Opera Company Database Design Project

November 2019

Helen Husca

16325880

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write.

Description of Database

The purpose of an opera company is to put on productions of operas, and information on these productions can be organised in a database.

The designed opera company presents productions, and each production has an ID, title, director, language, premiere date, movement director, set designer, costume designer, surtitle operator and lighting designer. It is assumed that there is only one of each creative staff, that the production is given an ID, and that there are always surtitles, even if the language of the production is English.

A production can be recorded as a DVD, which has an ID, release date and running time. A DVD records exactly one production, and a production may only be recorded as one DVD. It is assumed that each DVD has a unique ID, and that the title of the DVD does not differ from that of the production.

Each production stages exactly one work, and the work is assumed to be given a unique ID by the company for disambiguation, as titles may be shared by different works. A work has a title, language, premiere date, premiere location, composer and librettist. There can be multiple librettists and multiple composers of an opera work. It is assumed that that the language of the original work may differ from the language of the production of the work, e.g. a modern interpretation of an opera may be in English, and that the title of the production may differ from that of the work it is based on.

There are multiple performances of a production, and each performance shows exactly one production. A performance is assumed to be given an ID by the opera company, and also has a date and a start time. The assumption is that there is a main stage were productions are shown.

Each production includes roles, and each role is assumedly given a role ID for ease of distinction, and has a name, short description and voice type associated with it. Roles in a production of a work may differ from the original characters in a work e.g. a production may merge two characters into one, or leave one out entirely. There can't be a production without roles, but there can be roles without a production, e.g. roles may be the same as the original opera work. Roles can also be included in many productions e.g. there can be a many productions of an opera work presented by the company which include the same roles.

Roles are played by solo artists, each of which are given an ID by the company, have a name, birth date, email address, phone number and voice type. Roles can exist without a solo artist playing them e.g. they have not been cast yet. Solo artists can play many roles e.g. an artist may play two supporting roles. Roles can be played by many solo artists e.g. the principle performer of a main role is a solo artist, but the cover performer of the same role is another solo artist.

