

**Федеральное государственное автономное образовательное учреждение  
высшего профессионального образования  
«Национальный исследовательский ядерный университет «МИФИ»**

**Факультет «Информационная безопасность»**

Отчёт по

Лабораторной работе №1-1:

«Построение модели данных»

Выполнил: Чыонг Тхи Ан Хай

Группа: Б17-565

Преподаватель:  Куприяшин М.А.

Москва 2020

1. **Цель работы**

Приобретение навыков построения модели данных на основе неформального описания предметной области и реализации этой структуры данных средствами RDBMS Oracle.

1. **Описание предметной области**

**База данных сериалов.** Золотая классика лабораторных работ по базам данных. Компания предоставляет посетителям веб-сайта информацию о телевизионных фильмах: название, год выхода, бюджет, аннотацию, списки актёров и эпизодов, рецензии критиков, и т. д.

1. **Нормализация**
   1. 1NF

Переменная отношения находится в первой нормальной форме (1NF) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

Рис.1

|  |
| --- |
| Information of movies |
| **Movie\_id (PK)**  Movie\_name  Movie\_year  Movie\_premise  **Review\_id** **(PK)**  Review  **Episode\_id** **(PK)**  Episode\_name  **Actor\_id (PK)**  Actor\_name  Actor\_age |

* 1. 2NF

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа.

Рассмотриваем рис.1:

- Actor\_name, actor\_age только зависят от ключа Actor\_id,а не зависят от Movie\_id, Reviews\_id или Episode\_id.

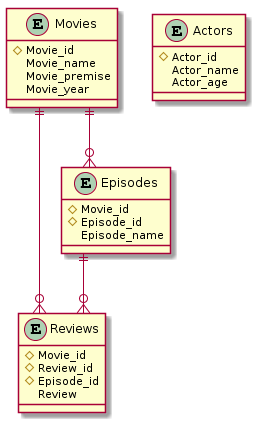
- Movie\_name, movie\_year,… не зависят от Actor\_id, Review\_id, Episode\_id

- Review, Episode\_name не зависят от Actor\_id

…

* Рис.1 не соответствует 2NF

Рис.2



- Каждый Movie может иметь 0 или несколько Reviews

- Каждый Movie может иметь 0 или несколько Episodes (если это сериал)

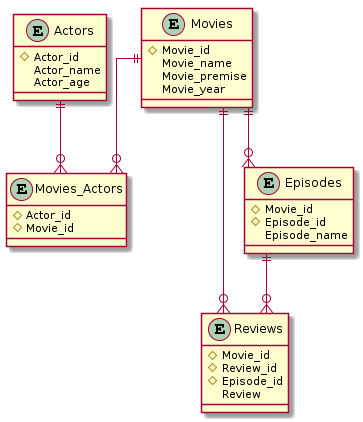
- Каждый Episodes (Reviews) может принадлежать одному и только одному Movie.

- Каждый Episodes может иметь 0 или несколько Reviews

- Каждый Reviews может принадлежать одному и только одному Episodes.

Нужно добавить таблицу Movies\_actors для соединения Movies и Аctors

Рис.3



- Каждый Actor может принадлежать несколькому Movies или не принадлежать ни одному Movie

- Каждый Movie может несколько Actors или не имеет информацию об этом.

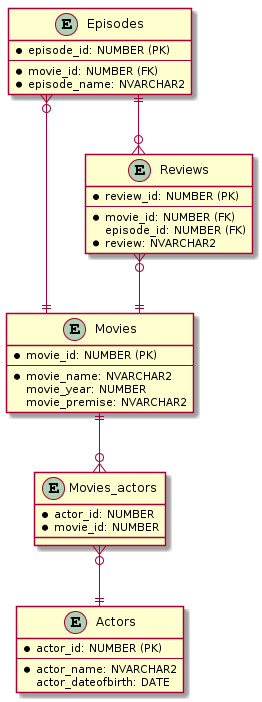
* 1. 3NF

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

Рис.3 уже находится в 3NF

1. **Диаграмму отношений сущность**

Рис.4

****

1. **Приложение**

SQL> ALTER SESSION SET "\_ORACLE\_SCRIPT"=true;

Session altered.

SQL> CREATE USER student IDENTIFIED BY 2211;

User created.

SQL> GRANT CREATE SESSION, CREATE TABLE to student;

Grant succeeded.

SQL> GRANT UNLIMITED TABLESPACE TO student;

Grant succeeded.

SQL> connect student

Connected.

