МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

**«Вологодский государственный университет»**

**Институт математики, естественных и компьютерных наук**

**Информатика и вычислительная техника**

**КУРСОВОЙ ПРОЕКТ**

Система продажи билетов в кинотеатре.

Дисциплина: «Базы данных»

Направление подготовки: 09.03.01. Информатика и вычислительная техника

|  |  |
| --- | --- |
| Руководитель | Ржеуцкая С.Ю. |
| Выполнили студенты | Пчелкина Ольга |
| Группа, курс | ВМ-31 |
| Дата сдачи | \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата защиты | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись преподавателя)* |

Вологда

2021 г.

Оглавление

[1. Анализ предметной области 3](#_Toc91538354)

[1.1 Постановка задачи 3](#_Toc91538355)

[1.2 Бизнес правила 3](#_Toc91538356)

[1.3 Лица, взаимодействующие с системой 4](#_Toc91538357)

[1.4 Функциональные требования 5](#_Toc91538358)

[2. Проектирование базы данных 5](#_Toc91538359)

[2.1 Логическая модель базы данных. 5](#_Toc91538360)

[2.2 Реляционная модель базы данных. 8](#_Toc91538361)

[3. Проектирование хранилища исторических данных 9](#_Toc91538362)

[3.1 Требования к хранилищу 9](#_Toc91538363)

[3.2 Логическая и реляционная модели хранилища данных 9](#_Toc91538364)

[4. Создание и наполнение базы данных 10](#_Toc91538365)

[4.1 Создание таблиц, последовательностей, триггеров, дополнительных ограничений. 10](#_Toc91538366)

[4.2 Наполнение таблиц базы данных. 11](#_Toc91538367)

[4.3 Создание хранилища базы данных. 12](#_Toc91538368)

[4.4 Анализ схемы оперативной БД на соответствие третьей нормальной формы. 13](#_Toc91538369)

[4.5 Анализ схемы хранилища соответствие третьей нормальной формы. 13](#_Toc91538370)

[5. Разработка запросов к индивидуальной базе и хранилищу данных. 13](#_Toc91538371)

[5.1 Запросы INSERT. 13](#_Toc91538372)

[5.2 Запросы UPDATE. 16](#_Toc91538373)

[5.3 Запросы на заполнение хранилища. 18](#_Toc91538374)

[5.4 Запросы на выборку. 19](#_Toc91538375)

[5.5 Запросы к архиву. 22](#_Toc91538376)

[Заключение 24](#_Toc91538377)

[Список использованных источников 25](#_Toc91538378)

[Приложение 26](#_Toc91538379)

1. Анализ предметной области
   1. Постановка задачи

Вариант 13

Система продажи билетов в кинотеатр. Система предназначена для продажи билетов в кинотеатр с одним кинозалом (в зале определённое количество рядов с определённым количеством мест в каждом ряду, возможно, в разных рядах и разное количество мест). Каждый фильм идет в течение определенного срока на различных сеансах в соответствии с расписанием показов, которое составляет администратор, он же вводит это расписание в базу данных. Билеты можно приобрести на любой показ, введённый в базу данных, заранее. Цена проданных билетов рассчитывается автоматически, она определяется как базовая цена на данный фильм, умноженная на некоторые поправочные коэффициенты (на время показа, допустим, утром в будни этот коэффициент равен 1, а на вечерние сеансы и сеансы в выходные дни его значение будет больше 1, на место – для vip-мест есть свой коэффициент). На отдельные показы администратор может назначить дополнительные поправочные коэффициенты (например, на все сеансы первого дня показа).

Базовая функциональность:

- ввод нового фильма, взятого в прокат, и информации обо всех показах (сеансах) данного фильма

- продажа билетов при наличии свободных мест на заданный сеанс

- возврат билетов

Данные о фильмах, показ которых уже закончен, собираются в архиве (хранилище данных). Хранилище используется администрацией для анализа деятельности кинотеатра по любым возможным показателям.

1.2 Бизнес правила

1. Кинотеатр имеет определенное количество фильмов проката. Каждый фильм имеет ограниченное количество прокатных дней, возрастное ограничение, длительность показа. Каждый фильм принадлежит определенному жанру, имеет название.
2. В зале кинотеатра определенное число обычных и vip мест на все сеансы.
3. Расписание сеансов составляется на весь прокатный период фильма, для каждого сеанса задаются дата, часы и минуты показа. После того, как расписание введено, клиенты могут купить билеты на любой из сеансов.
4. Цена билетов рассчитывается автоматически, она определяется как базовая цена на данный фильм, умноженная на некоторые поправочные коэффициенты (время сеанса, граничное время между утренними и вечерними сеансами задается, будни или выходной день, место в зале, поправочный коэффициент на место привязан к ряду). На отдельные показы администратор может назначить дополнительные поправочные коэффициенты.
5. Билеты продаются заранее на все сеансы в прокате, при покупке билета клиенту или кассиру выдается список свободных мест на сеанс, из которого он может выбрать место. Билет можно вернуть в любой момент до начала сеанса за полную стоимость.
6. На каждый сеанс можно продать не больше билетов, чем вмещает зал.
7. В момент начала показа диспетчер заносит данные о сеансе в архив.

1.3 Лица, взаимодействующие с системой

Основные пользователи системы — кассиры кинотеатра. Все кассиры работают с одним и тем же приложением. Их основные требования к интерфейсу — максимальная простота и удобство работы, обеспечение приемлемой скорости, сведение к минимуму количества возможных человеческих ошибок.

Администраторобязан поддерживать данные в актуальном состоянии. Для него должно быть создано отдельное приложение, особых требований к интерфейсу не предъявляется.

Для сбора данных о фильмах, показ которых уже закончен, создадим еще одно небольшое приложение для диспетчера.

Любой из посетителей мог бы воспользоваться электронным справочником, отрабатывающим запросы о расписании фильмов и о наличии билетов на конкретные сеансы. Разумеется, приложение-справочник должно работать с базой данных в режиме «только для чтения».

1.4 Функциональные требования

Функции администратора:

1. Своевременное обновление информации о фильмах в прокате
2. Составление расписания показов
3. Назначение дополнительны поправочных коэффициентов на отдельные показы

Функции кассира:

1. Определение, имеется ли в наличии необходимое число билетов на выбранный сеанс.
2. Продажа или бронирование билетов с фиксацией в базе данных
3. Возврат билетов

Функции диспетчера:

Отправка данных в электронный архив.

