Spotify Database
Phase 2

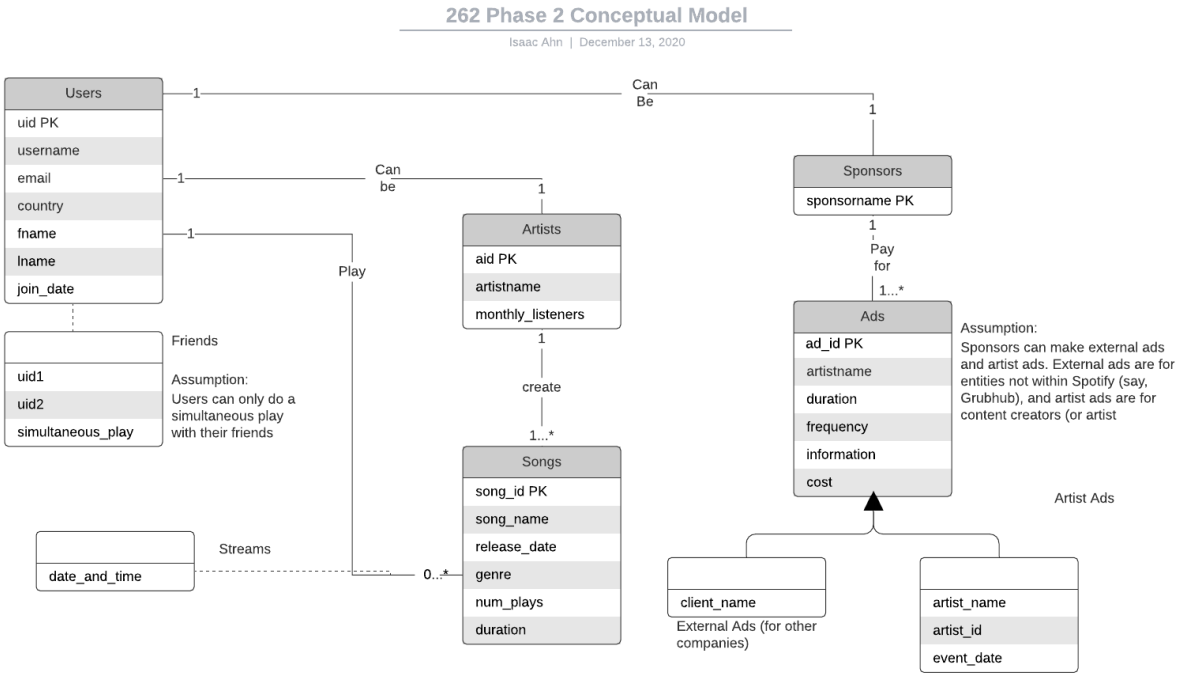
Isaac Ahn and Lisa Leung

December 13, 2020

Updated Conceptual Model

Attached below as a PNG and at the end of the document as a PDF for your convenience.

We minimized our scope for this iteration, which resulted in removing several extraneous tables.



User Stories

Changed significantly from Phase 1 to reduce the amount of tables needed and to model the very basic functions of Spotify on a surface level.

File Name	ID	Simple/ Complex/ Analytical	As a <role>	I want <goal>	So that <reason>
simple_query_1.py	US1	Simple	Artist	To post songs	I can gain revenue and followers
analytical_query_1.py	US2	Analytical	Artist	To see how many people stream each of my songs	I know what kind of content I should create in the future and what my followers like
analytical_query_2.py	US3	Analytical	Artist	To see how many songs I have posted	I know how much work I've committed to Spotify
simple_query_2.py	US4	Simple	Artist	To take down my content	I can post my content with a different company
complex_query_1.py	US5	Complex	Listener	To find out information about the song and content creator	I can listen to similar content in the genre and listen to more from the content creator
simple_query_3.py	US6	Simple	Listener	To friend my	I can listen to music

				friends on Spotify and listen simulataneoulslt	with them and hang out virtually
analytical_query_3.py	US7	Analytical	Listener	To find out how many times I've listened to a song	I can see what I'm doing with my time
simple_query_4.py	US8	Simple	Listener	To listen to songs simultaneously with my friends	We can enjoy high-quality music at the same time regardless of distance
complex_query_2.py	US9	Complex	Sponsor	To advertise on the homepage	Increase my client base and have more people use my goods and services (external company)
complex_query_3.py	US10	Complex	Sponsor	To advertise new releases for my content creator	They can get more followers (increasing followers and royalties) and specifically reach out to their current followers
complex_query_4.py	US11	Complex	Sponsor	Find out how much my ads have cost me in the past	I can budget accordingly and decide how I'll advertise on Spotify

					in the future

Relational Model

Entities

Users (uid, username, email, country, fname, lname, join date, **artist_id, sponsor_id**)
Artists (**artist_id**, artist_name, monthly_listeners)
Sponsors (**sponsor_id,** sponsor_name)
Songs (song_id, release_date, genre, num_plays, length, **artist_id**)
Ads (ad_id, duration, frequency, information, cost, **sponsor_id**)