Entity Relationship Model

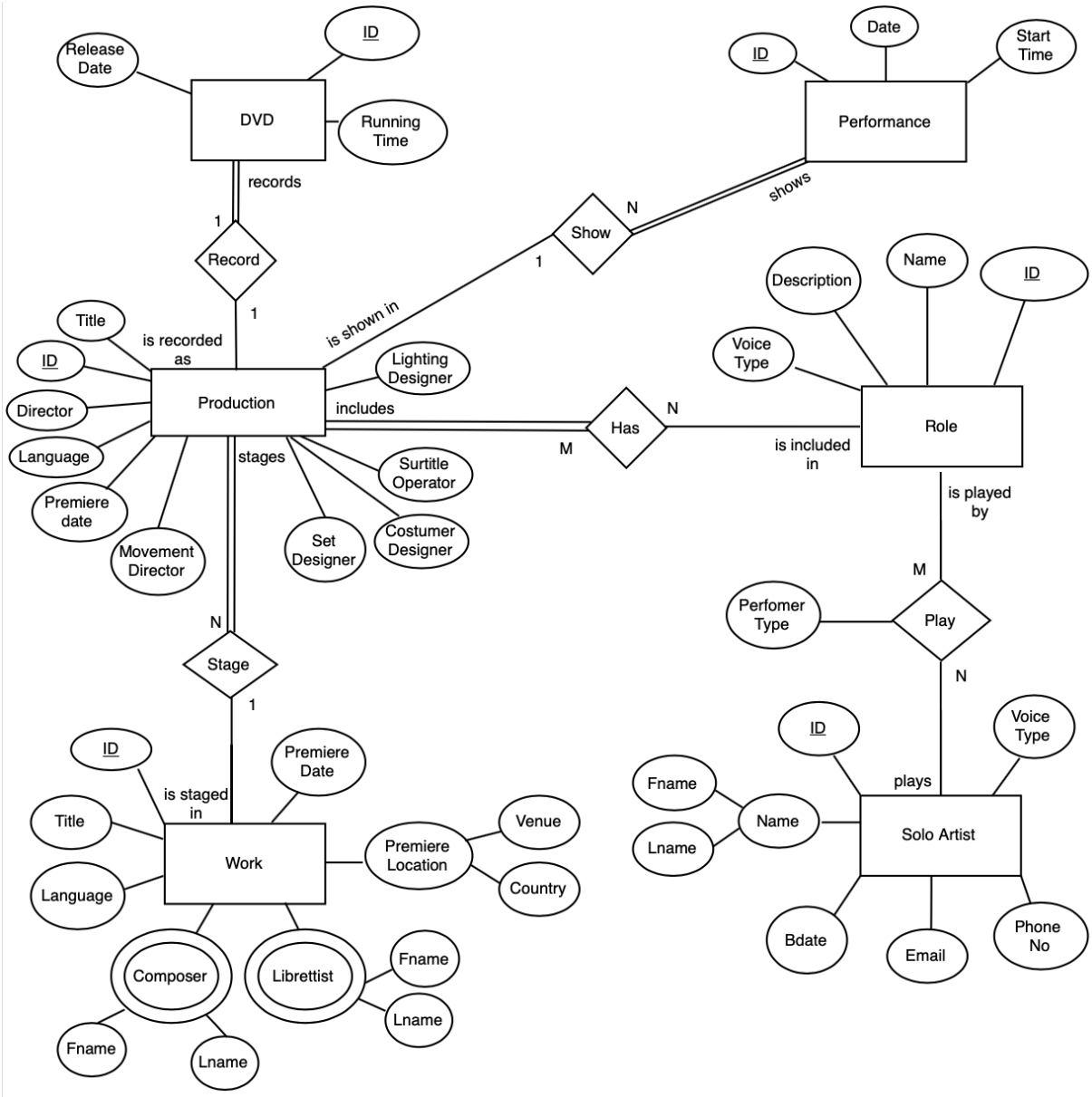

Figure 1: Entity Relationship Diagram. The boxes represent entity types, the diamonds represent relationship types and the ovals represent the attributes. The attributes which are underlined signify the primary keys.

