CS27020 Assignment: Moma madness

Lukasz Wrzolek
Department of Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, UK

This paper presents a novel approach to Modelling Databases. In this assignment I have been given a subset of this dataset in un-normalised form. My task was to understand the structure in this data and bring it up to 3NF, implement these 3NF relations in PostgreSQL, deliver an entity-relationship model, and execute some queries on this data. These activities I will described in this report which summarises and justifies the design and decisions I have made.

I. Introduction

ONE of the cool datasets that has been made freely available is from the New York based gallery MOMA. This Museum of Modern Art's collection dataset https://github.com/MuseumofModernArt/collection contains 126,101 records detailing some of the artwork held at New York's most famous art gallery. However, because we are dealing with artists rather than databasists, the dataset is in un-normalised form. Madness! [1]

II. Database Design

This section describe design the database where I had to do normalisation, identify primary keys and bring database to third normal form. Although, I had to find functional dependencies for each attribute are identified to proceed the next stages of firs, second and third normal form.

A. *Primary keys and functional dependencies*

I have been given a sql file from university with data which contain a complete table moma in un-normalized form.

I have listed the attributes of the moma table here: **moma** (title, artist, artist_bio, year, medium, dimensions, credit_line, moma_number, classification, department, date_acquired, curator_approved, object_id, url)

description of the attributes:

- title - The title of the work of art
- artist - the name and surname of the artist
- artist_bio - The artist information about date of birth, place and date of death
- year - The production year of the art
- medium - the medium of the art
- dimensions - dimension of the art
- credit_line - the company which gave that art for the museum
- moma_number - it is the number of moma in database
- classification - it shows where the art has been located and in which department
- department - departments in moma
- date_acquired - the date when museum got that art
- curator_approved - Y or N if the art is approved by curator or no
- object_id - an unique id for the art
- URL - url address of that art ended by object_id

The functional dependencies are desreibed below

1) *Functional dependencies*

In the above un-normalized form of the database I have found a few dependencies of the following attributes:

$artist \rightarrow artist\_bio$
$artist\_bio \rightarrow artist$
Artist and artist_bio is 1..1 dependency so that dependency goes in both ways.

$object\_id \rightarrow moma\_number$
$moma\_number \rightarrow object\_id$
This dependency is the same as artist and artist_bio. Their relation is 1..1 so it will work in both ways

$object\_id \rightarrow url$
If URL exist at the end of the address we have an object_id which separate different works of art

$object\_id \rightarrow (title, artist, artist\_bio, year, medium, dimensions, credit\_line, moma\_number, classification, department, date\_acquired, curator\_approved, url)$
Object_id determines pretty much everything, because as a primary key it has a unique value, so even if values are the same it doesnt matter because that value has relation (1..*)

$moma\_number \rightarrow (title, artist, artist\_bio, year, medium, dimensions, credit\_line, classification, department, date\_acquired, curator\_approved, object\_id, url)$ With moma_number we have the same situation like in object_id. The specific falues which determines multiple values i.e. title and artist are alright in functional dependencies

[1] Introduction is modified version on Introduction from Assignment Description

$(object\_id, moma\_number) \rightarrow (title, artist, artist\_bio, year, medium, dimensions, credit_line, classification, department, date\_acquired, curator\_approved, url)$

2) Primary Keys

Working thtough the functional dependencies I spotted that object_id and moma_number are enough to find information about the work of art so I chase one of it to be a primary key of moma for un normalised form

III. From UNF to 3NF: an account of the normalisation process

Once the primary keys are identified and funcional dependencies are descreibed I was able to start normalisation the structure.

