Arsh Singh

Data Book PART I: The Narrative
Music Industry Database:

Description:

In this age of digital media and big data, there is always a need of informative databases for companies. In the music world, companies need organized data to ensure they're operating efficiently. My database brings and relates all relevant pieces of information together to fulfill the needs of companies working in the music industry. It could bring value to streaming services, music licensing platforms, artist managers and others. Let's get to how I structured my database and what it entails.

First, singer is an integral part of this system because it relates to many other entities. A singer can sing many songs which then could be part of albums. So, this leads us to have three entities of singer, song and albums. Furthermore, we have music composers and songwriters. Both of them are related to singers in a many to many relationship. For clarity, a singer can work with many composers and songwriters. So far, we've covered five entities with composers and songwriters joining the list. Next, we have record labels who oversee and work with both singers and composers. By design, relationship between them is limited to one-to-one which means one singer and one composer can only be signed to one label at a time. Then we have studio which is unique to composers in which they work. Information is available about studio's location, who it belongs to, etc. so that singers and labels can make informed decision about working with them. After this, we have an entity which stores hit songs that also tracks their albums and release dates. Followed by that, remaining two entities that we need to cover are singer award and concert. Awards can only be won by singers. It has attributes that gives information about which year the award was won, who won it and for what. Lastly, concert table consists of concert's ID, singer's ID and other attributes relating to its venue, date and attendance.

The last section covered the general structure of the database though all relationships between entities were not touched in depth. Business rules section will go through each relationship in detail. Next, potential use cases of our database will be discussed.

The need of music data is high in the music industry, be it streaming services, music licensing platforms, analysts and more. Music streaming services can utilize this database to upload all the necessary metadata associated with a piece of content. This

can help improve the experience of users as they get information about their favorite artists. Streaming services can show upcoming or past concert data plus playlists of hit tracks to inform users of the current trends in the music industry. Besides streaming services, licensing platforms can use the database to find suitable content for their own needs. For instance, film industry and advertising companies need music for their content, thus they can get relevant information about songs, albums, singers, etc. to streamline their licensing process.

Here is the summary of all ten entities for reference: SINGER, MUSIC_COMPOSER, SONG, ALBUM, RECORD_LABEL, STUDIO, HIT_TRACK_LIST, SINGER_AWARD, CONCERT, SONGWRITER.

Arsh Singh

Data Dictionary:

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK OR FK	FK REFERENCED TABLE
SINGER	SINGER_ID	Singer ID	INT	99999	10000-99999	Y	PK	
	SINGER_LNAME	Singer's last name	VARCHAR(15)	xxxxxxx		Y		
	SINGER_FNAME	Singer's first name	VARCHAR(15)	xxxxxxx		Y		
	LABEL_ID	Label ID	INT	99999			FK	RECORD_LABEL
	SINGER_DOB	Singer's Date of Birth	DATE	yyyy-mm-dd				
SING_COMP_COLLAB	SING_COMP_ID	Singer and composer's collaboration ID	INT	99999			PK	
	SINGER_ID	Singer ID	INT	99999			FK	SINGER
	COMPOSER_ID	Composer ID	INT	99999			FK	MUSIC_COMPOSER
	SING_COMP_COLLAB_DATE	Collaboration date	DATE	yyyy-mm-dd				
	SONG_ID	Song ID	INT	99999			FK	
MUSIC_COMPOSER	COMPOSER_ID	Music composer's ID	INT	99999		Y	PK	
	COMPOSER_LNAME	Music composer's last name	VARCHAR(15)	xxxxxxx		Y		
	COMPOSER_FNAME	Music composer's first name	VARCHAR(15)	xxxxxxx		Y		
	LABEL_ID	Label ID	INT	99999			FK	RECORD_LABEL

