Специальность 09.02.07 Информационные системы и программирование

Проверил:

Ефимова Т.Н.

Дата: «\_\_»\_\_\_\_\_\_2020 г. Оценка:\_\_\_\_\_\_\_\_\_\_

Выполнил:

студент группы Y2337

Князев Андрей

ОТЧЕТ

О ПРАКТИЧЕСКОЙ РАБОТЕ

по теме: Создание базы данных и работа с ней

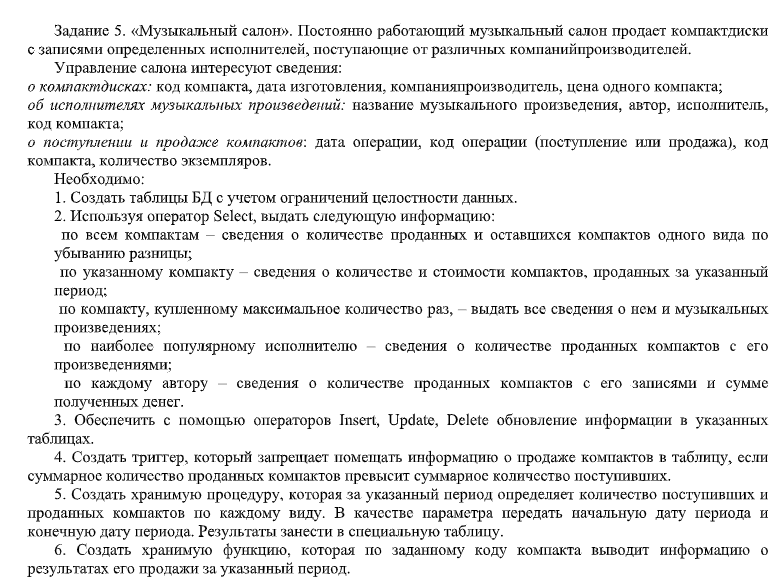
Университет ИТМО

Факультет среднего профессионального образования

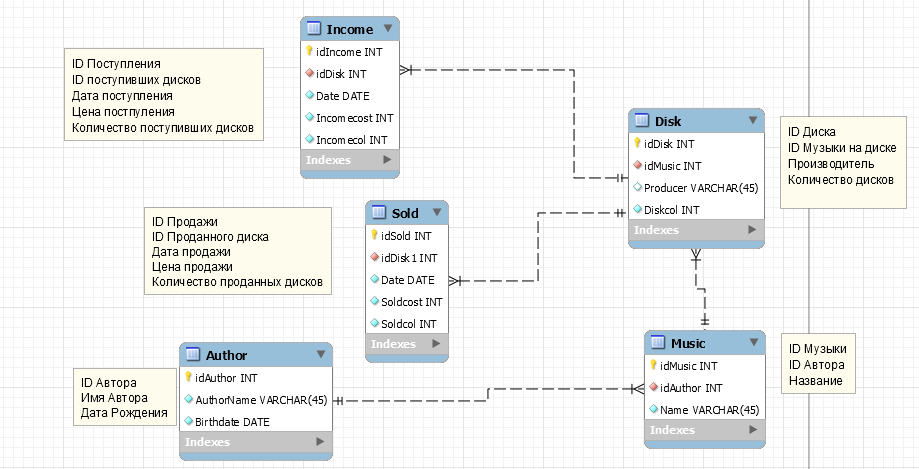
Санкт-Петербург 2020

**Вариант: 5**

1. Индивидуальное задание.



1. Диаграмма классов:



1. Структура таблицы:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Наименование  атрибута | Тип | Первичный ключ | | Внешний  ключ | Обязательность | Ограничения  целостности | Пояснение | Пример заполнения |
| Собственный  атрибут | Внешний  ключ |
| Disc(Диск) | | | | | | | | |
| idDisc | int | + |  | + | + | Уникален | Id диска | 1 |
| idMusic | int |  |  | + | + | Число | Id музыки на диске | 4 |
| Producer | Varchar(45) | + |  |  |  | Отсутствует | Производитель диска | FSPO ITMO |
| Diskcol | Int | + |  |  | + | Число | Количесво дисков | 5 |
| Music(Музыка) | | | | | | | | |
| idMusic | int | + |  | + | + | Уникален | Id музыки | 32 |
| idAuthor | int |  |  | + | + | Число | Id Автора | 43 |
| Name | varchar(35) | + |  |  | + | Отсутствует | Название музыки | Name |
| Author(Автор) | | | | | | | | |
| idAuthor | int | + |  | + | + | Уникален | Id Автора | 3 |
| AuthorName | Varchar(35) | + |  |  | + | Отсутствует | Имя автора | Andrey |
| Birthdate | date | + |  |  | + | Дата | Дата рождения автора | 2015-11-4 |
| Income(Поступление) | | | | | | | | |
| idIncome | Int | + |  |  | + | Число | Id Поступления | 2 |
| idDisk | int |  |  | + | + | Число | Id поступившего диска | 5 |
| Date | Date | + |  |  | + | Дата | Дата поступления | 2014-12-03 |
| Incomecost | Int | + |  |  | + | Число | Цена диска при поступлении | 500 |
| Incomecol | Int | + |  |  | + | Число | Количество поступивших дисков | 5 |
| Sold(Продажа) | | | | | | | | |
| idSold | Int | + |  |  | + | Число | Id Продажи | 4 |
| idDisk1 | Int | + |  | + |  | Число | Id проданного диска | 7 |
| Date | Date | + |  |  | + | Дата | Дата продажи | 2015-04-05 |
| Soldcost | Int | + |  |  | + | Число | Цена диска при продаже | 300 |
| Soldcol | Int | + |  |  | + | Число | Количество проданных дисков | 12 |