A. 1st Normal Form

First step of the normalisation process is identyfing the unique values. The candidate kyes are moma_number and object_id. Although, moma_number is a candidate key it has records like 2605.2008.1-13, so taht record can be splitted for 13 different record and every of that record will be unique. The beginning of it will be 2605.2008. followed by number from 1 to 13. I have done the same for moma_number by splitting it's tuples as I introduced before. After that moma_number wasn't unique. It has two the same records:

1) "ETCHINGS FROM ECCLESIASTES" with moma number 30.1966.1-18
2) "ECCLESIASTES III:1 (plate, folio 10) from ETCHINGS FROM EC-CLESIASTES" with moma number 30.1966.4

To avoid duplication in moma_number I have created a query which finding a duplicated values and adding next number after "-" mark. In that process I created new relation called dimensions (dimension_id, moma_number, dimensions, object_id). I left touple object_id in that relation because I want to know which dimension belong to the object.

The next part of bringing structure to first normal form is removing the repeating units found in the un-normalized form along with any attributes that are functionally dependent on them. Doing so I splitted moma_number, artist_bio for a different, atomic values. I have created new attributes caalled (artist, nationality, birth_date, death_date), and splitted artist_bio as a new four attributes.

Next non atomic atribute is medium. I had multiple values so I had to create next relation media. After that I moved splitted media to new relation with obiect_id and media_id has been created which is a primary key of media relation. Next thing was creation the relation called "materials" and copying obiect_id from "moma" relation. I set it up for a primary key in that table. Some of media values in "moma" relation has null values so not every value of obiect_id will be in "media" relation. Thats why I chase to fill obiect_id in "materials" relation from "moma" relation, to collect every value of that record. When I finished it I added touple media_id to "materials" and filled it up from "media" relation, leaving <null> where obiect_id didn't exist in "media" relation. When the separate id's from "media" table was collected with their object_id's I dropped that touple from "media" relation.
relation media (media_id, medium)
relation materials(object_id,media_id*,dimension_id*)
relation dimensions(dimension_id, moma_number, dimensions, object_id)

B. 2nd Normal Form

When I brought my database to the first normal form, so my tuples has atomic values I created table artists $(\underline{artist}, nationality, birth\_date, death\_date)$. In relation "moma" artist value is not unique so, when I was transfering artists to their table I had to do it by using distinct keyword and artist touple became a primary key itself.

Next relation created was classification (moma_number, classification, department, date_acquired, curator_approved, credit_line).

C. 3rd Normal Form

Next step was to create a new relation classifications $(\underline{classification}, department)$ to store classifications and their departments. There was a transitive dependency $title \rightarrow department$, because if I know the title I have a moma_number for it, following so I have classification and then I have department.

IV. Implementation in postgres

In this section I will describe step by step my proces of designing the database and my queries which I used to get form of the database.

A. Artist process design

First thing than needs to be done is set up the primary key for moma teble at un-normalised form. It will be object_id as I wrote before.

```
- -set up primary key for moma
ALTER TABLE moma ADD PRIMARY KEY (object_id);
```

Next thing is creating new tuples in "moma" relation which gonna hold our splitted values.

```
- -create columns for new values
ALTER TABLE luw19.moma
ADD COLUMN nationality TEXT,
ADD COLUMN birth_place TEXT,
ADD COLUMN birth_date INTEGER,
ADD COLUMN death_date INTEGER;
```

Now I was ready to write a query to split values and update moma relation

```
- - perform update moma table
WITH splitted_data AS (
SELECT title,
detail[1] AS nationality,
detail[4] AS birth_place,
detail[5] AS birth_date,
detail[6] AS death_date,
object_id
FROM (SELECT luw19.moma.*,
regexp_matches(luw19.moma.artist_bio,'\(([∧),]+),?\s?(born)?(([∧\.]+)\.?)?(\d{4})?(\d{4})?\)') AS detail
FROM luw19.moma
) t
)
UPDATE luw19.moma
SET
nationality = val.nationality,
birth_place = val.birth_place,
birth_date = val.birth_date :: INTEGER,
death_date = val.death_date :: INTEGER
FROM (
SELECT
nationality,
birth_place,
birth_date,
death_date,
object_id
FROM splitted_data
) AS val
WHERE luw19.moma.object_id = val.object_id;

ALTER TABLE moma DROP artist_bio;
```

[2]

[2] I am writing using latex so symbol ∧ is really weird displayed and it can generate problems if somebody copy that code and use it in future. Make sure that you check it before!