	COMPOSER_DOB	Composer's Date of Birth	DATE	yyyy-mm-dd				
SONG	SONG_ID	Song ID	INT	99999		Y	PK	
	SONG_REL_DATE	Song release date	DATE	yyyy-mm-dd				
	SINGER_ID	Singer ID	INT	99999	10000-99999	Y	FK	SINGER
	ALBUM_ID	Album ID	INT	99999			FK	ALBUM
	WRITER_ID	Songwriter ID	INT	99999			FK	SONGWRITER
	SONG_NAME	Song's title	VARCHAR(20)	xxxxx				
ALBUM	ALBUM_ID	Album ID	INT	99999		Y	PK	
	ALBUM_NAME	Album name	VARCHAR(20)	xxxxxxx		Y		
	ALBUM_REL_DATE	Album release date	DATE	yyyy-mm-dd				
	SINGER_ID	Singer ID	INT	99999	10000-99999	Y	FK	SINGER
	COMPOSER_ID	Composer ID	INT	99999			FK	MUSIC_COMPOSER
RECORD_LABEL	LABEL_ID	INT	CHAR(5)	99999		Y	PK	
	LABEL_NAME	Record label name	VARCHAR(15)	xxxxxxx		Y		
	LABEL_OWNER_FNAME	Label owner's first name	VARCHAR(15)	xxxxxxx		Y		
	LABEL_OWNER_LNAME	Label owner's last name	VARCHAR(15)	xxxxxxx		Y		
	LABEL_ESTABLISHED	Label established date	DATE	yyyy-mm-dd				
STUDIO	STUDIO_ID	Studio ID	INT	99999		Y	PK	

	STUDIO_NAME	Studio name	VARCHAR(15)	xxxxxxx		Y		
	COMPOSER_ID	Composer ID	INT	99999		Y	FK	MUSIC_COMPOSER
	STUDIO_LOCATION	Studio Location	VARCHAR(15)	xxxxxxx				
	STUDIO_ESTABLISHED	Studio established date	DATE	yyyy-mm-dd				
HIT_TRACK_LIST	HIT_TRACK_ID	Hit Track's ID	INT	99999		Y	PK	
	TRACK_CHART_POS	Song's peak chart position	VARCHAR(3)	xxxxxxx		Y		
	SONG_ID	SONG ID	INT	99999	10000-99999	Y	FK	SONG
	TRACK_GENRE	Song's music genre	VARCHAR(15)	mm-dd-yyyy				
	TRACK_CERTIFICATION	Certification of the song	VARCHAR(20)	99999				
SINGER_AWARD	AWARD_ID	Award ID	INT	99999		Y	PK	
	AWARD_NAME	Award name	VARCHAR(45)	xxxxxxx				
	SINGER_ID	Singer ID	INT	99999	10000-99999		FK	SINGER
	AWARD_GIVEN_DATE	Award Given Date	DATE	yyyy-mm-dd				
	AWARD_ORG	Organization that provides the award.	VARCHAR(30)	xxxxxxx				
CONCERT	CONCERT_ID	Concert ID	INT	99999		Y	PK	
	SINGER_ID	Singer ID	INT	99999	10000-99999	Y	FK	SINGER
	CONC_VENUE	Concert Venue (city, country)	VARCHAR(40)	xxxxxxx		Y		
	CONC_DATE	Concert date	DATE	yyyy-mm-dd				

	CONC_ATTENDANCE	Concert Attendance	BIGINT	999999				
SONGWRITER	WRITER_ID	Songwriter ID	INT	99999		Y	PK	
	WRITER_FNAME	Songwriter first name	VARCHAR(15)	xxxxxxx		Y		
	WRITER_LNAME	Songwriter last name	VARCHAR(15)	xxxxxxx		Y		
	WRITER_DOB	Songwriter Date of Birth	DATE	yyyy-mm-dd				
	WRITER_DEBUT	Songwriter debut year	YEAR	yyyy				
SING_WRITER_COLLAB	SING_WRITE_ID	Singer and writer collab ID	INT				PK	
	SINGER_ID	Singer ID	INT				FK	
	WRITER_ID	Writer ID	INT				FK	
	SING_WRITE_COLLAB_DATE	Singer and Writer Collaboration date	DATE	yyyy-mm-dd				
	SONG_NAME	Song Name	VARCHAR(20)					