1. Скрипты создания таблиц:

-- MySQL Script generated by MySQL Workbench

-- Wed Sep 23 11:54:07 2020

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema trs

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema trs

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `trs` DEFAULT CHARACTER SET utf8 ;

USE `trs` ;

-- -----------------------------------------------------

-- Table `trs`.`Author`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `trs`.`Author` (

`idAuthor` INT NOT NULL AUTO\_INCREMENT,

`AuthorName` VARCHAR(45) NOT NULL,

`Birthdate` DATE NOT NULL,

PRIMARY KEY (`idAuthor`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `trs`.`Music`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `trs`.`Music` (

`idMusic` INT NOT NULL AUTO\_INCREMENT,

`idAuthor` INT NOT NULL,

`Name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idMusic`),

INDEX `idAuthor\_idx` (`idAuthor` ASC) VISIBLE,

CONSTRAINT `idAuthor`

FOREIGN KEY (`idAuthor`)

REFERENCES `trs`.`Author` (`idAuthor`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `trs`.`Disk`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `trs`.`Disk` (

`idDisk` INT NOT NULL AUTO\_INCREMENT,

`idMusic` INT NOT NULL,

`Producer` VARCHAR(45) NULL,

`Diskcol` INT NOT NULL,

PRIMARY KEY (`idDisk`),

INDEX `idMusic\_idx` (`idMusic` ASC) VISIBLE,

CONSTRAINT `idMusic`

FOREIGN KEY (`idMusic`)

REFERENCES `trs`.`Music` (`idMusic`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `trs`.`Income`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `trs`.`Income` (

`idIncome` INT NOT NULL,

`idDisk` INT NOT NULL,

`Date` DATE NOT NULL,

`Incomecost` INT NOT NULL,

`Incomecol` INT NOT NULL,

PRIMARY KEY (`idIncome`),

INDEX `idDisk\_idx` (`idDisk` ASC) VISIBLE,

CONSTRAINT `idDisk`

FOREIGN KEY (`idDisk`)

REFERENCES `trs`.`Disk` (`idDisk`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `trs`.`Sold`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `trs`.`Sold` (

`idSold` INT NOT NULL,

`idDisk1` INT NOT NULL,

`Date` DATE NOT NULL,

`Soldcost` INT NOT NULL,

`Soldcol` INT NOT NULL,

PRIMARY KEY (`idSold`),

INDEX `idDisk\_idx` (`idDisk1` ASC) VISIBLE,

CONSTRAINT `idDisk1`

FOREIGN KEY (`idDisk1`)

REFERENCES `trs`.`Disk` (`idDisk`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

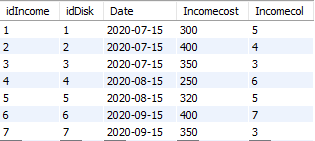
ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

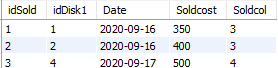
SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

1. Команды заполнения таблицы:

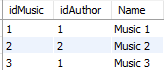
INSERT INTO Income VALUES (1, 1, '2020-07-15',300,5),(2, 2, '2020-07-15',400,4),(3, 3, '2020-07-15',350,3),(4,4,'2020-08-15',250,6),(5,5,'2020-08-15',320,5),(6,6,'2020-09-15',400,7),(7,7,'2020-09-15',350,3); 

INSERT INTO Sold VALUES (1,1,'2020-09-16',350,3),(2,2,'2020-09-16',400,3),(3,4,'2020-09-17','500',4);