TABLE I
LUW19.MOMA

column	type	modifiers
title	character varying(800)	
artist	character varying(100)	
year	integer	
medium	character varying(1500)	
dimensions	character varying(2500)	
credit_line	character varying(700)	
moma_number	character varying(20)	
classification	character varying(100)	
department	character varying(100)	
date_acquired	date	
curator_approved	character varying(1)	
object_id	integer	not null
url	character varying(200)	
nationality	text	
birth_place	text	
birth_date	integer	
dearth_date	integer	

Then moma relation look like this:
cs27020_15_16=> \ d moma I [3]

B. Media

Then I created media relation, and splitted medium by using delimeters and set first letter to capital to avoid duplicating records.

```
- -creating table which will hold a media
CREATE TABLE materials (object_id INTEGER, PRIMARY KEY (object_id));
INSERT INTO materials(object_id)
SELECT object_id FROM moma;
ALTER TABLE materials ADD COLUMN media_id INTEGER;
- -creating and splitting media table
CREATE TABLE media (object_id INTEGER,medium VARCHAR(1500));
- -inserting values
INSERT INTO media(object_id,medium)
SELECT object_id,
regexp_split_to_table("medium", ’(([,;] (?!printed))|[,;]? and )’) "medium"
FROM luw19.moma;
- -adding a primary key for that table
ALTER TABLE media ADD COLUMN media_id SERIAL PRIMARY KEY;
- -update materials from media
UPDATE materials
SET media_id=media.media_id
FROM media WHERE materials.object_id=media.object_id;
- -dropping object_id
ALTER TABLE media DROP object_id;
```

That query creates two relations: Materials II and media III

cs27020_15_16=> \ d materials II

Indexes: "materials_pkey" PRIMARY KEY, btree (object_id) Foreign-key constraints: "materials_dimensions_dimension_id_fk" FOREIGN KEY (dimension_id) REFERENCES dimensions(dimension_id) "materials_media_media_id_fk" FOREIGN KEY (media_id) REFERENCES media(media_id) Referenced by: TABLE "moma" CONSTRAINT "moma_materials_object_id_fk" FOREIGN KEY (object_id) REFERENCES materials(object_id)

[3] I had to create the table from the output because, output wasn’t clear. copied versions can be found here: VIII

TABLE II
LUW19.MATERIALS

Column	Type	Modifiers
obiect_id	integer	not null
media_id	integer	
dimension_id	text	

TABLE III
LUW19.MEDIA

Column	Type	Modifiers
medium	character varying(1500)	
media_id	integer	not null default nextval('media_media_id_seq'::regclass)

cs27020_15_16=> \ d media III

Indexes: "media_pkey" PRIMARY KEY, btree (media_id) Referenced by: TABLE "materials" CONSTRAINT "materials_media_media_id_fk" FOREIGN KEY (media_id) REFERENCES media(media_id)

