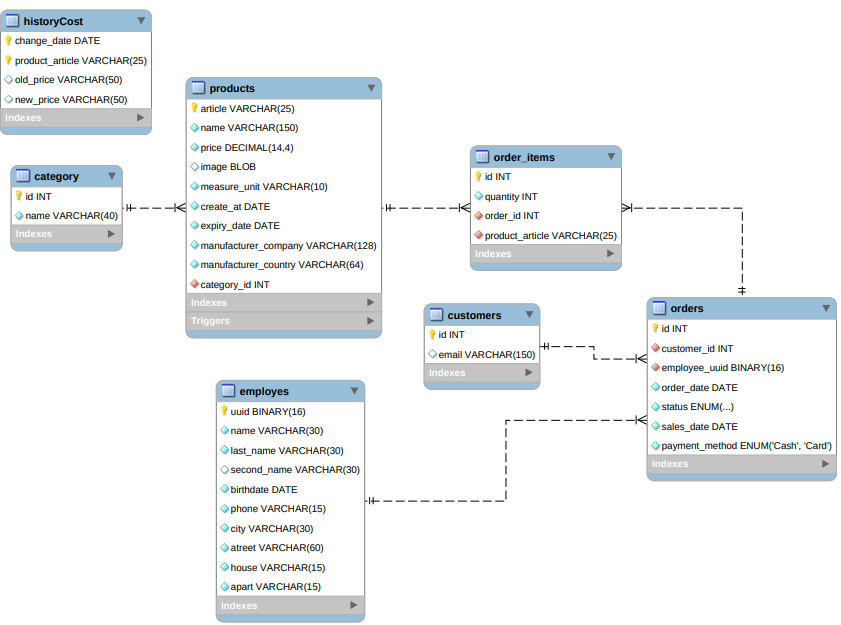


Рис. 1 – ER-диаграмма базы данных

Создание базы данных torg и его таблиц представлено фрагментом sql-скрипта:

create database torg;

use torg;

create table category(

id integer primary key AUTO\_INCREMENT,

name varchar(40) not null

);

create table customers(

id integer primary key AUTO\_INCREMENT,

email varchar(150)

);

create employes(

uuid binary(16) primary key not null default (uuid\_to\_bin(uuid())),

name varchar(30) not null,

last\_name varchar(30) not null,

second\_name varchar(30),

birthdate date not null,

phone varchar(15) not null,

city varchar(30) not null,

atreet varchar(60) not null,

house varchar(15) not null,

apart varchar(15) not null

);

creat table orders(

id integer primary key AUTO\_INCREMENT,

customer\_id integer not null,

employee\_uuid binary(16) not null,

order\_date date not null,

status enum('New', 'In progress', 'Ready') not null default 'New',

sales\_date date not null,

payment\_method enum('Cash', 'Card') not null default 'Card',

foreign key (customers) references customers(id),

foreign key (employee\_uuid) references employes(uuid)

);

create table products(

article varchar(25) primary key,

name varchar(150) not null,

price decimal(14,4) not null,

image blob,

measure\_unit varchar(10) not null,

create\_at date not null,

expiry\_date date not null,

manufacturer\_company varchar(128) not null,

manufacturer\_country varchar(64) not null,

category\_id integer not null,

foreign key (category\_id) references category(id)

);

create table order\_items(

id integer primary key AUTO\_INCREMENT,

quantity integer not null,

order\_id integer not null,

product\_article varchar(25) not null,

foreign key (order\_id) references orders(id),

foreign key (product\_article) references products(article)

);

create table historyCost(

change\_date date primary key,

product\_article varchar(25) primary key,

old\_price varchar(50),

new\_price varchar(50)

);

На рисунке 2 изображен результат выполнения sql-скрипта.

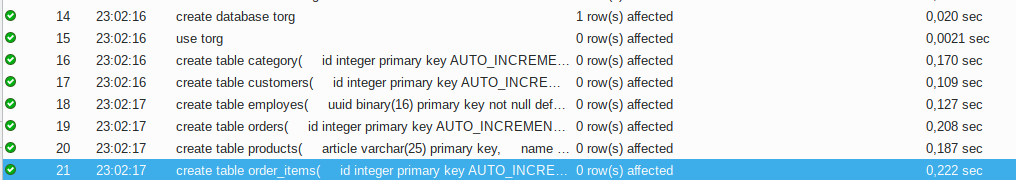


Рис. 2 – Результат выполнения sql-скрипта

Заполнение таблиц базы данных torg представлено фрагментом sql-скрипта:

insert into category (id, name) values (1, 'Продукты');

insert into category (id, name) values (2, 'Игрушки');

insert into customers (id, email) values (1, 'darrommir@mail.ru');

insert into customers (id, email) values (2, 'hih1');

insert into employes (uuid, name, last\_name, second\_name, birthdate, phone, city, atreet, house, apart) values (UUID\_TO\_BIN('3184292e-e49b-11ee-859f-00155df54038'), 'Петрова', 'Анна', 'Сергеевна', '2004-04-09', '89765676677', 'Москва', 'Иванова', '8', '34');