Arsh Singh

ERM Components Table:

ENTITY	RELATIONSHIP	CONNECTIVITY	ENTITY
SINGER	sings	1:M	SONG
SINGER	works with	M:N	MUSIC_COMPOSER
SINGER	works	M:N	SONGWRITER
SINGER	has	1:M	ALBUM
SINGER	can win	1:M	SINGER_AWARD
SINGER	can perform at	1:M	CONCERT
MUSIC_COMPOSER	works in	1:1	STUDIO
MUSIC_COMPOSER	can have	1:M	ALBUM
ALBUM	can have	1:M	SONG
RECORD_LABEL	can sign	1:M	SINGER
RECORD_LABEL	can sign	1:M	MUSIC_COMPOSER
HIT_TRACK_LIST	can have	1:1	SONG
SONGWRITER	writes	1:M	SONG
Note: Two composite entities (bridge tables) were created to establish the M:N relationship between SINGER and MUSIC_COMPOSER ,and SNGER and SONGWRITER. These entities are in data dictionary: SING_COMP_COLLAB and SING_WRITER_COLLAB			

Business Rules:

SINGER and SONG

- Each SINGER can sing many SONG.

- Each SONG can only be sung by one SINGER.

SINGER and MUSIC_COMPOSER
- Each SINGER can work with many MUSIC_COMPOSER.
- Each MUSIC_COMPOSER can work with many singers.

SINGER and SONGWRITER
- Each SINGER can work with many SONGWRITER.
- EACH SONGWRITER can work with many SINGERS.

SINGER and ALBUM
- Each SINGER can have many ALBUM.
- Each ALBUM can only have one SINGER.

SINGER and SINGER_AWARD
- Each SINGER can win many SINGER_AWARD
- Each SINGER_AWARD can be won by only one SINGER.

SINGER and CONCERT
- Each SINGER can perform at many CONCERT.
- Each CONCERT can only be performed by one SINGER.

MUSIC_COMPOSER and STUDIO
- Each MUSIC_COMPOSER works in only one STUDIO.
- Each STUDIO is only utilized by one MUSIC_COMPOSER.

MUSIC_COMPOSER and ALBUM
- Each MUSIC_COMPOSER can have many ALBUM.
- Each ALBUM can only be composed by one MUSIC_COMPOSER.

ALBUM and SONG
- Each ALBUM can have many SONG.

- Each SONG can only be part of one ALBUM

RECORD_LABEL and SINGER
- Each RECORD_LABEL can sign many SINGER.
- Each SINGER can only be signed to one RECORD_LABEL.

RECORD_LABEL and MUSIC_COMPOSER
- Each RECORD_LABEL can sign many MUSIC_COMPOSER.
- Each MUSIC_COMPOSER can only be signed to one RECORD_LABEL.

HIT_TRACK_LIST and SONG
- Each HIT_TRACK_LIST instance can have only one SONG.
- Each SONG instance can only appear once or never in HIT_TRACK_LIST.

SONGWRITER and SONG
- Each SONGWRITER can write many SONG.
- Each SONG can only be written by one SONGWRITER.

SING_COMP_COLLAB and SONG
- Each SING_COMP_COLLAB instance can only have one SONG.
- Each SONG instance can only appear once in SING_COMP_COLLAB.