C. Dimensions

As I mentioned before dimensions which had "-" are splittend and putted to different relation.

```
- - dimensions table with splitted dimensions as dimension_id
CREATE TABLE dimensions
AS - -splitting dimensions
WITH splitted_dimension AS (
SELECT luw19.moma.moma_number
, regexp_replace(moma.moma_number,' ([0 − 9\.] + \.)([0 − 9]+) − ([0 − 9]+)', e'\\1', e'g') AS dimension_id
, regexp_replace(moma.moma_number,' ([0 − 9\.] + \.)([0 − 9]+) − ([0 − 9]+)', e'\\2', e'g') AS dimension
, regexp_replace(moma.moma_number,' ([0 − 9\.] + \.)([0 − 9]+) − ([0 − 9]+)', e'\\3', e'g') AS object
, moma.dimensions
, moma.object_id
FROM luw19.moma
)
SELECT a1.moma_number
, a1.dimension_id || generate_series( a1.dimension::integer, a1.object::integer)::TEXT AS dimension_id
, a1.dimensions, a1.object_id
FROM splitted_dimension a1
WHERE a1.dimension <> a1.dimension_id
UNION ALL
SELECT a0.moma_number
, a0.dimension_id AS dimension_id
, a0.dimensions, a0.object_id
FROM splitted_dimension a0
WHERE a0.dimension = a0.dimension_id;
- -selecting non-unique values and giving them a selected index
with stats as (
SELECT "dimension_id",
"object_id",
row_number() over (partition by "dimension_id" order by object_id) as rn
FROM dimensions
)
UPDATE dimensions
SET "dimension_id" = CASE WHEN s.rn > 1 THEN s."dimension_id"|| '-'|| s.rn
ELSE s."dimension_id"
END
FROM stats s
WHERE S."dimension_id" = dimensions."dimension_id"
AND S."object_id" = dimensions."object_id";
```

TABLE IV
LUW19.DIMENSIONS

Column	Type	Modifiers
moma_number	character varying(20)	
dimension_id	text	not null
dimensions	character varying(2500)	
obiejct_id	integer	

TABLE V
LUW19.ARTISTS

Column	Type	Modifiers
artist	character varying(100)	not null
nationality	character varying(100)	
birth_place	character varying(100)	
birth_date	integer	
death_date	integer	

```
- -adding dimension_id column to materials
ALTER TABLE materials ADD COLUMN dimension_id TEXT; - -update column to hold dimension_id
UPDATE materials
SET dimension_id=dimensions.dimension_id
FROM dimensions WHERE dimensions.object_id=materials.object_id;
```

```
cs27020_15_16=> \ d dimensions IV
```

```
Indexes: "dimensions_pkey" PRIMARY KEY, btree (dimension_id) Referenced by: TABLE "materials"
CONSTRAINT "materials_dimensions_dimension_id_fk" FOREIGN KEY (dimension_id) REFERENCES
dimensions(dimension_id)
```

D. Artist Table

Simple code to move artists and their details to the relation "artists"

```
- -creating artist table
CREATE TABLE artists (artist VARCHAR(100) PRIMARY KEY , nationality VARCHAR(100), birth_place
VARCHAR(100), birth_date INTEGER, death_date INTEGER);
INSERT INTO artists (artist, nationality, birth_place, birth_date, death_date)
SELECT DISTINCT artist,
nationality,
birth_place,
birth_date,
death_date
FROM luw19.moma;
- -dropping old values
ALTER TABLE luw19.moma
DROP artist,
DROP nationality,
DROP birth_date,
DROP birth_place,
DROP death_date;
```

```
cs27020_15_16=> \ d artists V
```

```
Indexes: "artists_pkey" PRIMARY KEY, btree (artist) Referenced by: TABLE "moma" CONSTRAINT
"moma_artists_artist_fk" FOREIGN KEY (artist) REFERENCES artists(artist)
```

TABLE VI
luw19.classifications

Column	Type	Modifiers
classification	character varying(100)	not null
department	character varying(100)	

TABLE VII
luw19.moma

Column	Type	Modifiers
title	character varying(800)	
artist	character varying(100)	
year	integer	
medium	character varying(1500)	
credit_line	character varying(700)	
classification	character varying(100)	
date_acquired	date	
curatior_approved	character varying(1)	
object_id	integer	not null
url	character varying(200)	

E. Classifications

And The last relation classifications to hold distinct classification separatelly.

```
- - creating table classifications for the distinct classifiacation
CREATE TABLE classifications (classification VARCHAR(100) PRIMARY KEY, department VARCHAR(100));
INSERT INTO classifications (classification, department)
SELECT DISTINCT classification,department
FROM classification;
```

cs27020_15_16=> \ d classifications VI

Indexes: "classifications_pkey" PRIMARY KEY, btree (classification) Referenced by: TABLE "classification" CONSTRAINT "classification_classifications_classification_fk" FOREIGN KEY (classification) REFERENCES classifications(classification)