Функции посетителей:

Быстрый поиск доступной для них информации из базы данных.

1. Проектирование базы данных

Для разработки используем Case-средство Oracle SQL Developer Data Modeler, которое позволит по сформированной нами логической модели базы данных кинотеатра автоматически получить реляционную модель и сценарий на языке SQL для создания базы данных.

2.1 Логическая модель базы данных.

На основании бизнес-правил выделим следующие сущности предметной области и их атрибуты.

1. **bilets** (билеты)

ID\_bilet (ИД билета, суррогатный ключ)

price (цена за билет, рассчитывается запросами)

ID\_session (ИД сеанса, внешний ключ)

ID\_seats (ИД места, внешний ключ)

2. **sessions** (сеансы)

ID\_session (ИД, суррогатный ключ)

date\_session (дата сеанса)

time\_session (время сеанса)

week\_day (день недели)

factor\_session (логическая переменная, true – есть добавочный коэффициент на сеанс, false – нет добавочного коэффициента на сеанс)

number\_free\_seats (количество свободных мест, для упрощения запросов при покупке билетов)

ID\_rent (ИД проката фильма, внешний ключ)

3. **seats** (автобусы)

ID\_seats (ИД места, суррогатный ключ)

num\_seats (номер места)

num\_row (номер ряда)

status (статус vip или обычное место)

free (логическая переменная, true – место свободно, false – место занято)

4. **rents\_films** (прокаты фильмов)

ID\_rent (ИД, суррогатный ключ)

start\_date (дата начала проката)

end\_date (дата окончания проката)

box\_office (кассовые сборы)

number\_sessions (количество сеансов)

ID\_film (ИД фильма, внешний ключ)

5. **films** (фильмы)

ID\_film (ИД, суррогатный ключ)

title\_film (название фильма)

duration (длительность фильма)

age\_limit (возрастное ограничение)

ID\_genres (ИД жанра, внешний ключ)

6. **genres** (жанры)

ID\_genres (ИД, суррогатный ключ)

name\_genres (название жанра)

7. **constants** (константы, для расчета стоимости билета)

factor\_session (коэффициент на сеанс)

factor\_seat (коэффициент на место)

factor\_time (коэффициент на время сеанса)

factor\_week\_day (коэффициент на день недели сеанса)

price (стоимость билета без надбавок)

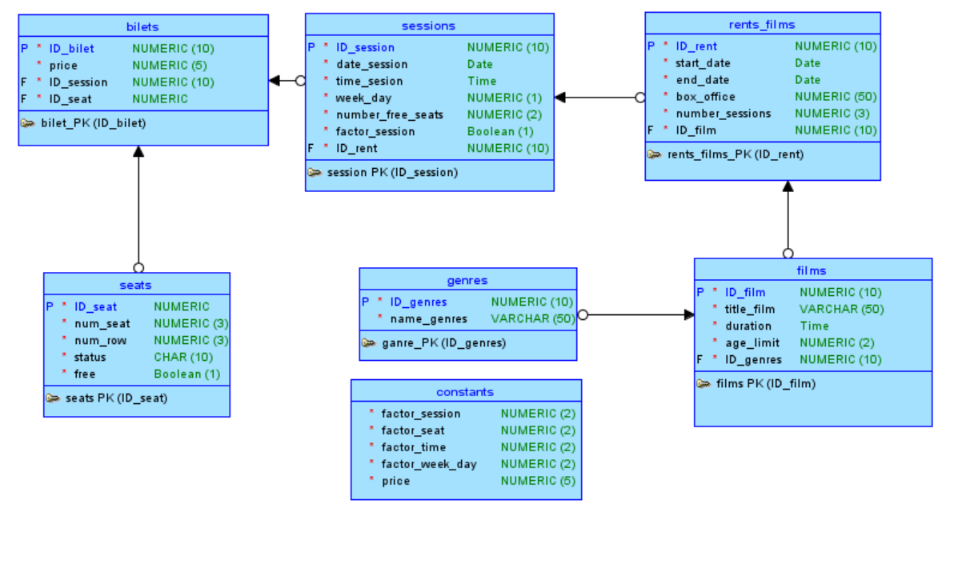


Рис. 1. Логическая модель базы данных

2.2 Реляционная модель базы данных.

Основные типы данных были выбраны с учетом используемой СУБД. Большинство данных типа NUMBER, так как они числовые, так же данные, отвечающие за время, пришлось заменить на числовые. Для дат используется тип данных – DATE. Для логических полей VARCHAR.