SQL> create table Movies(

2 Movie\_id NUMBER (5,0) NOT NULL,

3 Movie\_name NVARCHAR2 (40) NOT NULL,

4 Movie\_year NUMBER (4,0),

5 Movie\_premise NVARCHAR2 (50),

6 PRIMARY KEY (Movie\_id));

Table created.

SQL> create table Actors(

2 Actor\_id NUMBER (5,0) NOT NULL,

3 Actor\_name NVARCHAR2 (40) NOT NULL,

4 Actor\_dateofbirth DATE,

5 PRIMARY KEY (Actor\_id));

Table created.

SQL> create table Movies\_actors(

2 Movie\_id NUMBER (5,0) NOT NULL,

3 Actor\_id NUMBER (5,0) NOT NULL,

4 foreign key (Movie\_id) references Movies(Movie\_id),

5 foreign key (Actor\_id) references Actors(Actor\_id));

Table created.

SQL> create table Episodes(

2 Episode\_id NUMBER (5,0) NOT NULL,

3 Movie\_id NUMBER (5,0) NOT NULL,

4 Episode\_name NVARCHAR2 (40) NOT NULL,

5 PRIMARY KEY (Episode\_id),

6 foreign key (Movie\_id) references Movies(Movie\_id));

Table created.

SQL> create table Reviews(

2 Review\_id NUMBER (5,0) NOT NULL,

3 Movie\_id NUMBER (5,0) NOT NULL,

4 Episode\_id NUMBER (5,0),

5 Review NVARCHAR2(50));

Table created.

SQL> insert all

2 into Movies(Movie\_id,Movie\_name) values (1,'The good doctor')

3 into Movies(Movie\_id,Movie\_name) values (2,'Charlie and the Chocolate Factory')

4 SELECT \* FROM dual;

2 rows created.

SQL> insert all

2 into Actors(Actor\_id,Actor\_name) values (1,'Freddie Highmore')

3 into Actors(Actor\_id,Actor\_name) values (2,'Johnny Depp')

4 into Actors(Actor\_id,Actor\_name) values (3,'Fiona Gbelman')

5 SELECT \* FROM dual;

3 rows created.

SQL> insert all

2 into Movies\_actors values (1,1)

3 into Movies\_actors values (1,3)

4 into Movies\_actors values (2,1)

5 into Movies\_actors values (2,3)

6 SELECT \* FROM dual;

4 rows created.

SQL> insert all

2 into Episodes values (1,1,'Burnt Food')

3 into Episodes values (2,1,'Mount Rushmore')

4 SELECT \* FROM dual;

2 rows created.

SQL> insert all

2 into Reviews(Review\_id,Movie\_id,Review) values (1,2,'This is the good film')

3 into Reviews values (2,1,1,'This is the best episode in this serie')

4 SELECT \* FROM dual;

2 rows created.

SQL> SELECT \* FROM Movies;

MOVIE\_ID MOVIE\_NAME MOVIE\_YEAR

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

MOVIE\_PREMISE

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

1 The good doctor

2 Charlie and the Chocolate Factory

SQL> SELECT \* FROM Actors ;

ACTOR\_ID ACTOR\_NAME ACTOR\_DAT

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

1 Freddie Highmore

2 Johnny Depp

3 Fiona Gbelman

SQL> SELECT \* FROM Movies\_actors ;

MOVIE\_ID ACTOR\_ID

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

1 1

1 3

2 1

2 3

SQL> SELECT \* FROM Reviews ;

REVIEW\_ID MOVIE\_ID EPISODE\_ID

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

REVIEW

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

1 2

This is the good film

2 1 1

This is the best episode in this serie

SQL> SELECT \* FROM Episodes ;

EPISODE\_ID MOVIE\_ID EPISODE\_NAME

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

1 1 Burnt Food

2 1 Mount Rushmore

SQL> spool off

<https://github.com/TruongThiAnHai/SQL/blob/master/Lab1>

1. **Заключение**

Целью данного исследования было построения модели данных на основе неформального описания предметной области и последующая реализация этой структуры данных средствами RDBMS Oracle.

В данной работе:

- Нормализовали модель данных до мере третьей нормальной формы. Построили диаграмму отношений для разработанной модели данных (рис.4).

- Создали пользователя в подключаемой базе данных Oracle и, используя его учётную запись, реализовали разработанную модель данных.

1. **Список литературы**
2. Entity Relationship Diagram. <https://plantuml.com/ie-diagram>
3. Нормальная форма. <https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0>
4. SQL\*Plus. [https://www.orafaq.com/wiki/SQL\*Plus](https://www.orafaq.com/wiki/SQL*Plus)
5. Нормализация отношений. Шесть нормальных форм. <https://habr.com/ru/post/254773/>