DBM1: Databases
Spotify data dive

Hongjie Zheng
Luca Nardone

Dataset and Preprocessing

Top 10000 Songs on Spotify 1950-Now kaggle

- `9999 rows x 35 columns in a single table` → redundancy and irregularities

  ➗ **`SPLITTING NORMALIZATION`** :subdivision into three main entities

SONG		
TRACK URI	TRACK NAME	ALBUM URI

ARTIST	
ARTIST URI	ARTIST NAME

ALBUM			
ALBUM URI	ALBUM NAME	RELEASE DATE	ARTIST URI

- `Raw dataset without any insertion rules` → redundancy, missing values and incorrect Format

  🧹**`DATA CLEANING:`**

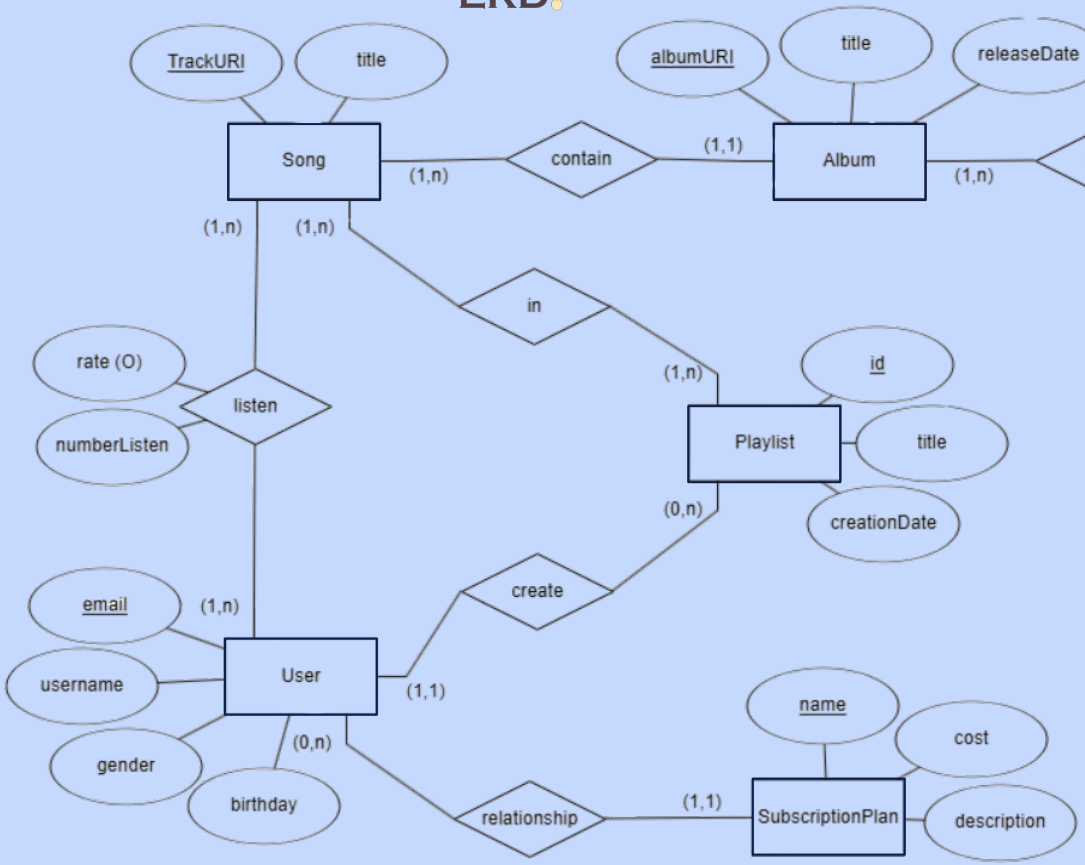
Data Formatting
Standardized the format of the data to prevent discrepancies

Eliminating duplicates
Removed duplicates to ensure that each entity was uniquely represented

Handling missing values
Eliminated records with missing critical fields, preserving data integrity

ER diagram

ERD



Relational Schema

Song(<u>trackUri</u>, title, *album_uri*)
User (<u>email</u>, username, gender, birthday, *subscription_plan*)
SubscriptionPlan(<u>name</u>, cost, description)
Playlist(<u>id</u>, title, creationDate, *email_user*)
Album(<u>albumUri</u>, title, releaseDate, *artist_uri*)
Artist(<u>artistUri</u>,name)
Listen(*<u>user_email</u>*, *<u>song_uri</u>*, rate(o), numberListen)
Playlist_Song(*<u>playlist_id</u>*, *<u>song_uri</u>*)

