МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Н.В. Путилова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №6 |
| Хранимые процедуры |
| по курсу: Проектирование баз данных |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4932 |  |  |  | Н.С. Иванов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

#### Текст задания

8. костюмерная театра: роль, спектакль, название костюма, деталь костюма, размер, автор модели, дата разработки

а. спектакли, в которых используются костюмы, имеющие в названии слово «принц»

б. костюм, в котором есть и плащ и штаны

в. спектакль, на который пока нет костюмов

г. роль, к которой разрабатывался самый старый из костюмов

д. автор, разработавший наибольшее число костюмов

е. костюм, в котором есть все типы деталей

ж. автор, не разрабатывавший костюмы к «Золушке», но разрабатывавший к «Мастеру и Маргарите»

#### Физическая модель



#### Текст запросов на SQL

# создать в БД хранимые процедуры, реализующие:  
# ---------------------------------------------  
  
# ================================================================================================  
# 1

##### # — вставку с пополнением справочников

# - получаем ссылку на внешний ключ по значению данных из родительской таблицы  
# - если данных нет - добавляем в родительскую  
# - затем вставляем в дочернюю  
  
  
delimiter //  
  
create procedure *insert\_concrete\_detail* (name\_type varchar(20), color\_detail varchar(20)) begin  
  
 declare type\_id\_new int;  
  
 if exists(select \* from type\_detail where nameType = name\_type) then  
 select td.id\_type\_detail into type\_id\_new from type\_detail td where nameType = name\_type;  
 else  
 begin  
 insert into type\_detail(nameType)  
 values (name\_type);  
 set type\_id\_new = last\_insert\_id();  
 end;  
 end if;  
  
 insert into concrete\_detail (id\_type\_detail, colorDetail)  
 values (type\_id\_new, color\_detail);  
end; //  
  
delimiter ;  
  
# ------------------------------------------------------------------------------------------------  
call *insert\_concrete\_detail*('Не существующий тип1', 'Белый1');  
call *insert\_concrete\_detail*('Брюки', 'Розовый');  
  
  
# details list  
select cd.id\_concrete\_detail, cd.colorDetail, td.id\_type\_detail, td.nameType  
from concrete\_detail cd  
 right join type\_detail td  
 on cd.id\_type\_detail = td.id\_type\_detail  
group by cd.id\_concrete\_detail;  
# ================================================================================================  
# 2

##### # — удаление с очисткой справочников

# - если после удаления данных из дочерней у строки родительской больше нет зависимых  
# - удаление данных из родительской таблицы  
#  
# (удаляется информация о студенте,  
# если в его группе нет больше студентов,  
# запись удаляется из таблицы с перечнем групп);  
  
# удалить деталь;  
# если (нет деталей этого типа):  
# удалить этот тип;  
  
delimiter //  
  
create procedure *delete\_type\_detail*(id\_concrete\_detail\_delete int) begin  
 declare id\_type\_detail\_delete int;  
 select cd.id\_type\_detail into id\_type\_detail\_delete from concrete\_detail cd  
 where cd.id\_concrete\_detail = id\_concrete\_detail\_delete;  
  
 delete from concrete\_detail cd where cd.id\_concrete\_detail = id\_concrete\_detail\_delete;  
 if not exists(select \* from concrete\_detail cd where cd.id\_type\_detail = id\_type\_detail\_delete) then  
 delete from type\_detail td where td.id\_type\_detail = id\_type\_detail\_delete;  
 end if;  
  
end;  
  
delimiter ;  
  
# ------------------------------------------------------------------------------------------------  
call *delete\_type\_detail*(1);  
call *delete\_type\_detail*(2);  
call *delete\_type\_detail*(21);  
  
# details list  
select cd.id\_concrete\_detail, cd.colorDetail, td.id\_type\_detail, td.nameType  
from concrete\_detail cd  
 right join type\_detail td  
 on cd.id\_type\_detail = td.id\_type\_detail  
group by cd.id\_concrete\_detail;  
  
# ================================================================================================  
# 3

##### # — каскадное удаление

# (удаление всех зависимых данных)  
  
# удалить связи "одежда" - "конкретная деталь"  
# удалить все конкретные детали;  
# удалить все типы деталей;  
  
delimiter //  
  
create procedure *cascade\_delete\_type\_detail*(id\_type\_detail\_delete int) begin  
  
 delete from clothes\_concrete\_detail ccd where ccd.id\_concrete\_detail in  
 (select cd.id\_concrete\_detail from concrete\_detail cd where id\_type\_detail = id\_type\_detail\_delete);  
  
 delete from concrete\_detail cd where id\_type\_detail = id\_type\_detail\_delete;  
  
 delete from type\_detail td where id\_type\_detail = id\_type\_detail\_delete;  
  
end;  
  
delimiter ;  
  
# ------------------------------------------------------------------------------------------------  
call *cascade\_delete\_type\_detail*(1);  
  
# details list  
select c.id\_clothes, c.nameClothes,  
 cd.id\_concrete\_detail, cd.colorDetail,  
 td.id\_type\_detail, td.nameType  
from concrete\_detail cd  
 right join type\_detail td on cd.id\_type\_detail = td.id\_type\_detail  
 join clothes\_concrete\_detail ccd on cd.id\_concrete\_detail = ccd.id\_concrete\_detail  
 right join clothes c on ccd.id\_clothes = c.id\_clothes  
order by id\_type\_detail;  
  
# ================================================================================================  
# 4

##### # — вычисление и возврат значения агрегатной функции

# (т.к. агрегатная функция дает единственный результат)  
# (задача - вернуть данные из процедуры/функции)  
  
delimiter //  
  
create procedure *count\_clothes*(out c\_clothes int) begin  
 select ifnull(count(id\_clothes),0) into c\_clothes from clothes;  
  
end;  
  
delimiter ;  
  
# ------------------------------------------------------------------------------------------------  
call *count\_clothes*(@c);  
select @c;  
  
# ================================================================================================  
# 5

##### # — формирование статистики во временной таблице

# (задача - работа с временными таблицами)  
  
set sql\_safe\_updates = 0;  
drop procedure if exists *authors\_statistic*;  
  
delimiter //  
  
create procedure *authors\_statistic*() begin  
  
 drop table if exists stat;  
 create temporary table if not exists stat(  
 id\_stat int primary key not null auto\_increment,  
 id\_author int,  
 count\_clothes\_author int,  
 avg\_count\_clothes\_author double default 0,  
 diff\_count\_clothes\_author double default 0  
 );  
  
 insert into stat (id\_author, count\_clothes\_author)  
 select id\_author, count(ca.id\_clothes) as count\_clothes  
 from clothes c left join clothes\_author ca  
 on c.id\_clothes = ca.id\_clothes  
 group by id\_author;  
  
 update stat set avg\_count\_clothes\_author =  
 (select avg(q.count\_clothes) from  
 (select count(ca.id\_clothes) as count\_clothes  
 from clothes c left join clothes\_author ca  
 on c.id\_clothes = ca.id\_clothes  
 group by id\_author)q);  
  
 update stat s set diff\_count\_clothes\_author = s.count\_clothes\_author - s.avg\_count\_clothes\_author  
 where id\_stat > 0;  
  
 select distinct  
 stat.id\_author,  
 stat.count\_clothes\_author,  
 stat.avg\_count\_clothes\_author,  
 stat.diff\_count\_clothes\_author from stat;  
  
 drop table stat;  
end;//  
  
delimiter ;  
  
  
# ------------------------------------------------------------------------------------------------  
call *authors\_statistic*();  
set sql\_safe\_updates = 0;