F. Moma in 3rd NF

Moma relation after all operations
cs27020_15_16=> \ d moma VII

Indexes: "moma_pkey" PRIMARY KEY, btree (object_id) Foreign-key constraints: "moma_artists_artist_fk" FOREIGN KEY (artist) REFERENCES artists(artist) "moma_classifications_classification_fk" FOREIGN KEY (classification) REFERENCES classifications(classification) "moma_materials_object_id_fk" FOREIGN KEY (object_id) REFERENCES materials(object_id)

V. SQL Queries

A. Yoko Ono

```
- - query to find a title and department for Yoko Ono.
WITH yoko AS (
SELECT title,object_id
FROM moma
WHERE artist = 'Yoko Ono'
)
SELECT
moma.title, classifications.department
FROM
yoko
INNER JOIN moma on moma.object_id=yoko.object_id
INNER JOIN classifications ON moma.classification = classifications.classification;
```

and produced output is 10 rows table VIII :

TABLE VIII
Yoko Ono

title	department
Mend Piece for John from S.M.S. No. 5	Prints &amp; Illustrated Books
No. 4 (Bottoms)	Film
Instructions for Paintings	Prints &amp; Illustrated Books
Piece for Nam June Paik no. 1	Prints &amp; Illustrated Books
Grapefruit	Prints &amp; Illustrated Books
Typescript for Do It Yourself Fluxfest Presents Yoko Ono &amp; Dance Co.	Prints &amp; Illustrated Books
Catalogue for Yoko Ono's exhibition This is Not Here at the Everson Museum, Syracuse, New York	Prints &amp; Illustrated Books
Grapefruit	Prints &amp; Illustrated Books
Part Painting/Series 5/to Tony Cox	Prints &amp; Illustrated Books
Do It Yourself Fluxfest Presents Yoko Ono &amp; Dance Co.	Prints &amp; Illustrated Books

B. List of the artists

I have tried to do b. part of 4th point but, I don't why the query is not working correctly.

```
SELECT artist, count(*) AS works, array_agg(distinct year order by year asc)
FROM (SELECT
media_id,
department
FROM (SELECT media_id, classification FROM materials
LEFT JOIN moma ON materials.object_id = moma.object_id)
AS media_with_classifications
LEFT JOIN classifications ON media_with_classifications.classification = classifications.classification)t
JOIN artists ON moma.artist = artists.artist
JOIN moma ON moma.object_id=materials.object_id
GROUP BY artist ORDER BY works DESC LIMIT 10;
```