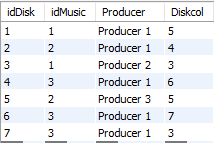


INSERT INTO Author VALUES (1,'Name 1','2013-06-12'),(2,'Name 2','2014-03-14');



INSERT INTO Music VALUES (1,1,'Music 1'),(2,2,'Music 2'),(3,1,'Music 3'); 

INSERT INTO Disk VALUES (1,1,'Producer 1',5),(2,2,'Producer 1',4),(3,1,'Producer 2',3),(4,3,'Producer 1',6),(5,2,'Producer 3',5),(6,3,'Producer 1',7),(7,3,'Producer 1',3);



1. Скриншоты проверки внешних ключей:

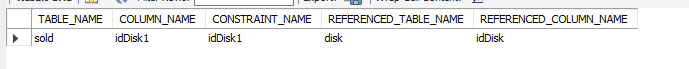
use INFORMATION\_SCHEMA;

select TABLE\_NAME, COLUMN\_NAME, CONSTRAINT\_NAME, REFERENCED\_TABLE\_NAME, REFERENCED\_COLUMN\_NAME from KEY\_COLUMN\_USAGE where TABLE\_SCHEMA = "trs" and TABLE\_NAME = "Disk" and REFERENCED\_COLUMN\_NAME is not NULL;

use INFORMATION\_SCHEMA;

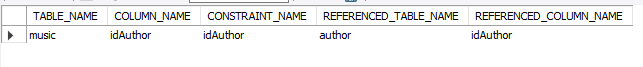
select TABLE\_NAME, COLUMN\_NAME, CONSTRAINT\_NAME, REFERENCED\_TABLE\_NAME, REFERENCED\_COLUMN\_NAME from KEY\_COLUMN\_USAGE where TABLE\_SCHEMA = "trs" and TABLE\_NAME = "income" and REFERENCED\_COLUMN\_NAME is not NULL;

use INFORMATION\_SCHEMA;

select TABLE\_NAME, COLUMN\_NAME, CONSTRAINT\_NAME, REFERENCED\_TABLE\_NAME, REFERENCED\_COLUMN\_NAME from KEY\_COLUMN\_USAGE where TABLE\_SCHEMA = "trs" and TABLE\_NAME = "sold" and REFERENCED\_COLUMN\_NAME is not NULL; 

use INFORMATION\_SCHEMA;

select TABLE\_NAME, COLUMN\_NAME, CONSTRAINT\_NAME, REFERENCED\_TABLE\_NAME, REFERENCED\_COLUMN\_NAME from KEY\_COLUMN\_USAGE where TABLE\_SCHEMA = "trs" and TABLE\_NAME = "music" and REFERENCED\_COLUMN\_NAME is not NULL;



1. Определение объема занимаемой памяти:

select table\_schema AS "Name", ROUND(SUM(data\_length + index\_length) / 1024 /1024 , 2) AS "Size in MB" FROM information\_schema.TABLES where table\_schema="trs" GROUP BY table\_schema;



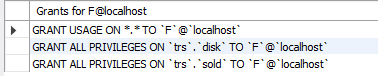
1. Создание пользователей с различными привилегиями:

create user 'F'@'localhost' identified by '0000';

grant all privileges on `trs`.disk to 'F'@localhost;

grant all privileges on `trs`.sold to 'F'@localhost;

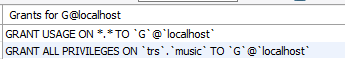
show grants for 'F'@'localhost';



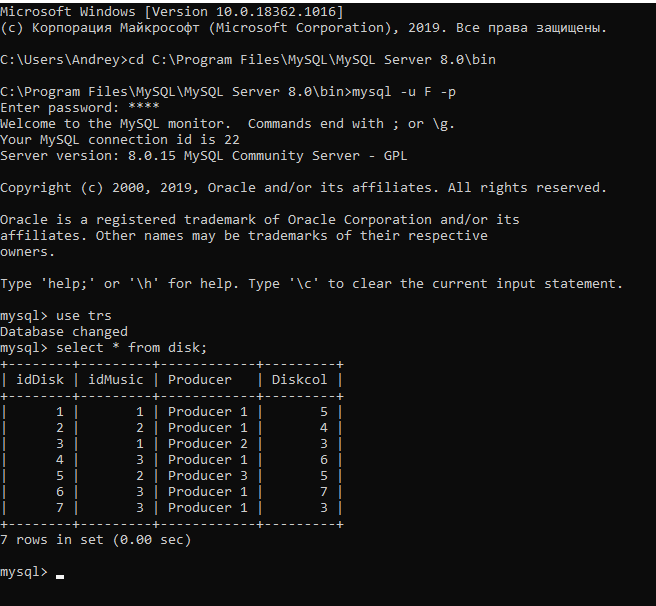
create user 'G'@'localhost' identified by '1111';

grant all privileges on `trs`.music to 'G'@localhost;

show grants for 'G'@'localhost'



1. Соединение по заданному хосту, имени и паролю, проверка привилегий.