DATA BOOK PART II

Relational Schema for my database:

SINGER (**SINGER_ID,** SINGER_LNAME, SINGER_FNAME, LABEL_ID, SINGER_DOB)

SING_COMP_COLLAB (**SING_COMP_ID**, SINGER_ID, COMPOSER_ID, SING_COMP_COLLAB_DATE, SONG_ID)

MUSIC_COMPOSER(**COMPOSER_ID**, COMPOSER_LNAME, COMPOSER_FNAME, LABEL_ID, COMPOSER_DOB)

SONG(**SONG_ID**, SONG_REL_DATE, SINGER_ID, ALBUM_ID, WRITER_ID, SONG_NAME)

ALBUM(**ALBUM_ID**, ALBUM_NAME, ALBUM_REL_DATE, SINGER_ID, COMPOSER_ID)

RECORD_LABEL(**LABEL_ID**, LABEL_NAME, LABEL_OWNER_FNAME, LABEL_OWNER_LNAME, LABEL_ESTABLISHED)

STUDIO(**STUDIO_ID**, STUDIO_NAME, COMPOSER_ID, STUDIO_LOCATION, STUDIO_ESTABLISHED)

HIT_TRACK_LIST(**HIT_TRACK_ID**, TRACK_CHART_POS, SINGER_ID, TRACK_GENRE, TRACK_CERTIFICATION)

SINGER_AWARD(**AWARD_ID**, AWARD_NAME, SINGER_ID, AWARD_GIVEN_DATE, AWARD_ORG)

CONCERT(**CONCERT_ID**, SINGER_ID, CONC_VENUE, CONC_DATE, CONC_ATTENDANCE)

SONGWRITER(**WRITER_ID**, WRITER_FNAME, WRITER_LNAME, WRITER_AGE, WRITER_DEBUT)

SING_WRITER_COLLAB(**SING_WRITE_ID**, SINGER_ID, WRITER_ID, SONG_WRITE_COLLAB_DATE. SONG_NAME)

Normalization Process:

All of our current entities in the database are already in the desired 3NF. However, I'll denormalize the database to some extent for demonstration or illustration purposes.

How *pre-normalized* database structure would look like:

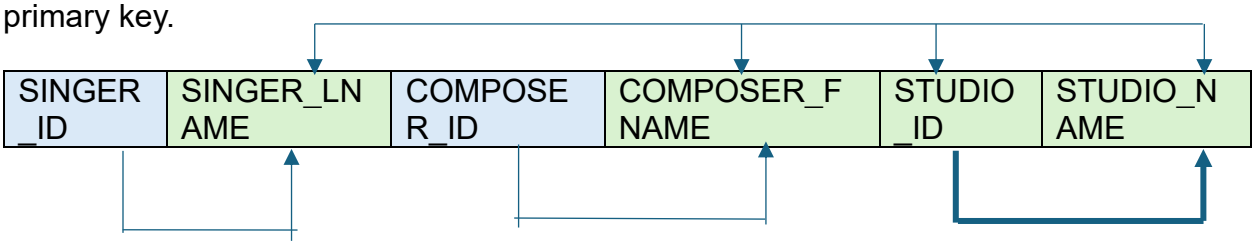
SINGER_ID	SINGER_LNAME	COMPOSER_ID	COMPOSER_FNAME	STUDIO_ID	STUDIO_NAME
91234	Sidhu	81234	Karan	71123	Karan's Studio
		81235	Jordan	71124	Real Music Room
		81249	Mxrci	71150	Creation Place
91235	Raf	81270	Deep	71190	Music House
		81235	Jordan	71124	Real Music Room
		81265	Kidd	71195	Kidd's Studio
91236	GG	81234	Karan	71123	Karan's Studio
		81249	Mxrci	71150	Creation Place

The table above has some issues that could be refined. First, a primary key is undefined as of now, and there are some empty repeating groups present. So, let's set this to 1NF.

SINGER_ID	SINGER_LNAME	COMPOSER_ID	COMPOSER_FNAME	STUDIO_ID	STUDIO_NAME
91234	Sidhu	81234	Karan	71123	Karan's Studio
91234	Sidhu	81235	Jordan	71124	Real Music Room
91234	Sidhu	81249	Mxrci	71150	Creation Place

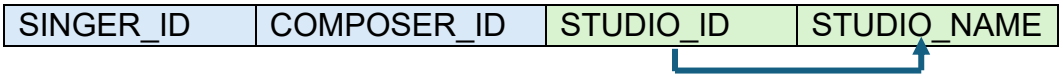
91235	Raf	81270	Deep	71190	Music House
91235	Raf	81235	Jordan	71124	Real Music Room
91235	Raf	81265	Kidd	71195	Kidd's Studio
91236	GG	81234	Karan	71123	Karan's Studio
91236	GG	81249	Mxrci	71150	Creation Place

We're currently at 1NF. The blue shaded attributes are prime attributes and serve as the primary key.