Query & relational Algebra

Albums released between 2016 and 2020

```
SELECT albumURI, title, releaseDate
FROM album
WHERE releaseDate BETWEEN '2016-01-01' AND '2020-12-31';
```

$$\pi_{\text{albumURI,title,releaseDate}}\sigma_{\text{2016-01-01}\leq\text{releaseDate}\leq\text{2020-12-31}}(\text{album})$$

OR

```
SELECT albumURI, title, releaseDate
FROM album
WHERE releaseDate >= '2016-01-01' AND releaseDate <= '2020-12-31';
```

$$\pi_{\text{albumURI,title,releaseDate}}\left(\sigma_{\text{releaseDate}\geq\text{2016-01-01}\wedge\text{releaseDate}\leq\text{2020-12-31}}(\text{album})\right)$$

Find all users of the "Female" genre who have created at least one playlist

```
SELECT email, username, gender
FROM users
WHERE gender = 'Female' AND email IN (SELECT email FROM playlist);
```

$$\sigma_{\text{gender}=\text{`Female'}}(\text{User}) \cap \pi_{\text{email}}(\text{Playlist})$$

The most popular subscription plan

```
SELECT subscription_plan,
COUNT(*)

AS user_count

FROM "users"

GROUP BY subscription_plan

ORDER BY user_count DESC

LIMIT 1;
```

The number of album created by all the artists

```
SELECT a.name AS artist_name,
COUNT(al.albumuri) AS num_albums

FROM artist a

JOIN album al ON a.artisturi =
al.artist_uri

GROUP BY a.name

ORDER BY num_albums DESC;
```

Get the most 3 popular album based on the total number of listens for its songs, including the album ID, title, and artist name

```
SELECT a.albumuri AS album_id, a.title AS
album_title, ar.name AS artist_name,

SUM(l.number_listen) AS total_listens

FROM Album a

JOIN Song s ON s.album_uri = a.albumuri

JOIN listens l ON s.TrackURI = l.song_uri

JOIN Artist ar ON ar.artisturi = a.artist_uri

GROUP BY a.albumuri, a.title, ar.name

ORDER BY total_listens DESC LIMIT 3;
```

Performance Improvements

Index

SELECT * FROM album WHERE title = 'Let It Bleed';

default

	Planning time(ms)	Execution time(ms)
NO index	4.049	22.968
WITH index: B_tree	1.624	0.173
WITH index: HASH	0.527	0.099

SELECT * FROM album WHERE releaseDate > '2020-01-01';

default

	Planning time(ms)	Execution time(ms)
NO index	0.098	1.309
WITH index: B_tree	0.091	0.475
WITH index: HASH	0.219	1.320

Transaction

```
BEGIN TRANSACTION;

INSERT INTO artist (artistUri, name)
VALUES ('spotify:artist:3ALm6zJLaJMWV0r89kuYtu', 'Modà');

INSERT INTO album (albumUri, title, releaseDate, artist_uri)
VALUES ('spotify:album:2euaairB4BCeCBHrpeMqHu', 'Viva i Romantici',
'2011-02-16', 'spotify:artist:3ALm6zJLaJMWV0r89kuYtu');

INSERT INTO song (trackUri, title, album_uri)
VALUES
('spotify:track:2KDUheuY5UkgATQvR2K3un', 'Come un pittore',
'spotify:album:2euaairB4BCeCBHrpeMqHu'),
('spotify:track:5sUeQNphyv55ywYlARruNb', 'La notte',
'spotify:album:2euaairB4BCeCBHrpeMqHu'),
('spotify:track:25RA3QmKxNTof4hk3tlnnf', 'Tappeto di fragole',
'spotify:album:2euaairB4BCeCBHrpeMqHu'),
('spotify:track:0HHCJVAwOjEckFKFQ5aqZO', 'Arriverà',
'spotify:album:2euaairB4BCeCBHrpeMqHu'),
('spotify:track:06DSeOsqgLf3ao7mMMB28l', 'Sono già solo',
'spotify:album:2euaairB4BCeCBHrpeMqHu');

COMMIT;
```

Integrity Constraints

```
ALTER TABLE "users"
ADD CONSTRAINT check_birthdate
CHECK (birthday < CURRENT_DATE);

ALTER TABLE "listens"
ADD CONSTRAINT unique_user_song_pair
UNIQUE (user_email, song_uri);
```

Wrong insertion:

```
INSERT INTO "users" (email, username, gender,
birthday, subscription_plan)
VALUES ('futureuser@example.com', 'FutureUser',
'Female', '2050-01-01', 'Free');
```

ERROR: Failing row contains (futureuser@example.com, FutureUser, Female, 2050-01-01, Free).new row for relation "users" violates check constraint "check_birthdate"

Thanks for your attention

Any Question?