Relational Schema Mapping

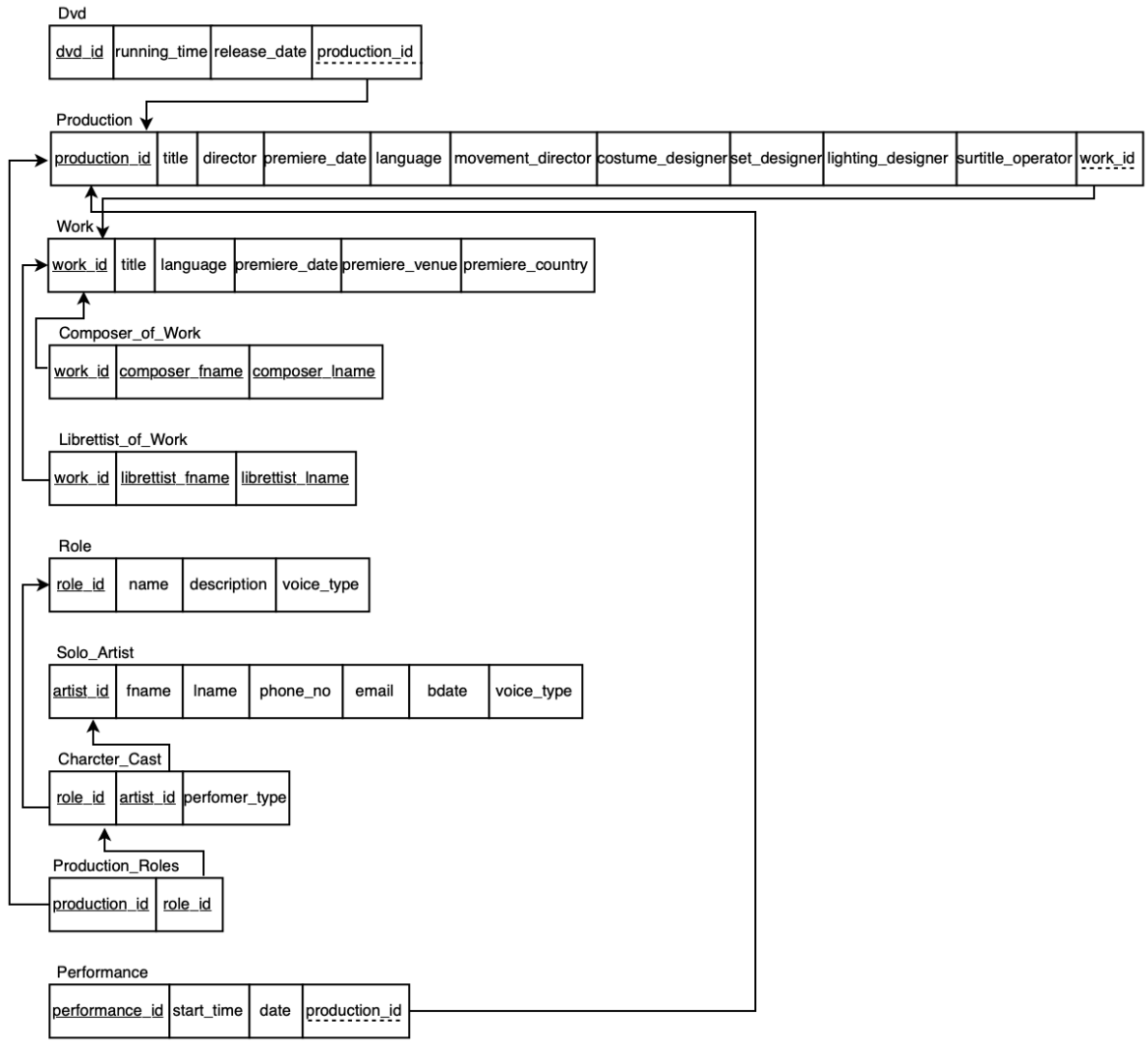

Figure 2: Relational Schema. The underlined attributes denote primary keys, and dotted lines indicate foreign keys. The arrows describe primary key/ foreign key pairs.

Functional Dependencies

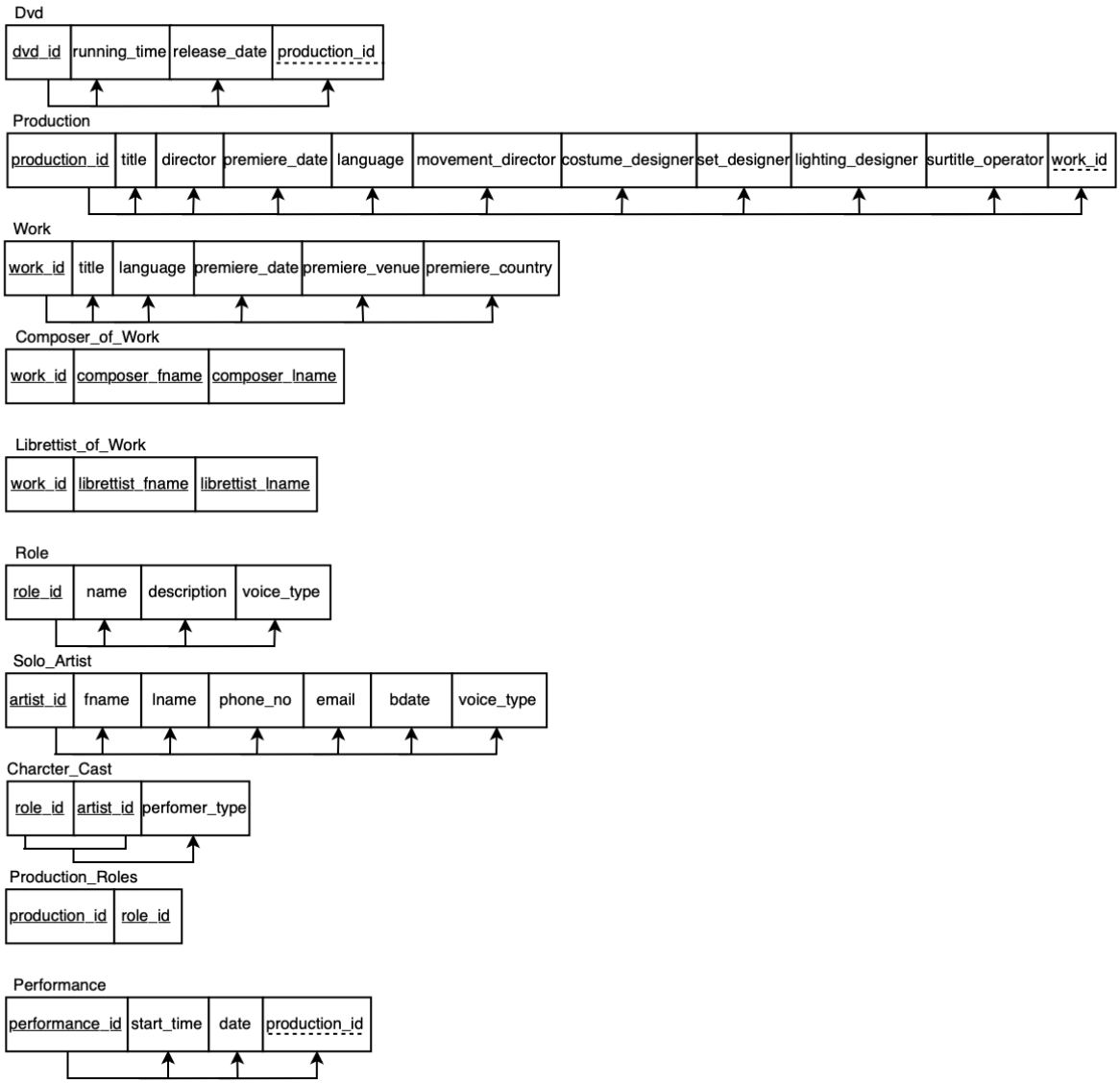

Figure 3: Functional Dependency Diagram. The underlined attributes denote primary keys, and dotted lines indicate foreign keys. The arrows describe dependencies.