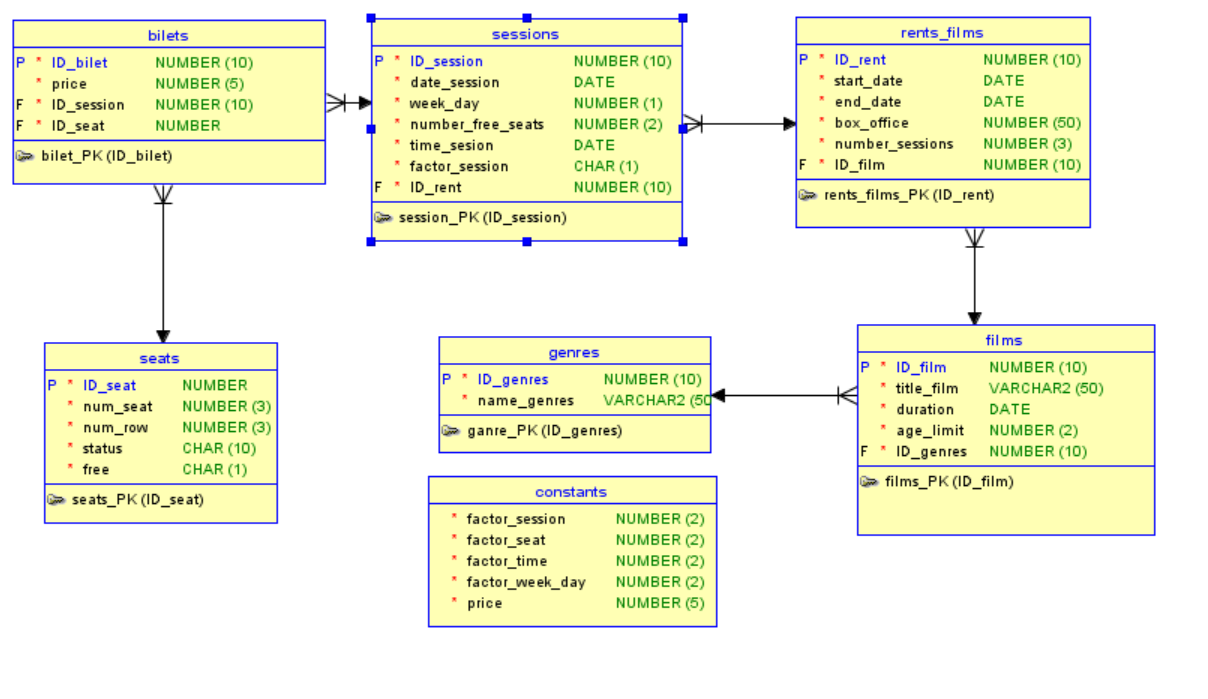


Рис.2 Реляционная модель базы данных Кинотеатра

1. Проектирование хранилища исторических данных

3.1 Требования к хранилищу

С помощью архива можно отследить выручку и наполняемость зала по каждому сеансу, в конкретный день и время. С помощью накопленных данных можно сформировать такие аналитические отчеты, как в какие дни и время наполняемость зала максимальная, на какие фильмы, прокаты каких фильмов набрали большие кассовые сборы, какие жанры более актуальны.

3.2 Логическая и реляционная модели хранилища данных

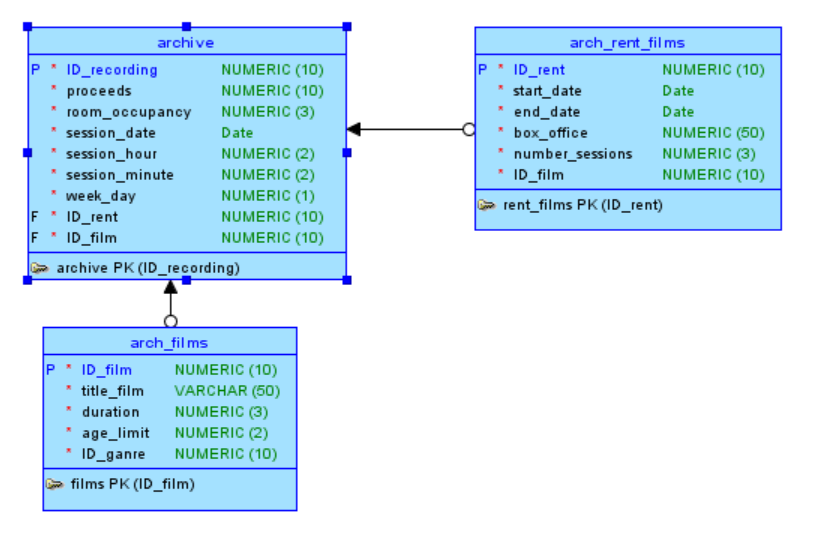


Рисунок 3.1. Реляционная модель архива базы данных

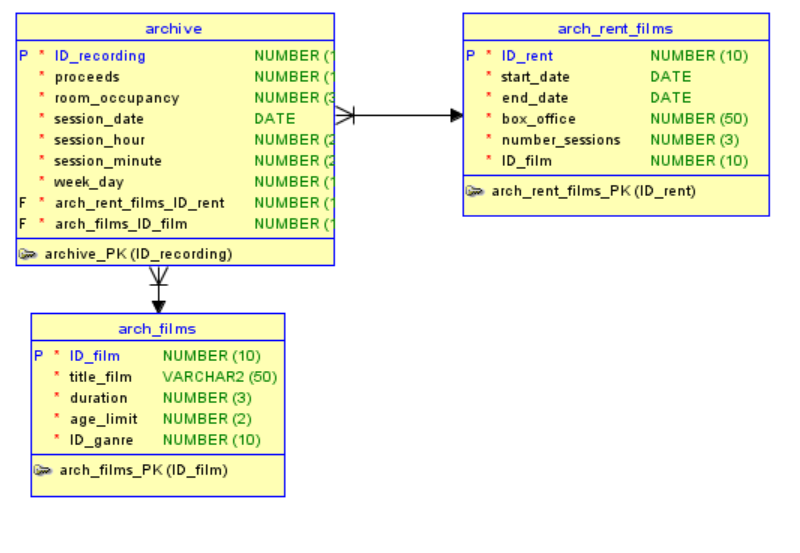


Рисунок 3.2. Логическая модель архива базы данных

1. Создание и наполнение базы данных

4.1 Создание таблиц, последовательностей, триггеров, дополнительных ограничений.

Сценарий создания таблиц для базы данных приведен в приложении 1.

Чтобы автоматически получать уникальные значения первичных ключей, воспользуемся имеющимся в Oracle объектом ***Sequence***.

create sequence bilet\_seq;

Для обращения к значениям последовательности в выражении SQL используются псевдостолбцы ***currval*** и ***nextval***. ***Currval*** возвращает текущее значение, ***NextVal*** инкрементирует текущее значение и возвращает результат, при этом он становится текущим значением. Чтобы использовать последовательность для генерации уникальных значений первичного ключа, необходимо создать триггер, срабатывающий перед добавлением новой записи.

create trigger bilet\_keys  
before insert on bilets  
for each row  
begin  
select bilet\_seq.nextval into :new.id\_bilet from dual;  
end;/

Здесь строчка ***for each row*** показывает, что триггер будет выполняться каждый раз при вставке новой записи. Выражение ***:new*** представляет собой новую запись, которая после выполнения триггера окажется в таблице bilets. Специальная системная таблица *Dual* используется для получения констант в команде Select.

Ограничение на ввод дня недели:  
alter table sessions add  
constraint weekDaySessionsLimit check ((week\_day >= 1) and (week\_day <=7));

Ограничение на ввод часов:  
alter table sessions add  
constraint hourSessionLimit check ((hour >= 0) and (hour <=23));

Ограничение на ввод минут:  
alter table sessions add  
constraint minuteSessionLimit check ((minute >= 0) and (minute <=59));

Ограничение на ввод номера места:   
alter table seats add  
constraint numSeeatSeatsLimit check ((num\_seat >= 1) and (num\_seat <=15));

Ограничение на воод номера ряда:  
alter table seats add  
constraint numRowSeatsLimit check ((num\_row >= 1) and (num\_row <=10));

Ограничение на ввод возрастного ограничения:  
alter table films add  
constraint ageLimitFilmsLimit check ((age\_limit >= 0) and (age\_limit <=18));

4.2 Наполнение таблиц базы данных.

