# Introduction to Databases and Data Modelling

Oliver Gregory

2024-10-04

## Activity 3

1. Do you agree with the modelling and existing relationships among albums, artists, tracks, playlists, genres, and media_types? Can you spot any missing attributes in those entities?

- The relationship between between albums and tracks has albums having a minimum of 0 tracks with no maximum. I would disagree with this as an album should really have at least one track. Similarly, tracks are limited to a minimum of 0 albums and a maximum of 1. This is arguably incorrect as a track should belong to at least one album but could also belong to many.
- The relationship between artists and albums is incorrect as it allows an artist to be associated with 0 albums on the database. This means it is possible to have redundant artists in the database.
- The relationship between genres and tracks is incorrect as it allows a genre to be associated with 0 tracks. This means it is possible to have redundant genres in the database. Tracks are limited to a minimum of 0 genres and a maximum of 1. This is arguably incorrect as a track should belong to at least one genre but could also belong to many.
- The relationship between media_types and tracks is incorrect as it allows a media type to be associated with 0 tracks. This means it is possible to have redundant media types in the database.
- The relationship between playlist track and playlist allows a playlist track to only be associated with 1 playlist. This could be wasteful as it requires tracks that are in multiple playlists have separate 'playlist_tracks' for each playlist.
- There is a missing path attribute in the tracks entity that should point to where the track itself is stored.

2. Can you explain why the attribute Bytes is included in the tracks table?

The tracks entity will refer to a file where the track itself is stored. This file will have a certain size that will be stored in the bytes attribute. This attribute allows the database user to query how many bytes of storage all the tracks that are stored use for example.

3. Do you realise that the table playlist_track is used to capture an M:N relationship between playlists and tracks (one playlist has several tracks and one track can pertain to several playlists)? Do you imagine a different way of modelling and/or implementing this relationship?

You could have a table called 'playlists' that stores a unique playlist_id and playlist_name. Then you could have a multivalued attribute in the track entity that can be optional and would include all of the playlist_ids that a the playlists the track is in pertain to.

4. Looking at invoice_items, do you spot any inconsistency (missing or unnecessary attribute)?

The unit_price attribute is unnecessary as that information is already stored in the track entity. Instead it could simply be referenced to in the trackId.

5. Can you explain/imagine the relationship between employees and customers? What practical implications does this relationship bring when registering a new customer or removing an existing employee?

The relationship between employees and customers is a 1:M relationship. This means that one employee can have many customers but one customer can only have one employee. This relationship is useful as it allows the database to store which employee is responsible for a customer. This could be useful for example if a customer has a complaint, the employee responsible for that customer can be easily identified. When registering a new customer, the employee responsible for that customer must be specified. When removing an existing employee, the customers that are associated with that employee must be reassigned to another employee.

Other things to consider might be that an employee could be a customer themselves. This would require a self-referencing relationship between employees and customers. Instead you could have a separate entity called 'person' that stores all the information about a person and then have a relationship between employees and persons and customers and persons.