Associations

Friends (**uid1, uid2,** simultaneous_play)
Stream (stream_id, **uid, song_id**, date, time)

Generalizations

External Ads: (**ad_id**, client_name)
Artist Ads: (**ad_id**, artist_name, artist_id, event_date)

Functional Dependencies

Entities

uid → uid, username, email, country, fname, lname, join_date
artist_id → artist_id, artistname, monthly_listeners, uid
sponsor_id → sponsor_id, sponsor_name, uid
song_id → song_id, song_name, release_date, genre, num_plays, duration, artist_id
ad_id → ad_id, duration, frequency, information, cost, sponsor_id

Associations

uid1, uid2 → uid1, uid2, simultaneous_play
uid, song_id → uid, song_id, date, time

Generalizations

ad_id → ad_id, client_name
ad_id → ad_id, artist_id, event_date

Normalization

Users Table

Columns: uid, username, email, country, fname, lname, join_date
uid → uid, username, email, country, fname, lname, join_date
{uid}+: uid, username, email, country, fname, lname, join_date

The Users table is in BCNF by definition; the LHS (uid) will obtain every column in the table.

Artists

Columns: artist_id, artistname, monthly_listeners, uid
artist_id → artist_id, artistname, monthly_listeners, uid
{artist_id}+: artist_id, artistname, monthly_listeners, uid

The Artists table is in BCNF by definition; the LHS (artist_id) will obtain every column in the table.

Sponsors

Columns: sponsor_id, sponsor_name, uid
sponsor_id → sponsor_id, sponsor_name, uid
{sponsor_id}+: sponsor_id, sponsor_name, uid

The Sponsors table is in BCNF by definition; the LHS (sponsor_id) will obtain every column in the table.

Songs

Columns: song_id, song_name, release_date, genre, num_plays, duration, artist_id
Song_id → song_id, song_name, release_date, genre, num_plays, duration, artist_id
{song_id}+: song_id, song_name, release_date, genre, num_plays, duration, artist_id

The Songs table is in BCNF by definition; the LHS (song_id) will obtain every column in the table.

Ads

Columns: ad_id, duration, frequency, information, cost, sponsor_id
ad_id → ad_id, duration, frequency, information, cost, sponsor_id
{ad_id}+: ad_id, duration, frequency, information, cost, sponsor_id

The Ads table is in BCNF by definition; the LHS (ad_id) will obtain every column in the table.

Friends

Columns: uid1, uid2, simultaneous_play
Uid1, uid2 → uid1, uid2, simultaneous_play
{uid1 uid2}+: uid1, uid2, simultaneous_play

The Friends table is in BCNF by definition; the LHS (uid1, uid2) will obtain every column in the table.

Stream

Columns: uid, song_id, date, time
Uid, song_id → uid, song_id, date, time
{stream_id}+: uid, song_id, date, time

The Streams table is in BCNF by definition; the LHS (stream_id) will obtain every column in the table.