insert into employes (uuid, name, last\_name, second\_name, birthdate, phone, city, atreet, house, apart) values (UUID\_TO\_BIN('3bb27c28-e49b-11ee-859f-00155df54038'), 'Папоян', 'Татьяна', 'Робертовна', '2004-01-03', '89765676678', 'Москва', 'Петрова', '9', '35');

insert into orders(id, customer\_id, order\_date, status, sales\_date, payment\_method, employee\_uuid) values (1, 1, '2024-02-13', 'Ready', '2024-02-14', 'Card', UUID\_TO\_BIN('3184292e-e49b-11ee-859f-00155df54038'));

insert into orders(id, customer\_id, order\_date, status, sales\_date, payment\_method, employee\_uuid) values (2, 1, '2024-02-13', 'Ready', '2024-02-14', 'Card', UUID\_TO\_BIN('3184292e-e49b-11ee-859f-00155df54038'));

insert into orders(id, customer\_id, order\_date, status, sales\_date, payment\_method, employee\_uuid) values (3, 1, '2024-02-14', 'Ready', '2024-02-15', 'Cash', UUID\_TO\_BIN('3bb27c28-e49b-11ee-859f-00155df54038'));

insert into orders(id, customer\_id, order\_date, status, sales\_date, payment\_method, employee\_uuid) values (4, 1, '2024-02-14', 'Ready', '2024-02-15', 'Cash', UUID\_TO\_BIN('3bb27c28-e49b-11ee-859f-00155df54038'));

insert into orders(id, customer\_id, order\_date, status, sales\_date, payment\_method, employee\_uuid) values (5, 1, '2024-02-15', 'Ready', '2024-02-16', 'Cash', UUID\_TO\_BIN('3184292e-e49b-11ee-859f-00155df54038'));

insert into orders(id, customer\_id, order\_date, status, sales\_date, payment\_method, employee\_uuid) values (6, 2, '2024-02-16', 'Ready', '2024-02-17', 'Cash', UUID\_TO\_BIN('3184292e-e49b-11ee-859f-00155df54038'));

insert into orders(id, customer\_id, order\_date, status, sales\_date, payment\_method, employee\_uuid) values (7, 2, '2024-02-17', 'In progress', '2024-02-17', 'Card', UUID\_TO\_BIN('3bb27c28-e49b-11ee-859f-00155df54038'));

insert into orders(id, customer\_id, order\_date, status, sales\_date, payment\_method, employee\_uuid) values (8, 2, '2024-03-13', 'In progress', '2024-03-13', 'Card', UUID\_TO\_BIN('3184292e-e49b-11ee-859f-00155df54038'));

insert into orders(id, customer\_id, order\_date, status, sales\_date, payment\_method, employee\_uuid) values (9, 2, '2024-03-15', 'New', '2024-03-15', 'Card', UUID\_TO\_BIN('3bb27c28-e49b-11ee-859f-00155df54038'));

insert into orders(id, customer\_id, order\_date, status, sales\_date, payment\_method, employee\_uuid) values (10, 2, '2024-03-16', 'New', '2024-03-16', 'Card', UUID\_TO\_BIN('3184292e-e49b-11ee-859f-00155df54038'));

insert into products (article, name, category\_id, price, measure\_unit, create\_at, expiry\_date, manufacturer\_company, manufacturer\_country) values ('456456', 'Молоко', 1, 456.5, 'шт', '2024-03-01', '2024-04-02', 'Домик в деревне', 'Российская Федерация');

insert into products (article, name, category\_id, price, measure\_unit, create\_at, expiry\_date, manufacturer\_company, manufacturer\_country) values ('888888', 'Самолет Лего', 2, 1232.34, 'шт', '2024-02-01', '2025-04-02', 'LEGO', 'США');

insert into products (article, name, category\_id, price, measure\_unit, create\_at, expiry\_date, manufacturer\_company, manufacturer\_country) values ('455666', 'Масло', 1, 234.5, 'шт', '2024-03-01', '2024-03-10', 'Домик в деревне', 'Российская Федерация');

insert into order\_items (id, order\_id, product\_article, quantity) values (1, 1, '456456', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (2, 1, '888888', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (3, 2, '888888', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (4, 2, '456456', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (5, 3, '888888', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (6, 3, '456456', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (7, 4, '888888', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (8, 4, '456456', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (9, 5, '888888', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (10, 5, '456456', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (11, 6, '888888', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (12, 6, '456456', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (13, 7, '888888', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (14, 7, '456456', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (15, 8, '888888', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (16, 8, '456456', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (17, 9, '888888', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (18, 9, '456456', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (19, 10, '456456', 1);

insert into order\_items (id, order\_id, product\_article, quantity) values (20, 10, '888888', 1);

На рисунке 3 изображен результат выполнения sql-скрипта.

