**Lab 2**

Group 1

Lab team: Ievgenii Nudga, Aryan Sultan

Instructor: Florian Ocker

HAW Hamburg Data Base

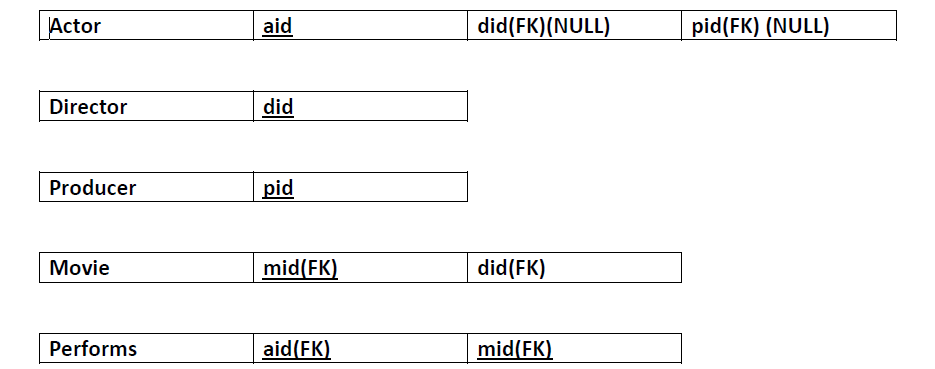
**Assignment 5: Transform the ERM to a Relation Schema**

a) Relations: Actor, Director, Producer, Movie, Performs, LeadRole, alsoProducer, alsoDirector, Produces

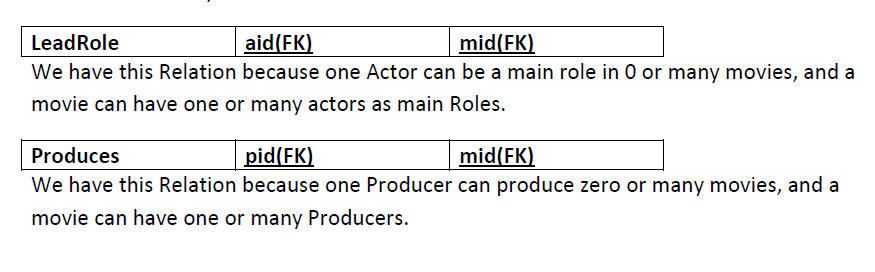
b)

option1: no relation alsoProducer / alsoDirector

This option has less Tables so we have a biggerActor table, possibily with emptyvalues for did and pid, so we need NULL on these columns.

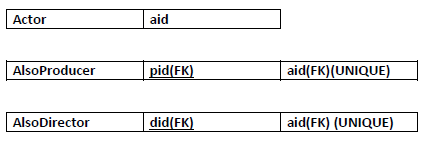


We have this Relation because one Actor can perform in 1 or many movies, and a movie can have 1 or many actors.



option2: with relation alsoProducer / alsoDirector

In this option we have two extra tables so we avoid empty values in the Actor Table. The UNIQUE is necessary so we avoid that two producers were the same actor, or two Actors cannot be the same Producer

****

c) Relational Schema

**Actor ( A\_id , Name, Birthday, Oscar winner)**

**Director ( D\_id , Name, Oscar winner, Company)**

**Producer ( P\_id , Name, Oscar winner, Company)**

**Movie( M\_id ,D\_id(FK),Title, Date)**

**Performs in (A\_id(FK), M\_id(FK))**

**Lead role( A\_id(FK), M\_id(FK) )**

**Directs (M\_id(FK), D\_id(FK))**

**Produces ( M\_id(FK), P\_id(FK) )**

**also Director ( D\_id(FK), A\_id(FK))**

**also producer ( P\_id(FK), A\_id(FK))**

**Assignment 6: Movie Database**

CREATE TABLE actor (aid INTEGER NOT NULL CONSTRAINT actor\_id PRIMARY KEY);

CREATE TABLE director (did INTEGER NOT NULL CONSTRAINT director\_id PRIMARY KEY);

CREATE TABLE producer (pid INTEGER NOT NULL CONSTRAINT producer\_id PRIMARY KEY);

CREATE TABLE movie (mid INTEGER NOT NULL CONSTRAINT movie\_id PRIMARY KEY,

director INTEGER CONSTRAINT fk\_movie\_director REFERENCES director(did) );

CREATE TABLE performs (actor INTEGER CONSTRAINT fk\_actor\_ REFERENCES actor(aid),

performs\_in INTEGER CONSTRAINT fk\_acts\_in REFERENCES movie(mid) );