Normalisation

The relations have been created so as to avoid anomalies when inserting values into or deleting values from the relations. They achieve the following normal form (NF) levels:

1NF: Every row-column intersection contains an atomic value based on the domain, and there are no lists of values. The relations Composer_of_Work and Librettist_of_Work were generated when the multivalued attributes, shown by the double-lined circle in Figure 1, were each pulled into a new table with the primary key of the entity Work.

2NF: Every non-key attribute in a relation is fully functionally dependent on the entire primary key. Figure 3 illustrates that all but one relation have a single-attribute primary key, making these relations in 2NF, and the relation Character_Cast, which has role_no and artist_id as its composite primary key, shows no partial dependencies as performer_type is dependent on both attributes together.

3NF: There is no transitive dependency of non-key attributes on the primary key. This is clearly displayed in Figure 3 through the lack of arrow starting from non-key attributes.

BCNF: All attributes in the relation are dependent on only the whole primary key. The primary key in each relation functions as the superkey in the relation. This is shown clearly in Figure 3, where each non-key attribute in a relation has a one arrow pointing directly to it from the entire primary key.

The tables are shown to be fully normalised and thus cannot contain redundant data. The insertion and deletion operations benefit from the elimination of redundancy through normalisation.

Database Constraints

The creation of the database covers table constraints which are specified as part of the relation definition.

NOT NULL

Many attributes were specified with a NOT NULL constraint so that NULL would not be a permissible attribute value. The following example shows that the last name of a solo artist must not be NULL, as an artist always has a name.

```
lname VARCHAR(15) NOT NULL
```

PRIMARY KEY

Each relation has a single attribute or composite primary key to uniquely identify a tuple in the relation. Primary keys can never be NULL. The following example illustrated the primary key of the Role relation.

```
role_no INT NOT NULL PRIMARY KEY
```

UNIQUE

Relations can have more than one candidate key. The Solo_Artist relation has phone_no as a secondary key:

```
phone_no INT UNIQUE
```

FOREIGN KEY

Foreign keys of a relation refer to primary keys of other relations. They may be NULL in certain circumstances, but there is no case which warrants a NOT NULL specification for a foreign key in this opera company database. The example below shows Production relation referring to the Work relation through the foreign key work_id.

```
work_id INT NOT NULL,
FOREIGN KEY(work_id) REFERENCES Work(work_id)
```

The types of integrity constraints that are enforced are Key Constraint, Entity Integrity Constraint and Referential Integrity Constraint. Any INSERT or UPDATE operation that violates the Key Constraint through duplication is rejected, since the primary key can not be a duplicate. As an example, the second INSERT operation below is rejected and the primary key, role number, is a duplicate:

```
INSERT INTO Role VALUES (31225, 'Aida', 'Ethiopian princess', 'Soprano');
INSERT INTO Role VALUES (31225, 'Radames', 'Captain of the Guard', 'Tenor');
```

If these operations violate the Entity Integrity Constraint through NULL values, they are rejected since the primary key can not be NULL if it is to unique identify a tuple. The following query is rejected:

```
INSERT INTO Role VALUES (NULL, 'Amneris', 'Daughter of the King, 'Mezzo-soprano');
```

Referential Integrity Constraint is formally specified by a foreign key and can be violated by INSERT, UPDATE or DELETE operations. Like the violation of Key and Entity Integrity Constraints, the default action is to reject the operation. It is possible however to specify an alternative action such as SET NULL, CASCADE and SET DEFAULT if preferable. Since a foreign key can never be NULL in this database, and there aren't default values for foreign keys present, CASCADE remains the only alternative if deemed appropriate. The following example shows that work_id foreign key in the Production relation cascades the update of the work_id primary key in the Work relation:

```
work_id INT NOT NULL,
FOREIGN KEY(work_id) REFERENCES Work(work_id)
ON UPDATE CASCADE
```

Since all instances where work_id is a foreign key have a CASCADE clause attached, the value of the work_id is updated wherever it is referenced by tuples from relations other than Work, so that the old value before the update is no longer stored in the database.