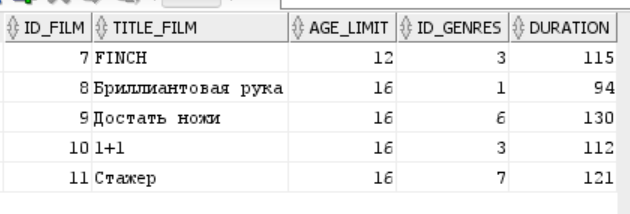


Рисунок 4.1. Заполнение таблицы «Фильмы»

Рисунок 4.2. Заполнение таблицы «Жанры»

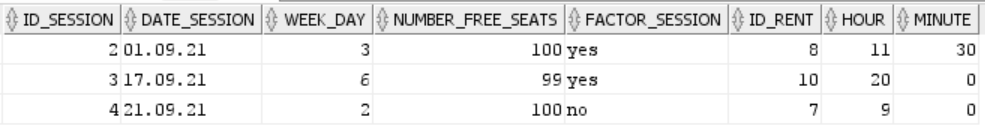


Рисунок 4.3. Заполнение таблицы «Сеансы»

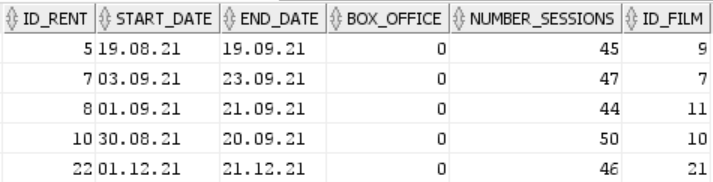


Рисунок 4.4. Заполнение таблицы «Прокаты»

4.3 Создание хранилища базы данных.

Сценарий таблиц хранилища данных приведен в приложении 2.



Рисунок 4.5. Создание таблиц хранилища

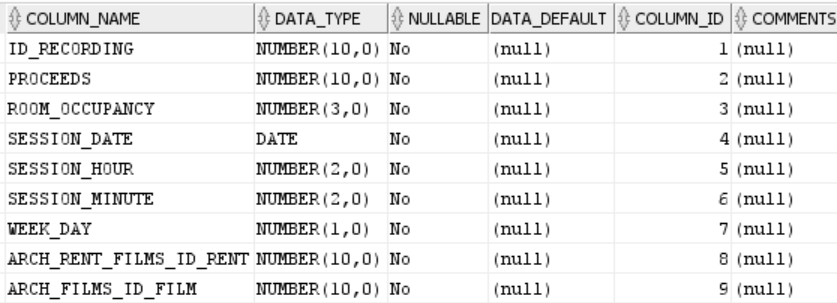


Рисунок 4.6. Создание таблицы хранилища «Архив»

4.4 Анализ схемы оперативной БД на соответствие третьей нормальной формы.

Неключевой столбец в таблице SESSIONS NUMBER\_FREE\_SEATS. Вычисляется путем запроса при покупке билета, вычитается количество купленных билетов из общего числа мест в зале.

Столбцы NUMBER\_SESSION и BOX\_OFFICE, зависимы от таблицы SESSIONS и BILETS. Рассчитываются запросами.

4.5 Анализ схемы хранилища соответствие третьей нормальной формы.

Зависимыми в хранилище БД являются все неключевые столбцы. Их заполнение будет производится путем запросов. Информация в основном будет браться из таблицы SESSIONS

1. Разработка запросов к индивидуальной базе и хранилищу данных.

5.1 Запросы INSERT.

Заполнение таблицы SEATS. Заполняется в цикле, так количество мест в ряду изменяется от 1 до 10.

Заполнение таблицы SEATS 1 рад, 10 мест не “vip”:

DECLARE  
numSeat number := 1 ;  
numRow number;  
state varchar(10);   
BEGIN  
WHILE (numSeat < 11)  
LOOP  
numRow := 1;  
state := 'normal';  
INSERT INTO seats  
(NUM\_SEAT,NUM\_ROW,STATUS)  
 values(numSeat,numRow,state);  
numSeat := numSeat +1;  
END LOOP;  
COMMIT;  
END;

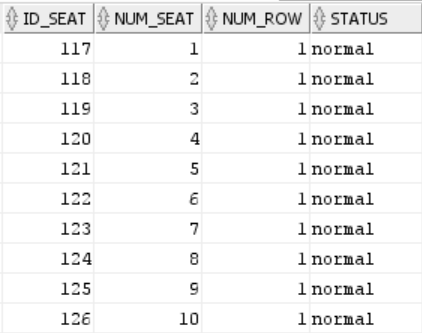


Рисунок 5.1. Заполнение таблицы «Места»

Заполнение таблицы SEATS 2 рад, 5 мест “vip”:

DECLARE  
numSeat number := 3;  
numRow number;  
state varchar(10);   
BEGIN  
WHILE (numSeat < 8)  
LOOP  
numRow := 2;  
state := 'vipl';  
INSERT INTO seats  
(NUM\_SEAT,NUM\_ROW,STATUS)  
 values(numSeat,numRow,state);  
numSeat := numSeat +1;  
END LOOP;  
COMMIT;  
END;

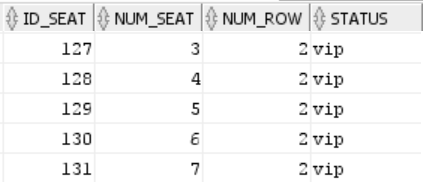


Рисунок 5.2. Заполнение таблицы «Места»

Заполнение таблицы BILETS проводится с помощью функции из-за необходимости высчитывать цену билета в зависимости от каких-либо факторов.