VI. Entity Relationship Diagram

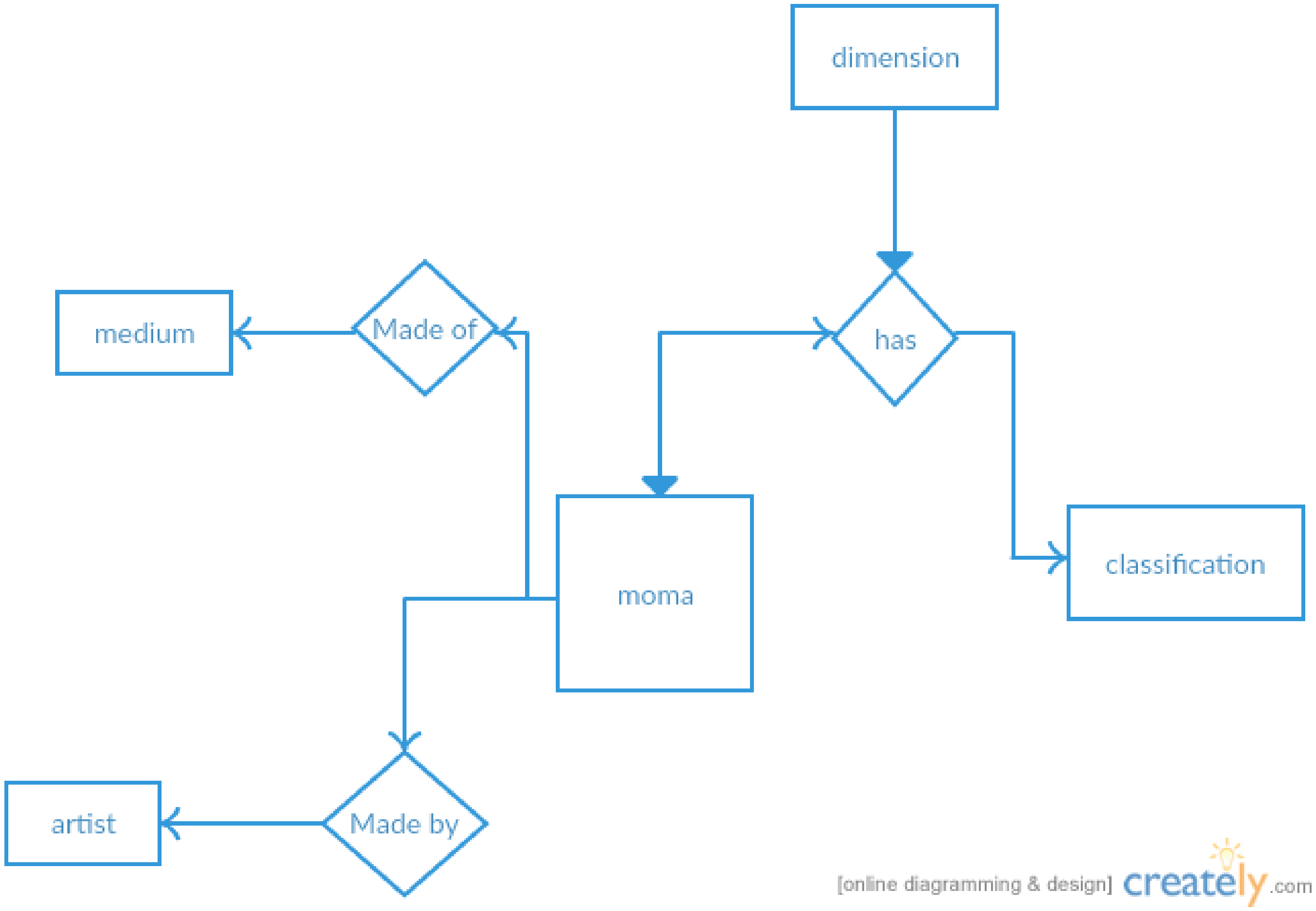
In this section I will show a diagrams of relationship. One of them is produced by DataGrip IDE 1(a), second is created by me using an online diagram creator VI.