More complex constraints such as CHECK and TRIGGER were also used to limit the values that an attribute can be assigned.

CHECK

A CHECK constraint restricts the permitted values of an attribute to a specified subset of the domain. Therefore, whenever a tuple is inserted or modified, it is tested against this constraint before any changes are made. The example below shows that the voice_type attribute in the Solo_Artist relation must be in the set of specified voice types.

```
voice_type VARCHAR(15) NOT NULL CHECK (voice_type IN ('Soprano','Mezzo-soprano','Contralto','Countertenor','Tenor','Baritone','Bass'))
```

A violation of this domain constraints results in an error being thrown and no further changes to the database.

TRIGGER

A trigger specifies the action to be taken when a condition is satisfied in a constraint checked on some event such as insert. When a tuple is added to the Production relation without a specified title and language, the following trigger assigns them the same values for these attributes as those of the work on which the production is based.

```
DELIMITER //
CREATE TRIGGER Production_Title_Language
BEFORE INSERT ON Production
FOR EACH ROW
BEGIN
IF NEW.title IS NULL
THEN
SET NEW.title = (SELECT Work.title
FROM Work
WHERE NEW.work_id = Work.work_id);
END IF;
IF NEW.language IS NULL
THEN
SET NEW.language = (SELECT Work.language
FROM Work
WHERE NEW.work_id = Work.work_id);
END IF;
END//
DELIMITER ;
```

The following operation illustrates the resulting action of the trigger, where the title ‘Lohegrin’ and the language ‘German’ are attribute values in the tuple without being specified as such in the operation.

```
INSERT INTO Production(production_id,director,premiere_date,work_id) VALUES
(13367, 'Michael Swan','2001-08-12',13762);
```

production_id	title	director	premiere_date	language	movement_director	costume_designer	set_designer	lighting_designer	surtitle_operator	work_id
13367	Lohengrin	Michael Swan	2001-08-12	German	NULL	NULL	NULL	NULL	NULL	13762

Another trigger was included to deal with inserts into the DVD relation where its release date occurs before the premiere date of the production. Since a recording of a DVD can not be released before the production is staged for the first time, it is set to NULL in the DVD relation.

```
DELIMITER //
CREATE TRIGGER Dvd_Date
BEFORE INSERT ON Dvd
FOR EACH ROW
BEGIN
IF NEW.release_date < (SELECT premiere_date
FROM Production
WHERE NEW.production_id = Production.production_id)
THEN
SET NEW.release_date = NULL;
END IF;
END//
DELIMITER ;
```

The operation below results in the release-date of the the DVD being set to NULL, since the premiere date of the production (2020-01-18) is larger than the release_date of the opera ('2020-01-01'). The release date as specified in the operation is not inserted with the rest of the attributes of DVD but instead is given a NULL value.

```
INSERT INTO Dvd VALUES(14292,223,'2020-01-01',19752);
```

dvd_id	running_time	release_date	production_id
14292	223	NULL	19752

Further Examples

Views were also defined in the database from other defining tables.
One such view showed the titles of all the works that a composer created, in ascending order of composer's last name and in secondary ordering of the composition's title. This provides easier viewing and querying of information that needs to be retrieved about the composer's compositions, as opposed to the frequently need to define the join of the tables in question, here Composer_of_Work and Work.

```
CREATE VIEW Composer_Composition(lname, fname, work_title) AS
SELECT composer_lname, composer_fname, title
FROM Composer_of_Work, Work
WHERE Composer_of_Work.work_id = Work.work_id
ORDER BY composer_lname, title;
```

Another view was defined to give information on the titles of the productions presented each year, as well as their number of performances. This provides a short summary the production and performances that the opera company organise.

```
CREATE VIEW Yearly_Productions (year, title, perfomance_amount) AS
SELECT Year(premiere_date), title, COUNT(performance_id)
FROM Production, Performance
WHERE Performance.production_id = Production.production_id
GROUP BY Year(premiere_date), title
ORDER BY Year(premiere_date) DESC, title;
```

PL/SQL variables can be used in the opera company database to provide information on the names of soprano and tenor singers who are stored in the database and their emails. This could prove helpful when the organisers are trying to find suitable cover performers for a role in a production from solo artists who are already stored in the database.

```
DELIMITER //
DECLARE
  voice_sop Solo_Artist.voice_type%type := 'Soprano';
  voice_ten Solo_Artist.voice_type%type := 'Tenor';
  a_fname  Solo_Artist.fname%type;
  a_lname Solo_Artist.lname%type;
  a_voice  Solo_Artist.voice_type%type;
  a_email Solo_Artist.email%type;

BEGIN
  SELECT fname, lname, voice_type, email INTO a_fname, a_lname, a_voice, a_email
  FROM Solo_Artist
  WHERE voice_type IN (voice_sop,voice_ten);
  dbms_output.put_line
  ('Name: ' || a_fname || ' ' || a_lname || ' with voice type: ' || a_voice  || ' has
email address: ' || a_email);
END;
DELIMITER ;
```