Функция для расчета стоимости билета:

CREATE OR REPLACE FUNCTION GetPriceBilet  
 (price NUMBER, seatStatus CHAR, sessionFactor VARCHAR2, weekDaySession NUMBER, hourSession NUMBER,  
 factorSession FLOAT, factorWeekDay FLOAT, factorSeat FLOAT, factorTime FLOAT)

RETURN NUMBER  
IS priceBilet NUMBER;  
BEGIN

priceBilet:= price;  
IF sessionFactor = 'yes' THEN priceBilet := priceBilet \* factorSession; END IF;  
IF seatStatus = 'vip' THEN priceBilet := priceBilet \* factorSeat; END IF;  
IF weekDaySession > 5 THEN priceBilet := priceBilet \* factorWeekDay; END IF;  
IF hourSession > 17 THEN priceBilet := priceBilet \* factorTime; END IF;

RETURN priceBilet;

END;

Заполнение таблицы BILETS:

INSERT INTO BILETS(ID\_SESSION, ID\_SEAT, FREE, PRICE)

SELECT sessions.id\_session, seats.id\_seat, 'true', GETPRICEBILET(CONSTANTS.PRICE, SEATS.STATUS, SESSIONS.FACTOR\_SESSION,

SESSIONS.week\_day, sessions.hour, constants.factor\_session, constants.factor\_week\_day, constants.factor\_seat, constants.factor\_time)

FROM SESSIONS, SEATS, CONSTANTS

WHERE sessions.id\_session = 3;

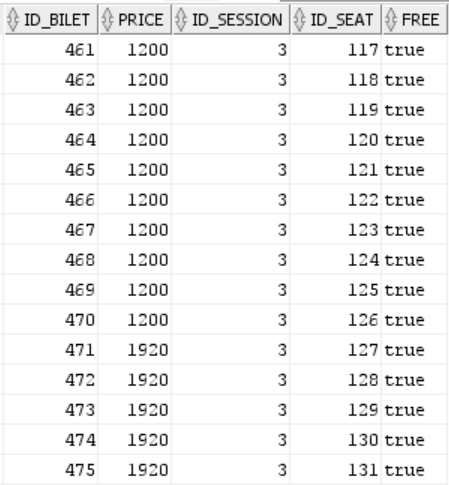


Рисунок 5.3. Заполнение таблицы «Билеты»

Всего данных в таблице BILETS – 15.

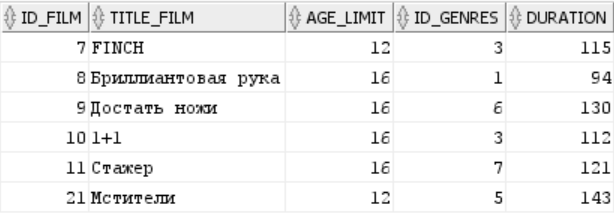
Запрос на добавление строки в таблицу GANRES:

INSERT INTO GENRES(NAME\_GENRES)  
VALUES ('Биография');



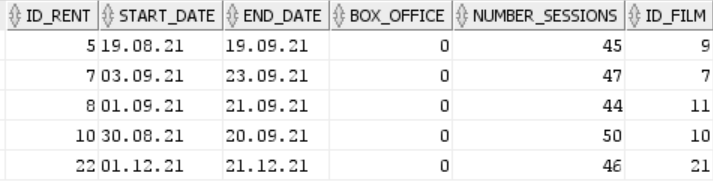
Запрос на добавление строки в таблицу FILMS:

INSERT INTO FILMS(AGE\_LIMIT, DURATION, TITLE\_FILM, ID\_GENRES)  
VALUES (12, 143, 'Мстители', 5);



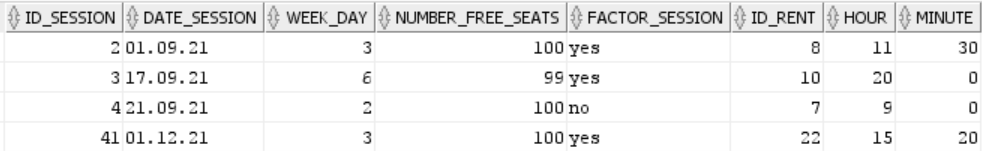
Запрос на добавление строки в таблицу RENTS\_FILMS:

INSERT INTO RENTS\_FILMS(BOX\_OFFICE,START\_DATE,END\_DATE,NUMBER\_SESSIONS,ID\_FILM)  
VALUES (0, '01/12/2021', '21/12/2021', 46, 21);



Запрос на добавление строки в таблицу SESSIONS:

INSERT INTO SESSIONS(DATE\_SESSION,FACTOR\_SESSION,HOUR,MINUTE,NUMBER\_FREE\_SEATS,ID\_RENT,WEEK\_DAY)  
VALUES ('01/12/2021', 'yes', 15, 20, 100, 22, 3);



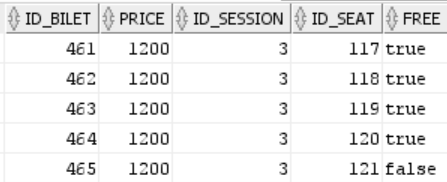
5.2 Запросы UPDATE.

Следующие запросы обновляют необходимые данные в базе данных при покупке билета.

Запрос на покупку билета:

UPDATE billets  
 SET bilets.free = 'false' WHERE id\_bilet = 352;

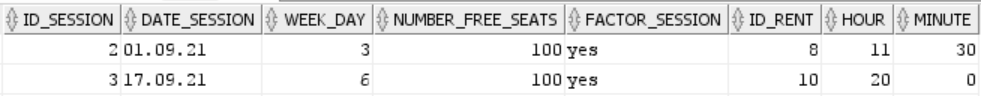
до: после:

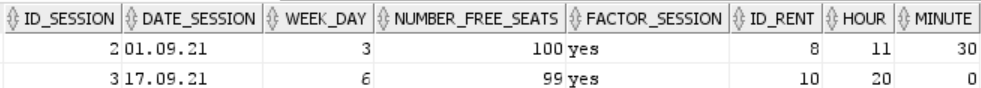
Запрос на изменение количества свободных мест на сеанс:

UPDATE sessions   
 SET number\_free\_seats = number\_free\_seats - (SELECT COUNT(id\_bilet) FROM bilets, sessions  
WHERE bilets.free = 'false' AND   
 sessions.id\_session = bilets.id\_session)  
 WHERE id\_session = (SELECT DISTINCT sessions.id\_session FROM sessions   
 INNER JOIN bilets ON sessions.id\_session = bilets.id\_session   
 WHERE bilets.free = 'false');

До:



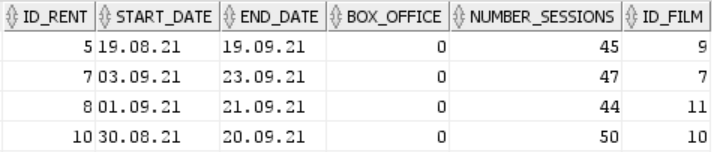
После:



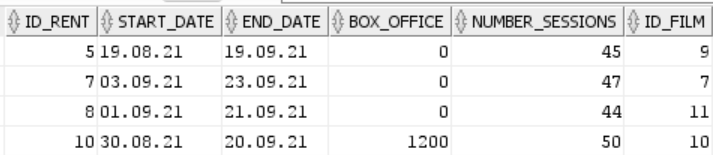
Запрос на изменение выручки за прокат фильма:

UPDATE rents\_films  
 SET box\_office = box\_office + (SELECT SUM(bilets.PRICE) FROM bilets, sessions, rents\_films   
 WHERE bilets.free = 'false' AND   
 sessions.id\_session = bilets.id\_session AND sessions.id\_rent = rents\_films.id\_rent)  
 WHERE id\_rent = (SELECT DISTINCT rents\_films.id\_rent FROM sessions, bilets, rents\_films  
 WHERE bilets.free = 'false' AND sessions.id\_session = bilets.id\_session AND  
 rents\_films.id\_rent = sessions.id\_rent);

До:



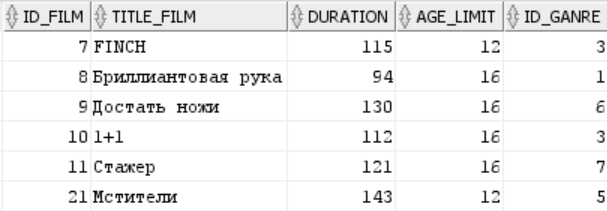
После:



5.3 Запросы на заполнение хранилища.

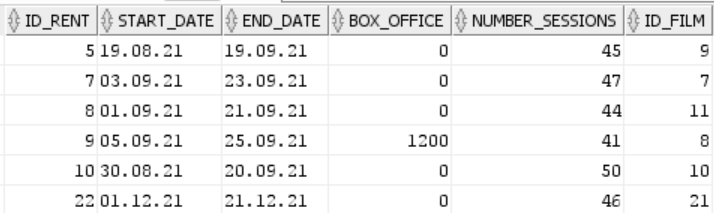
Заполнение таблицы хранилища с данными о фильмах:

INSERT INTO ARCH\_FILMS  
SELECT id\_film, title\_film,age\_limit, id\_genres, duration FROM films;



Заполнение таблицы хранилища с данными о прокатах:

INSERT INTO ARCH\_RENT\_FILMS  
SELECT id\_rent, start\_date, end\_date, box\_office, number\_sessions, id\_film FROM rents\_films;



Добавление в архив сеанса с номером 2:

INSERT INTO  
ARCHIVE(PROCEEDS,ROOM\_OCCUPANCY,SESSION\_DATE,SESSION\_HOUR,SESSION\_MINUTE,  
 WEEK\_DAY, ARCH\_RENT\_FILMS\_ID\_RENT, ARCH\_FILMS\_ID\_FILM)  
SELECT ARCH\_RENT\_FILMS.BOX\_OFFICE, (100 - sessions.number\_free\_seats),  
sessions.date\_session, sessions.hour, sessions.minute, sessions.week\_day, arch\_rent\_films.id\_rent, arch\_films.id\_film  
FROM ARCH\_RENT\_FILMS, ARCH\_FILMS, sessions WHERE ARCH\_RENT\_FILMS.ID\_RENT = sessions.id\_rent AND  
ARCH\_FILMS.ID\_FILM = arch\_rent\_films.id\_film AND sessions.id\_session = 2;



Удаление сеанса, который мы занесли в архив из основной базы данных:

DELETE FROM sessions WHERE id\_session = 2;

Добавление в архив проката с номером 10:

INSERT INTO ARCHIVE(PROCEEDS,ROOM\_OCCUPANCY,SESSION\_DATE,SESSION\_HOUR,SESSION\_MINUTE,  
 WEEK\_DAY, ARCH\_RENT\_FILMS\_ID\_RENT, ARCH\_FILMS\_ID\_FILM)  
SELECT ARCH\_RENT\_FILMS.BOX\_OFFICE,   
 (100 - sessions.number\_free\_seats), sessions.date\_session, sessions.hour, sessions.minute,  
 sessions.week\_day, arch\_rent\_films.id\_rent, arch\_films.id\_film  
FROM ARCH\_RENT\_FILMS, ARCH\_FILMS, sessions WHERE ARCH\_RENT\_FILMS.ID\_RENT = sessions.id\_rent AND  
ARCH\_FILMS.ID\_FILM = arch\_rent\_films.id\_film AND arch\_rent\_films.id\_rent = 10;



Удаление проката, который мы занесли в архив из основной базы данных:

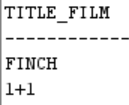
DELETE FROM rents\_films WHERE id\_rent = 10;

5.4 Запросы на выборку.

Запрос на выбор всех фильмов с определенным жанром:

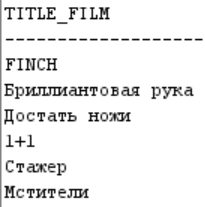
SELECT films.title\_film FROM films INNER JOIN genres ON films.ID\_GENRES = genres.id\_genres

WHERE genres.name\_genres = 'Драма';



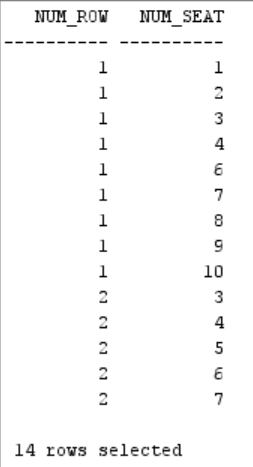
Запрос на выбор всех фильмов с возрастным ограничением меньше 18 лет:

SELECT title\_film FROM films WHERE age\_limit < 18;



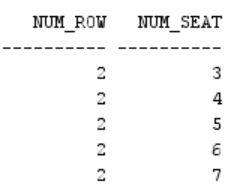
Запрос на вывод свободных мест на конкретный сеанс (всего 15 мест, занято одно):

SELECT seats.num\_row, seats.num\_seat FROM seats, bilets   
WHERE seats.id\_seat = bilets.id\_seat AND  
bilets.free = 'true' AND bilets.id\_session = 3;



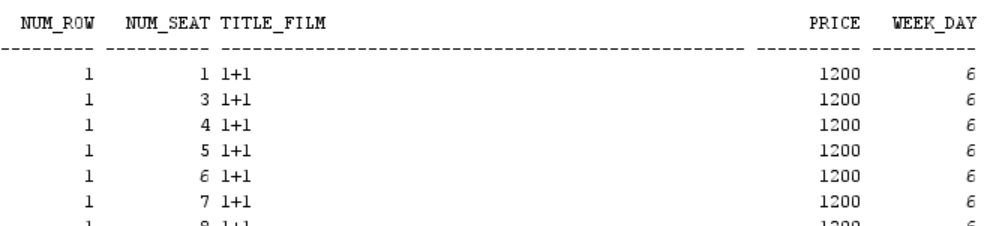
Запрос на вывод свободных ‘vip’ мест на конкретный сеанс:

SELECT seats.num\_row, seats.num\_seat FROM seats   
INNER JOIN bilets ON seats.id\_seat = bilets.id\_seat   
WHERE bilets.free = 'true' AND bilets.id\_session = 3 AND seats.status = 'vip';



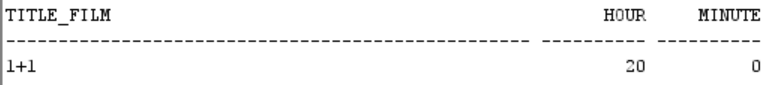
Запрос на вывод свободных свободных мест на конкретный фильм стоимостью меньше 1500, в выходной день:

SELECT seats.num\_row, seats.num\_seat, films.title\_film, bilets.price, sessions.week\_day FROM sessions, films, rents\_films, seats, billets  
WHERE bilets.free = 'true' AND bilets.id\_session = 3 AND bilets.price < 1500 AND   
bilets.id\_seat = seats.id\_seat AND sessions.id\_session = bilets.id\_session AND sessions.week\_day > 5 AND  
rents\_films.id\_rent = sessions.id\_rent AND films.id\_film = rents\_films.id\_film AND films.title\_film = '1+1';



Запрос на вывод сеансов в определенный день:

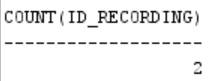
SELECT DISTINCT films.title\_film, sessions.hour, sessions.minute, bilets.price FROM sessions, films, rents\_films, billets  
WHERE rents\_films.id\_rent = sessions.id\_rent AND films.id\_film = rents\_films.id\_film AND   
sessions.date\_session = '17/09/2021';



5.5 Запросы к архиву.

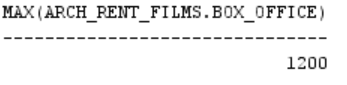
Запрос на вывод количества сеансов в архиве:

SELECT COUNT(id\_recording) FROM archive



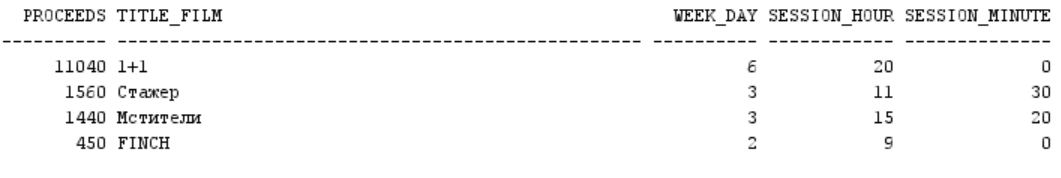
Запрос на вывод максимального кассового сбора:

SELECT MAX(arch\_rent\_films.box\_office)FROM archive,   
ARCH\_RENT\_FILMS WHERE   
archive.ARCH\_RENT\_FILMS\_ID\_RENT = ARCH\_RENT\_FILMS.ID\_RENT



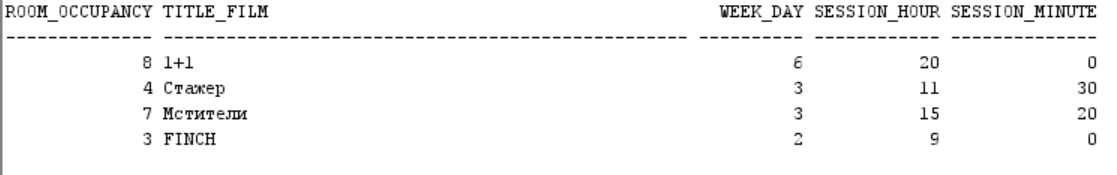
Аналитический запрос, показывает выручку за сеансы, благодаря сортировке, можно увидеть в какое время, день и какой фильм набрал максимальный/минимальный кассовый сбор.

SELECT archive.PROCEEDS, arch\_films.title\_film, archive.week\_day, archive.session\_hour,  
archive.session\_minute FROM archive, ARCH\_FILMS WHERE   
archive.ARCH\_FILMS\_id\_film = ARCH\_FIlMS.ID\_FiLM ORDER BY archive.PROCEEDS DESC;



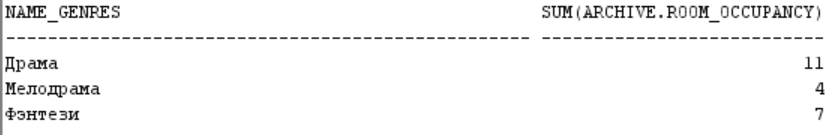
Запрос аналогичный предыдущему, позволяет составить статистику по заполняемости зала, в зависимости от дня недели и времени.

SELECT archive.ROOM\_OCCUPANCY, arch\_films.title\_film, archive.week\_day, archive.session\_hour,  
archive.session\_minute FROM archive, ARCH\_FILMS WHERE   
archive.ARCH\_FILMS\_id\_film = ARCH\_FIlMS.ID\_FiLM ORDER BY archive.PROCEEDS DESC;



Запрос на выявление самого популярного жанра за прошедшие прокаты (выводит сколько людей посетили определенный жанр:

SELECT GENRES.NAME\_GENRES, SUM(archive.ROOM\_OCCUPANCY) FROM archive,  
ARCH\_FILMS, GENRES WHERE   
archive.ARCH\_FILMS\_id\_film = ARCH\_FIlMS.ID\_FiLM AND  
ARCH\_FIlMS.id\_ganre = genres.id\_genres GROUP BY GENRES.NAME\_GENRES;



Заключение

В ходе курсовой работы мы создали базу данных для кинотеатра. Познакомились с логической и реляционной моделями баз данных. Создали хранилище базы данных для последующего анализа работы кинотеатра. Наполнили базу и хранилище данными. Написали запросы к данным в хранилище и основной базе. Выполнили добавление, обновление и удаление данных из таблиц.

Список использованных источников

1. Андрианов И.А., Ржеуцкая С.Ю. Базы данных. Программирование и администрирование: учеб. пособие. – Вологда: ВоГУ, 2018. – 71 с.

2. Ржеуцкая, С.Ю. Базы данных. Теоретические и языковые основы: Учеб. пособие. / С.Ю. Ржеуцкая. – Вологда : ВоГУ, 2016. – 112 с.