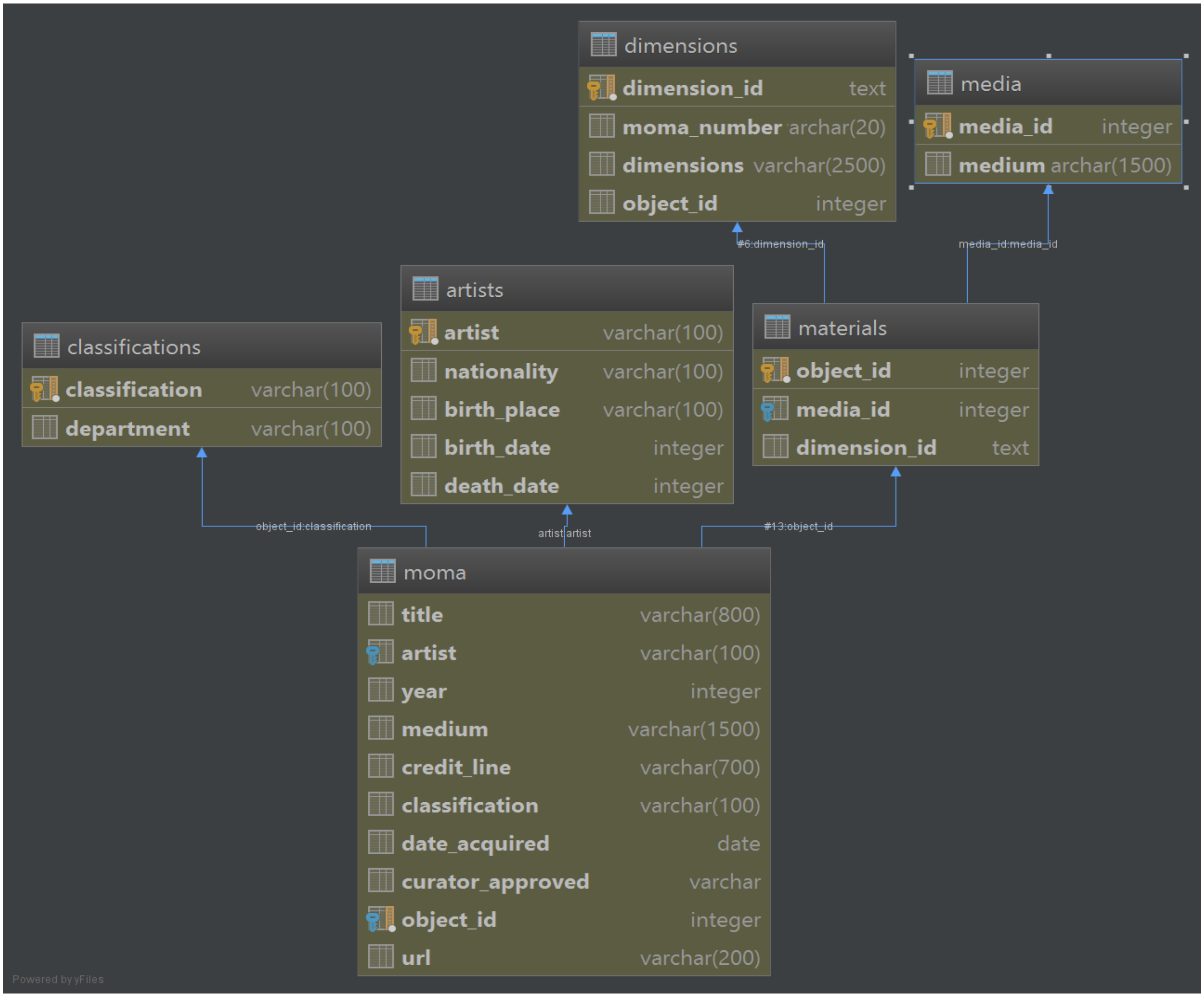
VII. Exploring MOMA further

I have downloaded "extra.sql" from the page and run my script on it. The queries works good and created bigger database set than before.

I have observed one interesting thing, namely is that I can work on the part of dataset from database, create a queries to design and sort out smaller database and after implement the same queries for the "parent" database with larger amount of information held. Although, on bigger dataset it takes much more time, so now I understand why is important do work on smaller sets of data

[Entity Relationship Diagram]





dimensions	
dimension_id	text
moma_number	archar(20)
dimensions	varchar(2500)
object_id	integer

media	
media_id	integer
medium	archar(1500)

artists	
artist	varchar(100)
nationality	varchar(100)
birth_place	varchar(100)
birth_date	integer
death_date	integer

materials	
object_id	integer
media_id	integer
dimension_id	text

classifications	
classification	varchar(100)
department	varchar(100)

moma	
title	varchar(800)
artist	varchar(100)
year	integer
medium	varchar(1500)
credit_line	varchar(700)
classification	varchar(100)
date_acquired	date
curator_approved	varchar
object_id	integer
url	varchar(200)