Appendix

```
CREATE TABLE Work(
        work_id INT NOT NULL PRIMARY KEY,
        title VARCHAR(150) NOT NULL,
        language VARCHAR(20) DEFAULT 'Italian',
        premiere_date DATE,
        premiere_venue VARCHAR(150),
        permiere_country VARCHAR(150));

CREATE TABLE Production(
        production_id INT NOT NULL PRIMARY KEY,
        title VARCHAR(150),
        director VARCHAR(50) NOT NULL,
        premiere_date DATE,
        language VARCHAR(20),
        movement_director VARCHAR(50),
        costume_designer VARCHAR(50),
        set_designer VARCHAR(50),
        lighting_designer VARCHAR(50),
        surtitle_operator VARCHAR(50),
        work_id INT NOT NULL,
        FOREIGN KEY(work_id) REFERENCES Work(work_id)
        ON UPDATE CASCADE);

CREATE TABLE Dvd(
        dvd_id INT NOT NULL PRIMARY KEY,
        running_time INT DEFAULT 150,
        release_date DATE,
        production_id INT NOT NULL,
        FOREIGN KEY(production_id) REFERENCES Production(production_id)
        ON UPDATE CASCADE);

CREATE TABLE Composer_of_Work(
        work_id INT NOT NULL,
        composer_fname VARCHAR(20) NOT NULL,
        composer_lname VARCHAR(20) NOT NULL,
        PRIMARY KEY(work_id,composer_fname,composer_lname),
        FOREIGN KEY(work_id) REFERENCES Work(work_id)
        ON UPDATE CASCADE);

CREATE TABLE Librettist_of_Work(
        work_id INT NOT NULL,
        librettist_fname VARCHAR(20) NOT NULL,
        librettist_lname VARCHAR(20) NOT NULL,
        PRIMARY KEY(work_id,librettist_fname,librettist_lname),
```

```
        FOREIGN KEY(work_id) REFERENCES Work(work_id)
        ON UPDATE CASCADE);

CREATE TABLE Role(
        role_id INT NOT NULL PRIMARY KEY,
        name VARCHAR(30) NOT NULL,
        description VARCHAR(150),
        voice_type VARCHAR(15) NOT NULL CHECK (voice_type IN ('Soprano','Mezzo-
        soprano','Contralto',
        'Countertenor','Tenor','Baritone','Bass')));

CREATE TABLE Solo_Artist(
        artist_id INT NOT NULL PRIMARY KEY,
        fname VARCHAR(20) NOT NULL,
        lname VARCHAR(20) NOT NULL,
        phone_no VARCHAR(20) UNIQUE,
        email VARCHAR(150) UNIQUE NOT NULL CHECK (email LIKE '%@%.%'),
        bdate DATE,
        voice_type VARCHAR(15) NOT NULL CHECK (voice_type IN ('Soprano','Mezzo-
        soprano','Contralto',
        'Countertenor','Tenor','Baritone','Bass')));

CREATE TABLE Character_Cast(
        role_id INT NOT NULL,
        artist_id INT NOT NULL,
        performer_type VARCHAR(15) CHECK (performer_type IN ('Principle','Cover')),
        PRIMARY KEY(role_id, artist_id),
        FOREIGN KEY(role_id) REFERENCES Role(role_id)
        ON UPDATE CASCADE,
        FOREIGN KEY(artist_id) REFERENCES Solo_Artist(artist_id)
        ON UPDATE CASCADE);

CREATE TABLE Production_Roles(
        production_id INT NOT NULL,
        role_id INT NOT NULL,
        PRIMARY KEY(production_id, role_id),
        FOREIGN KEY(production_id) REFERENCES Production(production_id)
        ON UPDATE CASCADE,
        FOREIGN KEY(role_id) REFERENCES Role(role_id)
        ON UPDATE CASCADE);

CREATE TABLE Performance(
        performance_id INT NOT NULL PRIMARY KEY,
        start_time VARCHAR(5) CHECK (start_time LIKE '__:__'),
        date DATE,
        production_id INT NOT NULL,
        FOREIGN KEY(production_id) REFERENCES Production(production_id)
```

```
        ON UPDATE CASCADE);

INSERT INTO Work VALUES(15353,'Aida','Italian','1871-12-24','Khedivial Opera
House','Egypt');