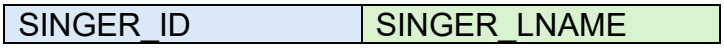


1NF(**SINGER_ID, COMPOSER_ID,** SINGER_LNAME, COMPOSER_FNAME, STUDIO_ID, STUDIO_NAME)

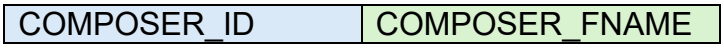
To move to 2NF, we need to get rid of partial dependencies which are when a non-prime attribute is dependent on only a part of primary key not entirety of it (thin blue lines underneath attributes).
We need to create three tables to do that.



(**SINGER_ID, COMPOSER_ID,** STUDIO_ID, STUDIO_NAME)



(**SINGER_ID.** SINGER_LNAME)



(**COMPOSER_ID,** COMPOSER_FNAME)

Now, we've achieved 2NF. All we've left to do is to deal with the transitive dependency in bold arrows to achieve 3NF. We need to split up STUDIO_ID.

STUDIO_ID	STUDIO_NAME

STUDIO

(**STUDIO_ID,** STUDIO_NAME)

SINGER_ID	COMPOSER_ID

SING_COMP_COLLAB

(**SINGER_ID, COMPOSER_ID**)

SINGER_ID	SINGER_LNAME

SINGER

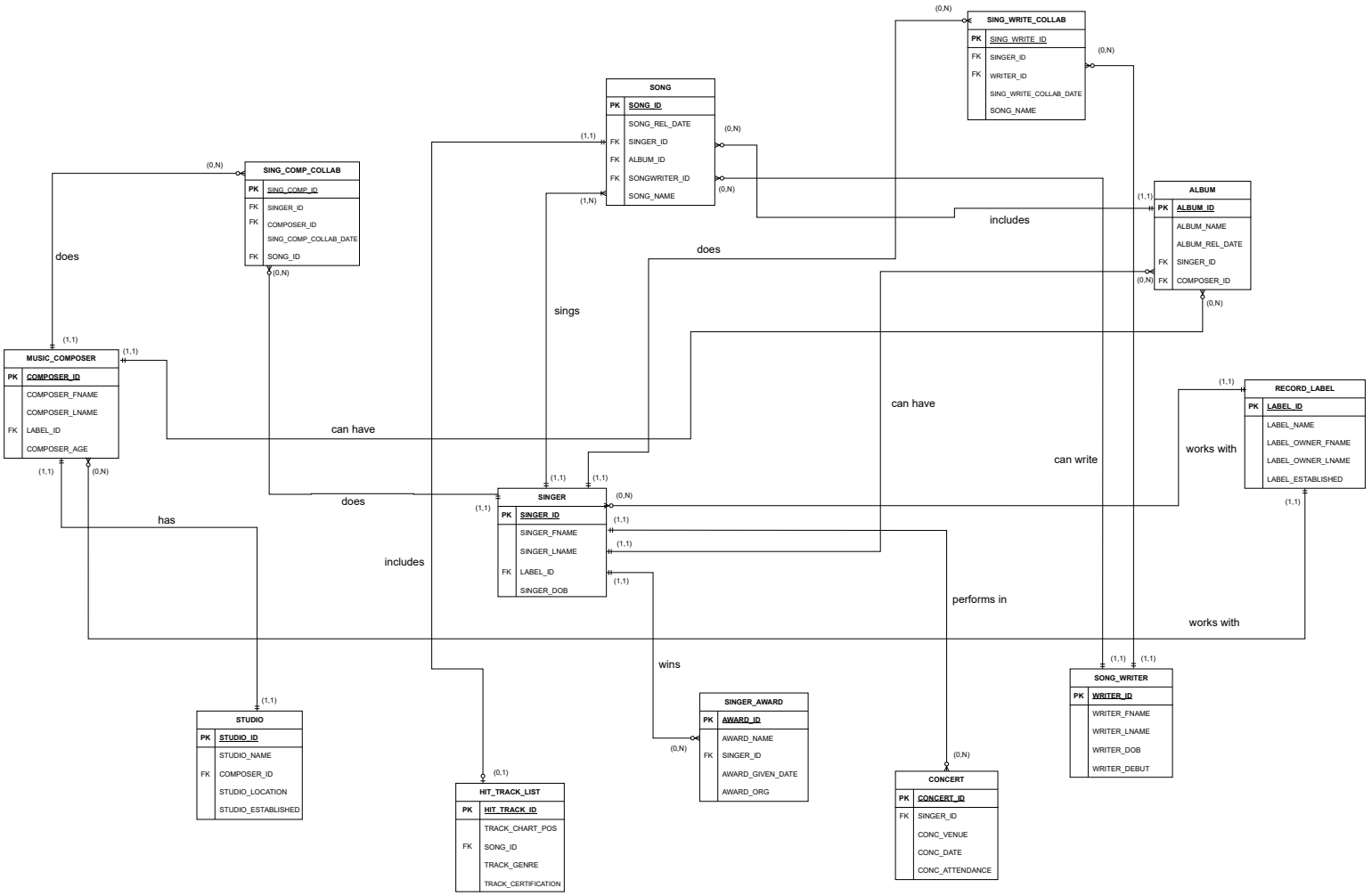
(**SINGER_ID,** SINGER_LNAME)