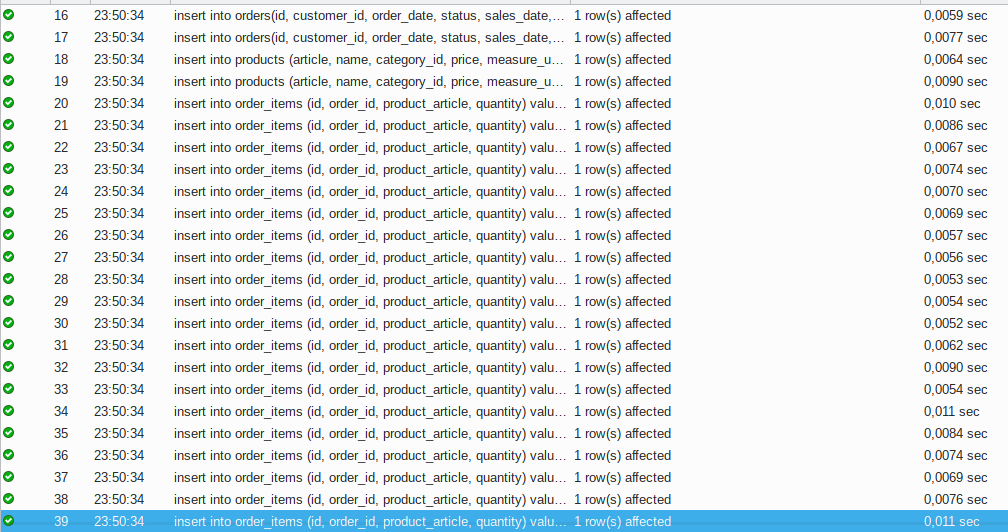


Рис. 3 – Результат выполнения sql-скрипта

Создание процедуры, которая проверяет адрес электронной почты, хранящийся в базе данных, на корректность. Корректным является адрес, если он содержит допустимые символы в каждой из частей и части разделены сначала символом @ затем «.» (Формат адреса электронной почты: A-Z0-9 @ A-Z0-9 . A-Z0-9). Проверяет, чтобы в адресе отсутствовали символы, из-за которых часто возникают ошибки ( [ " < > ' ]). Процедура выводит все адреса электронных почт с указанием соответствующего признака валидности (1-валидный, 0 – не валидный).

Создание процедуры checkEmail() представлено фрагментом sql-скрипта:

DELIMITER //

CREATE PROCEDURE checkEmail()

BEGIN

SELECT id, email,

CASE WHEN email REGEXP '^[A-Z0-9]+@[A-Z0-9]+\.[A-Z0-9]+$' AND (

LOCATE('[', email) + LOCATE(']', email) + LOCATE('<', email) + LOCATE('>', email) + LOCATE('\'', email) + LOCATE('"', email) = 0

) THEN 1

ELSE 0

END AS isValid

FROM customers;

END //

DELIMITER ;

На рисунке 4 изображен результат выполнения sql-скрипта.



Рис. 4 – Результат выполнения sql-скрипта

Вызов процедуры checkEmail() представлен фрагментом sql-скрипта:

call checkEmail;

На рисунке 5 изображен результат выполнения sql-скрипта.

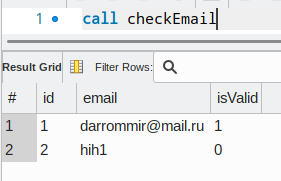


Рис. 5 – Результат выполнения sql-скрипта

Разработка триггера, который при изменении цены будет записывать данные в таблицу HistoryCost.

Создание триггера updateProducts представлено фрагментом sql-скрипта:

DELIMITER //

CREATE TRIGGER updateProducts

AFTER UPDATE ON products

FOR EACH ROW

BEGIN

IF new.price <> old.price THEN

INSERT INTO historyCost (change\_date, product\_article, old\_price, new\_price) VALUES (now(), new.article, old.price, new.price);

END IF;

END;

//

DELIMITER ;

На рисунке 6 изображен результат выполнения sql-скрипта.

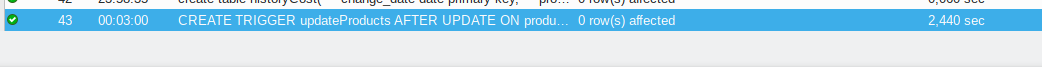


Рис. 6 – Результат выполнения sql-скрипта

Обновление цены в таблице products представлено фрагментов sql-скрипта:

update products

set products.price = products.price \* 0.85

WHERE products.article = 888888;

На рисунке 7 изображен результат выполнения sql-скрипта.



Рис. 6 – Результат выполнения sql-скрипта

На рисунке 7 представлена таблица historyCost после обновление цены в таблице products.

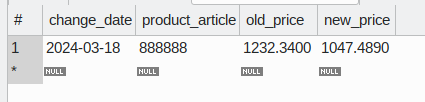


Рис. 7 – Таблица historyCost

Вывод

В ходе выполнения практической работы были получены навыки проектирования базы данных согласно предметной области и написания процедур и триггеров.