INSERT INTO Work VALUES(15134,'The Elder Tree','English','2016-09-14','Cork Repertory
Theater','Ireland');
INSERT INTO Work VALUES(15327,'Cosi fan tutte','Italian','1790-01-
26','Burgtheater','Austria');
INSERT INTO Work VALUES(13441,'Madama Butterfly','Italian','1904-02-17','La Scala','Italy');
INSERT INTO Work VALUES(16830,'Don Giovanni','Italian','1787-10-29','Estates
Theatre','Czech Republic');
INSERT INTO Work VALUES(13762,'Lohengrin','German','1850-08-28','Deutsches
Nationaltheater Weimar','Germany');

INSERT INTO Production VALUES(14245,'Aida','Christopher Moore','2013-05-
27','Italian','Tobias Schmidt','Anna Popescu','Svetlana Kuzmina','Ali Garcias','Conor
Gleeson',15353);
INSERT INTO Production VALUES(16332,'The Elder Tree','Maura Flynn','2019-06-
14','English','Maria Holt','Ellen Whittle','Svetlana Kuzmina','Celine Carolan','Franz
Herbert',15134);
INSERT INTO Production VALUES(12149,'Women are like that','Niall Hemming','2008-11-
25','Enlgish','Katrin Hertha','Anton Petrov','Martin Corbin','Ella Burke','Conor
Gleeson',15327);
INSERT INTO Production VALUES(11634,'Madama Butterfly','Christopher Moore', '2008-02-
14','Italian','Gordon Jacob','Andre Derain','Jonathan Watkins','Tim Reid', 'Shane
Austin',13441);
INSERT INTO Production VALUES(19304,'Don Giovanni','Maura Flynn','2017-05-
09','Italian','Maria Holt','Anna Popescu','Ciaran Bradley','Angela Anca','Judith Flynn',16830);
INSERT INTO Production VALUES(19752,'Lohengrin','Martin Corbin','2020-01-
18','German','Amy Santiago','Rosa Coffey','Svetlana Kuzmina','Maximilian Beel','Jane
Hart',13762);

INSERT INTO Dvd VALUES(18475,150,'2013-06-15',14245);
INSERT INTO Dvd VALUES(15218,90,'2019-07-10',16332);
INSERT INTO Dvd VALUES(13889,180,'2008-12-20',12149);
INSERT INTO Dvd VALUES(16322,145,'2008-03-05',11634);
INSERT INTO Dvd VALUES(14298,202,'2017-05-22',19304);

INSERT INTO Composer_of_Work VALUES(15353,'Giuseppe','Verdi');
INSERT INTO Composer_of_Work VALUES(15134,'Catherine','Finn');
INSERT INTO Composer_of_Work VALUES(15134,'Andrew','Synett');
INSERT INTO Composer_of_Work VALUES(15327,'Wolfgang A.','Mozart');
INSERT INTO Composer_of_Work VALUES(13441,'Giacomo','Puccini');
INSERT INTO Composer_of_Work VALUES(16830,'Wolfgang A.','Mozart');
INSERT INTO Composer_of_Work VALUES(13762,'Richard','Wagner');

INSERT INTO Librettist_of_Work VALUES(15353,'Antonio','Ghislanzoni');
```

```
INSERT INTO Librettist_of_Work VALUES(15134,'Nina','Moran');
INSERT INTO Librettist_of_Work VALUES(15327,'Lorenzo','Da Ponte');
INSERT INTO Librettist_of_Work VALUES(13441,'Luigi','Illica');
INSERT INTO Librettist_of_Work VALUES(13441,'Giuseppe','Giacosa');
INSERT INTO Librettist_of_Work VALUES(16830,'Lorenzo','Da Ponte');
INSERT INTO Librettist_of_Work VALUES(13762,'Richard','Wagner');

INSERT INTO Role VALUES(31225, 'Aida', 'Ethiopian princess', 'Soprano');
INSERT INTO Role VALUES(31228, 'Niamh','Housekeeper','Contralto');
INSERT INTO Role VALUES(25258, 'Ferrando','Soldier','Tenor');
INSERT INTO Role VALUES(11760,'Cio-Cio-san','Former geisha','Soprano');
INSERT INTO Role VALUES(20482,'Don Giovanni','Young nobleman','Baritone');
INSERT INTO Role VALUES(15287,'Ortrud','Telramunds wife','Mezzo-soprano');

INSERT INTO Solo_Artist
VALUES(16211,'Martina','Snow','00353879356731','martinasnow1987@gmail.com','1987-
05-20','Soprano');
INSERT INTO Solo_Artist
VALUES(12867,'Suzanne','Pucl','003281942774','puclsuzannemarie90@yahoo.com','1990-
01-29','Contralto');
INSERT INTO Solo_Artist