External Ads

Columns: ad_id, client_name
Ad_id → client_name
{ad_id}+: ad_id, client_name

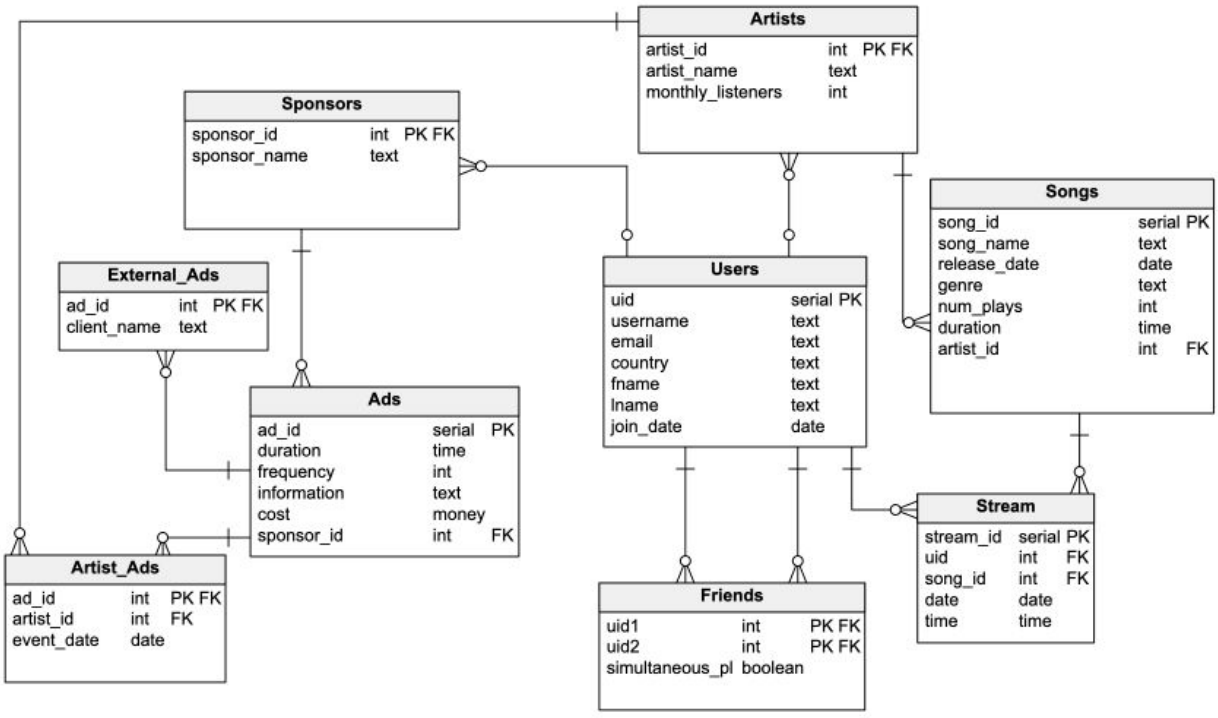
The External Ads table is in BCNF by definition; the LHS (ad_id) will obtain every column in the table.

Artist Ads

Columns: ad_id, artist_id, event_date
Ad_id → ad_id artist_id, event_date
{ad_id}+: ad_id, artist_id, event_date

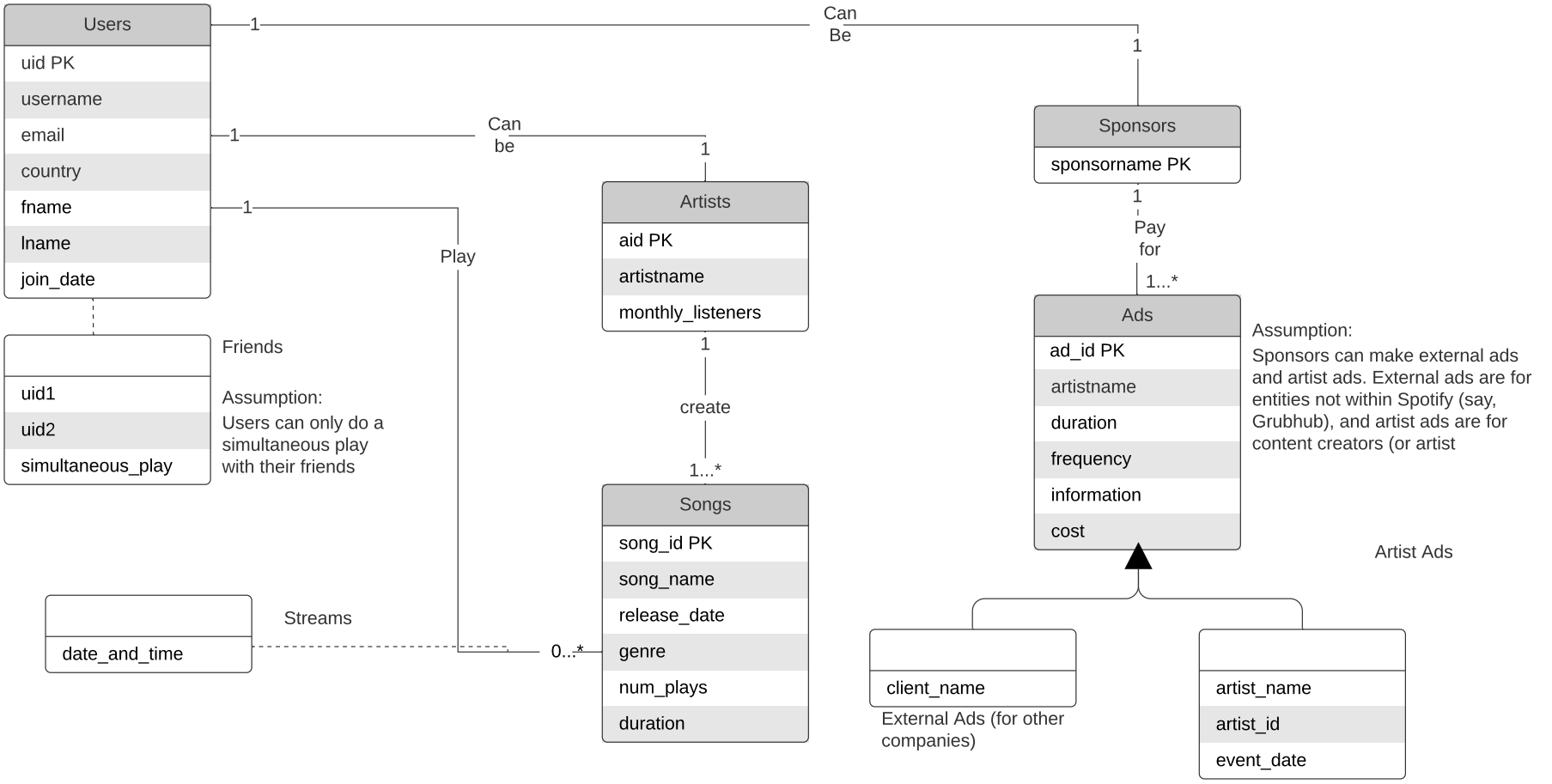
The Artists Ads table is in BCNF by definition; the LHS (ad_id) will obtain every column in the table.