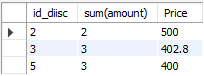
1. Выполненные запросы:

Выдать сведения о музыке, диск с которой был продан наибольшее количество раз.

произведениях;

1. По указанному компактдиску – сведения о количестве и стоимости компактов, проданных за указанный период;

SQL-код запроса: select distinct id\_diisc,sum(amount),Price from sales where date between '2020-02-01' and '2021-01-01' group by id\_diisc;



1. По компактдиску, купленному минимальное количество раз – выдать все сведения о нем и имеющихся там играх.

SQL – код:

select id\_diisc,production\_date,Firm,name,sum(amount) as 'Кол-во купленных' rom sales,disc,game where sales.id\_diisc=disc.id\_disc and disc.id\_game=game.id\_game group by id\_diisc order by sum(amount) limit 1;



1. По наиболее популярной игре – сведения о количестве проданных компактдисков с этой игрой.

SQL – код: select id\_diisc,production\_date,Firm,name,sum(amount) as 'Кол-во купленных' rom sales,disc,game where sales.id\_diisc=disc.id\_disc and disc.id\_game=game.id\_game group by id\_diisc DESC order by sum(amount) limit 1;



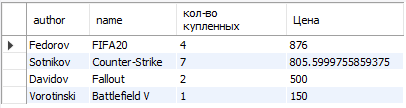
1. По всем компактдискам – сведения о количестве проданных и оставшихся компактдисков одного вида по убыванию разницы.

SQL –код: select id\_disc,(select sum(amount) from sales where id\_diisc=id\_disc) as 'Кол-во проданных',((select sum(amount) from buy where id\_discc=id\_disc)-(select sum(amount) from sales where id\_diisc=id\_disc)) as 'Кол-во на складе' from disc;



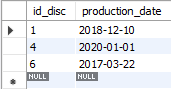
1. По каждому из авторов – сведения о количестве проданных компактдисков с его играми и сумме полученных денег.

SQLкод – select author,name,sum(amount) as 'кол-во купленных', (select sum(Price) from sales where sales.id\_diisc=disc.id\_disc) as 'Цена'from sales,game,disc where game.id\_game=disc.id\_game and sales.id\_diisc=disc.id\_disc group by author



1. Вывести диски, произведенные с «2015-01-01» по настоящий день.

SQL код – select id\_disc,production\_date from disc where production\_date between '2015-01-01' and current\_date()



1. Найти диск, произведенный позже всех остальных.

SQL код – select id\_disc,production\_date from disc where production\_date=(select max(production\_date) from disc);



1. Вывести тех авторов и id их дисков, диски которых продались больше чем одной копии. Авторов вывести по алфавиту.

select author,id\_diisc,count(amount) from sales,disc,game where sales.id\_diisc=disc.id\_disc and disc.id\_game=game.id\_game group by id\_diisc having count(amount)>1 order by author;

9) Вывести авторов и id дисков, в которых есть их игры.

select author,id\_disc from game, disc where game.id\_game=disc.id\_game;



10) Вывести id ,цену и кол-во проданных дисков, где цена не равна 100, меньше 200 и больше 90.

select distinct id\_diisc,amount,Price from sales where Price = any (select Price from sales where Price != 100 and Price < 200 and Price>90 )



1. Скрипты и скриншоты создания триггера, процедуры.

SQL – код триггера:

------------------------------------------------------------------------

SQL – код процедуры, которая выводит кол-во проданных и купленных игр за указанный срок: drop procedure if exists proc;

delimiter $$

create procedure proc(in datein date, in datesale date)

begin

select id\_disc,(select sum(amount) from buy where id\_discc=id\_disc and date between datein and datesale) as 'Кол-во поступивших',

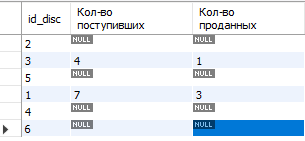
(select sum(amount) from sales where id\_diisc=id\_disc and Date between datein and datesale) as 'Кол-во проданных' from disc;

end $$

delimiter;

call proc('2001-03-15','2020-02-01');

Результат выполнения процедуры:



1. Explain до создания индекса и после:

explain select \* from buy where Price = (select MAX(Price) from buy)



CREATE INDEX myindex ON buy(`Price`);

explain select \* from buy where Price = (select MAX(Price) from buy); 