CREATE TABLE lead\_role (main\_actor INTEGER CONSTRAINT fk\_lead\_actor REFERENCES actor(aid),

lead\_role\_in INTEGER CONSTRAINT fk\_lead\_in REFERENCES movie(mid) );

CREATE TABLE produces (producer INTEGER CONSTRAINT fk\_producer\_id REFERENCES producer(pid),

movie\_produced INTEGER CONSTRAINT fk\_movie\_produced REFERENCES movie(mid) );

CREATE TABLE also\_producer (producer\_id INTEGER CONSTRAINT fk\_producer\_actor REFERENCES producer(pid),

actor\_id INTEGER CONSTRAINT fk\_actor\_producer REFERENCES actor(aid) );

CREATE TABLE also\_director ( director\_id INTEGER NOT NULL CONSTRAINT fk\_director\_actor REFERENCES director(did),

actor\_id INTEGER NOT NULL CONSTRAINT fk\_actor\_director REFERENCES actor(aid) );

DROP TABLE also\_producer;

DROP TABLE also\_director;

DROP TABLE performs;

DROP TABLE lead\_role;

DROP TABLE produces;

DROP TABLE movie;

DROP TABLE actor;

DROP TABLE producer;

DROP TABLE director;

Table ALSO\_PRODUCER dropped.

Table ALSO\_DIRECTOR dropped.

Table PERFORMS dropped.

Table LEAD\_ROLE dropped.

Table PRODUCES dropped.

Table MOVIE dropped.

Table ACTOR dropped.

Table PRODUCER dropped.

Table DIRECTOR dropped.

**Assignment 7: Recursive Relationship**

**A) Relational Schema:**

Person (Name, Birthday, Day\_of\_death, Father(FK), Mother(FK))

In this relational schema there is one primary key (PK), namely Name. And we have two foreign keys (FK), for father and mother respectively as they belong to the same Person entity.

**B) Script and output:**

DROP TABLE Person;

CREATE TABLE Person(

Name VARCHAR2(255) NOT NULL CONSTRAINT pk\_name PRIMARY KEY,

Birthday DATE NOT NULL,

Day\_of\_death DATE NULL,

Father VARCHAR2(255) NULL,

Mother VARCHAR2(255) NULL,

FOREIGN KEY (Father) REFERENCES Person(Name),

FOREIGN KEY (Mother) REFERENCES Person(Name)

);

**Output:**

Table PERSON dropped.

Table PERSON created.

**C) For every column, give a justification why it is nullable or not:**

Name column should be not null, every person has a name.

Birthday column should be not null, every person has a birthday.

Day\_of\_Death column should be nullable, every person is not dead (at least yet).

Father(FK) and Mother(FK) columns should be nullable.

**D) Inserting the data into table and explanation of nuances:**

First we enter the data for Grandparents, then Parents and then Children. The reason is the presence of foreign keys Father and Mother. These foreign keys contain the name of the person already present. So a Parent should be present before we enter Child’s data.

**Script(on next page):**

**Script:**

INSERT INTO Person VALUES('Rich', TO\_DATE('19-04-1935', 'DD-MM-YYYY'), NULL,NULL,NULL);

INSERT INTO Person VALUES('Sue', TO\_DATE('13-05-1940', 'DD-MM-YYYY'), TO\_DATE('28-05-2002', 'DD-MM-YYYY'),

NULL,NULL);

INSERT INTO Person VALUES('Walt', TO\_DATE('10-03-1930', 'DD-MM-YYYY'), TO\_DATE('20-12-2007', 'DD-MM-YYYY'),

NULL,NULL);

INSERT INTO Person VALUES('Ellen', TO\_DATE('09-09-1930', 'DD-MM-YYYY'), NULL,NULL,NULL);

INSERT INTO Person VALUES('Bob', TO\_DATE('11-11-1965', 'DD-MM-YYYY'), NULL,'Walt','Ellen');

INSERT INTO Person VALUES('Susan', TO\_DATE('08-08-1966', 'DD-MM-YYYY'), NULL,'Rich','Sue');

INSERT INTO Person VALUES('Jane', TO\_DATE('01-01-2006', 'DD-MM-YYYY'), NULL,'Bob','Susan');

INSERT INTO Person VALUES('Joe', TO\_DATE('02-02-2007', 'DD-MM-YYYY'), NULL,'Bob','Susan');

COMMIT;

**Script output:**

Table PERSON dropped.

Table PERSON created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Commit complete.