3. SQLBolt [Электронный ресурс]. – Режим доступа: https://sqlbolt.com/lesson/introduction. – Дата доступа: 24.12.2021.

4. Techrocks [Электронный ресурс]. – Режим доступа: https://techrocks.ru/2021/04/13/full-outer-join-in-sql/. – Дата доступа: 22.12.2021.

5. Habr [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/480838/. – Дата доступа: 20.12.2021.

Приложение

Приложение 1 – Сценарий создания базы данных.

CREATE TABLE billets(   
ID\_bilet NUMBER (10) NOT NULL ,  
price NUMBER (5) NOT NULL ,  
ID\_session NUMBER (10) NOT NULL ,  
 ID\_seat NUMBER NOT NULL ) ;  
ALTER TABLE bilets ADD CONSTRAINT bilet\_PK PRIMARY KEY ( ID\_bilet ) ;

CREATE TABLE constants(  
factor\_session NUMBER (2) NOT NULL ,  
factor\_seat NUMBER (2) NOT NULL ,  
factor\_time NUMBER (2) NOT NULL ,  
factor\_week\_day NUMBER (2) NOT NULL ,  
price NUMBER (5) NOT NULL) ;

CREATE TABLE films(  
ID\_film NUMBER (10) NOT NULL ,  
title\_film VARCHAR2 (50) NOT NULL ,  
duration DATE NOT NULL ,  
age\_limit NUMBER (2) NOT NULL ,  
ID\_genres NUMBER (10) NOT NULL) ;  
ALTER TABLE films ADD CONSTRAINT films\_PK PRIMARY KEY ( ID\_film ) ;

CREATE TABLE genres(  
ID\_genres NUMBER (10) NOT NULL ,  
name\_genres VARCHAR2 (50) NOT NULL) ;  
ALTER TABLE genres ADD CONSTRAINT ganre\_PK PRIMARY KEY ( ID\_genres ) ;

CREATE TABLE rents\_films(  
ID\_rent NUMBER (10) NOT NULL ,  
start\_date DATE NOT NULL ,  
end\_date DATE NOT NULL ,  
box\_office NUMBER (50) NOT NULL ,  
number\_sessions NUMBER (3) NOT NULL ,  
ID\_film NUMBER (10) NOT NULL) ;  
ALTER TABLE rents\_films ADD CONSTRAINT rents\_films\_PK PRIMARY KEY ( ID\_rent ) ;

CREATE TABLE seats(  
ID\_seat NUMBER NOT NULL ,  
num\_seat NUMBER (3) NOT NULL ,  
num\_row NUMBER (3) NOT NULL ,  
status CHAR (10) NOT NULL ,  
free CHAR (1) NOT NULL) ;  
ALTER TABLE seats ADD CONSTRAINT seats\_PK PRIMARY KEY ( ID\_seat ) ;

CREATE TABLE sessions(  
ID\_session NUMBER (10) NOT NULL ,  
date\_session DATE NOT NULL ,  
week\_day NUMBER (1) NOT NULL ,  
number\_free\_seats NUMBER (2) NOT NULL ,  
time\_sesion DATE NOT NULL ,  
factor\_session CHAR (1) NOT NULL ,  
ID\_rent NUMBER (10) NOT NULL) ;  
ALTER TABLE sessions ADD CONSTRAINT session\_PK PRIMARY KEY ( ID\_session ) ;

ALTER TABLE bilets ADD CONSTRAINT bilet\_seats\_FK FOREIGN KEY ( ID\_seat ) REFERENCES seats ( ID\_seat ) ;  
ALTER TABLE bilets ADD CONSTRAINT bilet\_session\_FK FOREIGN KEY ( ID\_session ) REFERENCES sessions ( ID\_session ) ;  
ALTER TABLE films ADD CONSTRAINT films\_ganre\_FK FOREIGN KEY ( ID\_genres ) REFERENCES genres ( ID\_genres ) ;  
ALTER TABLE rents\_films ADD CONSTRAINT rents\_films\_films\_FK FOREIGN KEY ( ID\_film ) REFERENCES films ( ID\_film ) ;  
ALTER TABLE sessions ADD CONSTRAINT session\_rents\_films\_FK FOREIGN KEY ( ID\_rent ) REFERENCES rents\_films ( ID\_rent ) ;

Приложение 2 – Сценарий создания хранилища данных.

CREATE TABLE arch\_films(  
ID\_film NUMBER (10) NOT NULL ,  
title\_film VARCHAR2 (50) NOT NULL ,  
duration NUMBER (3) NOT NULL ,  
age\_limit NUMBER (2) NOT NULL ,  
ID\_ganre NUMBER (10) NOT NULL) ;  
ALTER TABLE arch\_films ADD CONSTRAINT arch\_films\_PK PRIMARY KEY ( ID\_film ) ;

CREATE TABLE arch\_rent\_films(  
ID\_rent NUMBER (10) NOT NULL ,  
start\_date DATE NOT NULL ,  
end\_date DATE NOT NULL ,  
box\_office NUMBER (50) NOT NULL ,  
number\_sessions NUMBER (3) NOT NULL ,  
ID\_film NUMBER (10) NOT NULL) ;  
ALTER TABLE arch\_rent\_films ADD CONSTRAINT arch\_rent\_films\_PK PRIMARY KEY ( ID\_rent );

CREATE TABLE archive(  
ID\_recording NUMBER (10) NOT NULL ,  
proceeds NUMBER (10) NOT NULL ,  
room\_occupancy NUMBER (3) NOT NULL ,  
session\_date DATE NOT NULL ,  
session\_hour NUMBER (2) NOT NULL ,  
session\_minute NUMBER (2) NOT NULL ,  
week\_day NUMBER (1) NOT NULL ,  
arch\_rent\_films\_ID\_rent NUMBER (10) NOT NULL ,  
arch\_films\_ID\_film NUMBER (10) NOT NULL) ;  
ALTER TABLE archive ADD CONSTRAINT archive\_PK PRIMARY KEY ( ID\_recording ) ;

ALTER TABLE archive ADD CONSTRAINT archive\_arch\_films\_FK FOREIGN KEY ( arch\_films\_ID\_film ) REFERENCES arch\_films ( ID\_film ) ;  
ALTER TABLE archive ADD CONSTRAINT archive\_arch\_rent\_films\_FK FOREIGN KEY ( arch\_rent\_films\_ID\_rent ) REFERENCES arch\_rent\_films ( ID\_rent ) ;