VALUES(11280,'Marin','Antonescu','00401864267','antonescumarin1@gmail.com','1981-07-
13','Tenor');
INSERT INTO Solo_Artist
VALUES(28664,'Cecilia','Domenico','003519532562','domenicocecilia82@gmail.com','1982-
10-04','Soprano');
INSERT INTO Solo_Artist
VALUES(31896,'Samuel','Theda','00469264768','samuel75theda@yahoo.com','1975-09-
21','Baritone');
INSERT INTO Solo_Artist
VALUES(10769,'Caitlin','Nolan','003538654270','nolancaitlin7@gmail.com','1987-08-
01','Mezzo-soprano');

INSERT INTO Character_Cast VALUES(31225,16211,'Principle');
INSERT INTO Character_Cast VALUES(31228,12867,'Principle');
INSERT INTO Character_Cast VALUES(25258,11280,'Cover');
INSERT INTO Character_Cast VALUES(11760,28664,'Principle');
INSERT INTO Character_Cast VALUES(20482,31896,'Principle');
INSERT INTO Character_Cast VALUES(15287,10769,'Cover');
INSERT INTO Character_Cast VALUES(11760,16211,'Cover');

INSERT INTO Production_Roles VALUES(14245,31225);
INSERT INTO Production_Roles VALUES(16332,31228);
INSERT INTO Production_Roles VALUES(12149,25258);
INSERT INTO Production_Roles VALUES(11634,11760);
INSERT INTO Production_Roles VALUES(19304,20482);
INSERT INTO Production_Roles VALUES(19752,15287);
```

```
INSERT INTO Performance VALUES(10987,'18:30','2013-05-27',14245);
INSERT INTO Performance VALUES(10988,'18:30','2013-05-29',14245);
INSERT INTO Performance VALUES(19752,'19:00','2019-06-14',16332);
INSERT INTO Performance VALUES(27152,'18:00','2008-11-25',12149);
INSERT INTO Performance VALUES(19363,'17:30','2008-02-14',11634);
INSERT INTO Performance VALUES(19364,'18:00','2008-02-16',11634);
INSERT INTO Performance VALUES(12133,'18:30','2017-05-09',19304);
INSERT INTO Performance VALUES(12135,'19:00','2017-05-18',19304);

-- View which shows the titles of all the works that a composer created,
-- in ascending order of composer's last name

CREATE VIEW Composer_Composition(lname, fname, work_title) AS
SELECT composer_lname, composer_fname, title
FROM Composer_of_Work, Work
WHERE Composer_of_Work.work_id = Work.work_id
ORDER BY composer_lname, title;

-- View which shows for each year, the titles of the productions presented in this year
-- as well as their number of performances

CREATE VIEW Yearly_Productions (year, title, perfomance_amount) AS
SELECT Year(premiere_date), title, COUNT(performance_id)
FROM Production, Performance
WHERE Performance.production_id = Production.production_id
GROUP BY Year(premiere_date), title
ORDER BY Year(premiere_date) DESC, title;

-- TRIGGER when inserting new Production with no title or language

DELIMITER //
CREATE TRIGGER Production_Title_Language
BEFORE INSERT ON Production
FOR EACH ROW
BEGIN
IF NEW.title IS NULL
THEN
SET NEW.title = (SELECT Work.title
FROM Work
WHERE NEW.work_id = Work.work_id);
END IF;
IF NEW.language IS NULL
THEN
SET NEW.language = (SELECT Work.language
FROM Work
WHERE NEW.work_id = Work.work_id);
```

```
END IF;
END//
DELIMITER ;

-- Verify that Trigger works

INSERT INTO Production(production_id,director,premiere_date,work_id) VALUES (13367,
'Michael Swan','2001-08-12',13762);

-- TRIGGERS when inserting new dvd and its release date is before the
-- premiere date of the production

DELIMITER //
CREATE TRIGGER Dvd_Date
BEFORE INSERT ON Dvd
FOR EACH ROW
BEGIN
IF NEW.release_date < (SELECT premiere_date
FROM Production
WHERE NEW.production_id = Production.production_id)
THEN
SET NEW.release_date = NULL;
END IF;
END//
DELIMITER ;

-- Verify that trigger works

INSERT INTO Dvd VALUES(14292,223,'2020-01-01',19752);
```