COMPOSER_ID	COMPOSER_FNAME

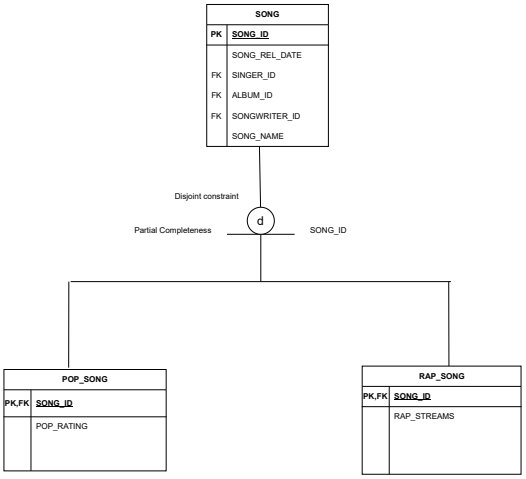
COMPOSER

(**COMPOSER_ID,** COMPOSER_FNAME)

Now, we've successfully moved to 3NF. SING_COMP_COLLAB can have any appropriate attribute to it.



Example Subtypes and Supertypes below



Three questions and appropriate SQL queries

Q1: *Who is the youngest singer?*

SQL: SELECT SINGER_FNAME, SINGER_LNAME

FROM SINGER

ORDER BY SINGER_DOB DESC

LIMIT 1;

Q2: *What are the names of singers and the labels they're signed to?*

SQL: SELECT SINGER.SINGER_FNAME, RECORD_LABEL.LABEL_NAME

FROM SINGER

JOIN RECORD_LABEL

ON SINGER.LABEL_ID = RECORD_LABEL.LABEL_ID ;

Q3: *What are the names of singers who received an award after 2020?*

SQL: SELECT SINGER.SINGER_LNAME, SINGER.SINGER_FNAME

FROM SINGER

WHERE SINGER.SINGER_ID IN (

SELECT SINGER_AWARD.SINGER_ID

FROM SINGER_AWARD

WHERE SINGER_AWARD.AWARD_GIVEN_DATE > '2020-12-31') ;