Physical Model



262 Phase 2 Conceptual Model

Isaac Ahn | December 13, 2020



Database model documentation

Table of contents

1. Model details

Model name:

Phase 2

Version:

2.3

Database engine:

PostgreSQL

Description:

2. Tables

2.1. Table Sponsors

2.1.1. Columns

Column name	Type	Properties	Description
sponsor_id	int	PK	
sponsor_name	text		

2.2. Table Ads

2.2.1. Columns

Column name	Type	Properties	Description
ad_id	serial	PK	
duration	time		
frequency	int		
information	text		
cost	money		
sponsor_id	int		

2.3. Table Users

2.3.1. Columns

Column name	Type	Properties	Description
uid	serial	PK	
username	text		
email	text		
country	text		
fname	text		
lname	text		
join_date	date		

2.4. Table Friends

2.4.1. Columns

Column name	Type	Properties	Description
uid1	int	PK	
uid2	int	PK	
simultaneous_play	boolean		

2.5. Table Songs

2.5.1. Columns

Column name	Type	Properties	Description
song_id	serial	PK	
song_name	text		
release_date	date		
genre	text		
duration	time		
artist_id	int		

2.6. Table Artists

2.6.1. Columns

Column name	Type	Properties	Description
artist_id	int	PK	
artist_name	text		
monthly_listeners	int		

2.7. Table External_Ads

2.7.1. Columns

Column name	Type	Properties	Description
ad_id	int	PK	
client_name	text		

2.8. Table Artist_Ads

2.8.1. Columns

Column name	Type	Properties	Description
ad_id	int	PK	
artist_id	int		
event_date	date		

2.9. Table Stream

2.9.1. Columns

Column name	Type	Properties	Description
stream_id	serial	PK	
uid	int		
song_id	int		
date	date		
time	time		

3. References

3.1. Reference Sponsors_Users

Users	0..*	Sponsors
uid	<->	sponsor_id

3.2. Reference Artists_Users

Users	0..*	Artists
uid	<->	artist_id

3.3. Reference Friends_users_1

Users	0..*	Friends
uid	<->	uid1

3.4. Reference Friends_users_2

Users	0..*	Friends
uid	<->	uid2

3.5. Reference Ads_Sponsors

Sponsors	0..*	Ads
sponsor_id	<->	sponsor_id

3.6. Reference Songs_Artists

Artists	0..*	Songs
artist_id	<->	artist_id

3.7. Reference Artist_Ads_Ads

Ads	0..*	Artist_Ads
ad_id	<->	ad_id

3.8. Reference External_Ads_Ads

Ads	0..*	External_Ads
ad_id	<->	ad_id

3.9. Reference Stream_Users

Users	0..*	Stream
uid	<->	uid

3.10. Reference Stream_Songs

Songs	0..*	Stream
song_id	<->	song_id

3.11. Reference Artist_Ads_Artists

Artists	0..*	Artist_Ads
artist_id	<->	artist_id