(a) Diagram produced by DataGrip

VIII. Copied versions from cmd

cs27020_15_16=> \ d moma

Table "luw19.moma"

Column | Type | Modifiers ——————+————————-+———— title | character varying(800) | artist | character varying(100) | year | integer | medium | character varying(1500) | dimensions | character varying(2500) | credit_line | character varying(700) | moma_number | character varying(20) | classification | character varying(100) | department | character varying(100) | date_acquired | date | curator_approved | character varying(1) | object_id | integer | not null url | character varying(200) | nationality | text | birth_place | text | birth_date | integer | death_date | integer |

Indexes: "moma_pkey" PRIMARY KEY, btree (object_id)

cs27020_15_16=> \ d materials

Table "luw19.materials"

Column | Type | Modifiers —————+———+———— object_id | integer | not null media_id | integer | dimension_id | text |

Indexes: "materials_pkey" PRIMARY KEY, btree (object_id) Foreign-key constraints: "materials_dimensions_dimension_id_fk" FOREIGN KEY (dimension_id) REFERENCES dimensions(dimension_id) "materials_media_media_id_fk" FOREIGN KEY (media_id) REFERENCES media(media_id) Referenced by: TABLE "moma" CONSTRAINT "moma_materials_object_id_fk" FOREIGN KEY (object_id) REFERENCES materials(object_id)

cs27020_15_16=> \ d media

Table "luw19.media" Column | Type | Modifiers ———-+————————-+———————————————————- medium | character varying(1500) | media_id | integer | not null default nextval('media_media_id_seq'::regclass)

Indexes: "media_pkey" PRIMARY KEY, btree (media_id) Referenced by: TABLE "materials" CONSTRAINT "materials_media_media_id_fk" FOREIGN KEY (media_id) REFERENCES media(media_id)

cs27020_15_16=> \ d dimensions

Table "luw19.dimensions" Column | Type | Modifiers —————+————————-+———— moma_number | character varying(20) | dimension_id | text | not null dimensions | character varying(2500) | object_id | integer |

Indexes: "dimensions_pkey" PRIMARY KEY, btree (dimension_id) Referenced by: TABLE "materials" CONSTRAINT "materials_dimensions_dimension_id_fk" FOREIGN KEY (dimension_id) REFERENCES dimensions(dimension_id)

cs27020_15_16=> \ d artists

Table "luw19.artists"

Column | Type | Modifiers ————-+————————+———— artist | character varying(100) | not null nationality | character varying(100) | birth_place | character varying(100) | birth_date | integer | death_date | integer |

Indexes: "artists_pkey" PRIMARY KEY, btree (artist) Referenced by: TABLE "moma" CONSTRAINT "moma_artists_artist_fk" FOREIGN KEY (artist) REFERENCES artists(artist)

cs27020_15_16=> \ d classifications

Table "luw19.classifications"

Column | Type | Modifiers —————-+————————+———— classification | character varying(100) | not null department | character varying(100) |

Indexes: "classifications_pkey" PRIMARY KEY, btree (classification) Referenced by: TABLE "classification" CONSTRAINT "classification_classifications_classification_fk" FOREIGN KEY (classification) REFERENCES classifications(classification)

cs27020_15_16=> \ d moma

Table "luw19.moma"

Column | Type | Modifiers ——————+————————-+———— title | character varying(800) | artist | character varying(100) | year | integer | medium | character varying(1500) | credit_line | character varying(700) | classification | character varying(100) | date_acquired | date | curator_approved | character varying(1) | object_id | integer | not null url | character varying(200) |

Indexes: "moma_pkey" PRIMARY KEY, btree (object_id) Foreign-key constraints: "moma_artists_artist_fk" FOREIGN KEY (artist) REFERENCES artists(artist) "moma_classifications_classification_fk" FOREIGN KEY (classification) REFERENCES classifications(classification) "moma_materials_object_id_fk" FOREIGN KEY (object_id) REFERENCES materials(object_id)