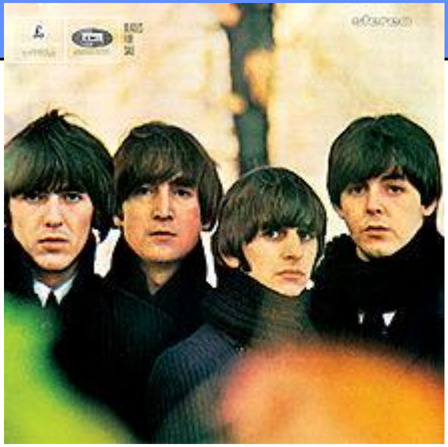
Impedence Miss-match between Data and Object Models

Sample database and UML Object Models based on The Beatles studio albums (for 2404 A1 and A2)

PARLOPHONE
PLEASE PLEASE ME • THE BEATLES
stereo
THE BEATLES
PLEASE
PLEASE ME
with Love Me Do
and 12 other songs

PARLOPHONE
with
the
beatles
stereo

PARLOPHONE
THE BEATLES
stereo
A HARD DAY'S NIGHT


BEATLES
FOR
SALE
stereo

PARLOPHONE
THE BEATLES
stereo
HELP!

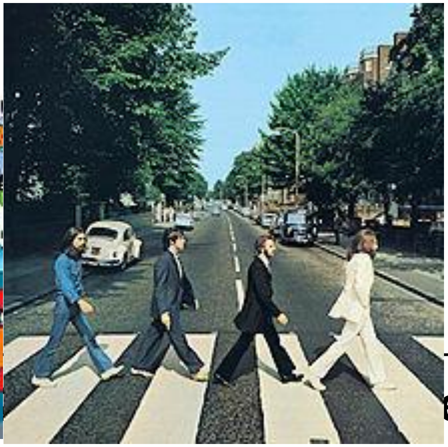
RUBBER
SOUL

REVOLVER

SGT. PEPPERS
LONELY HEARTS
CLUB BAND
BEATLES

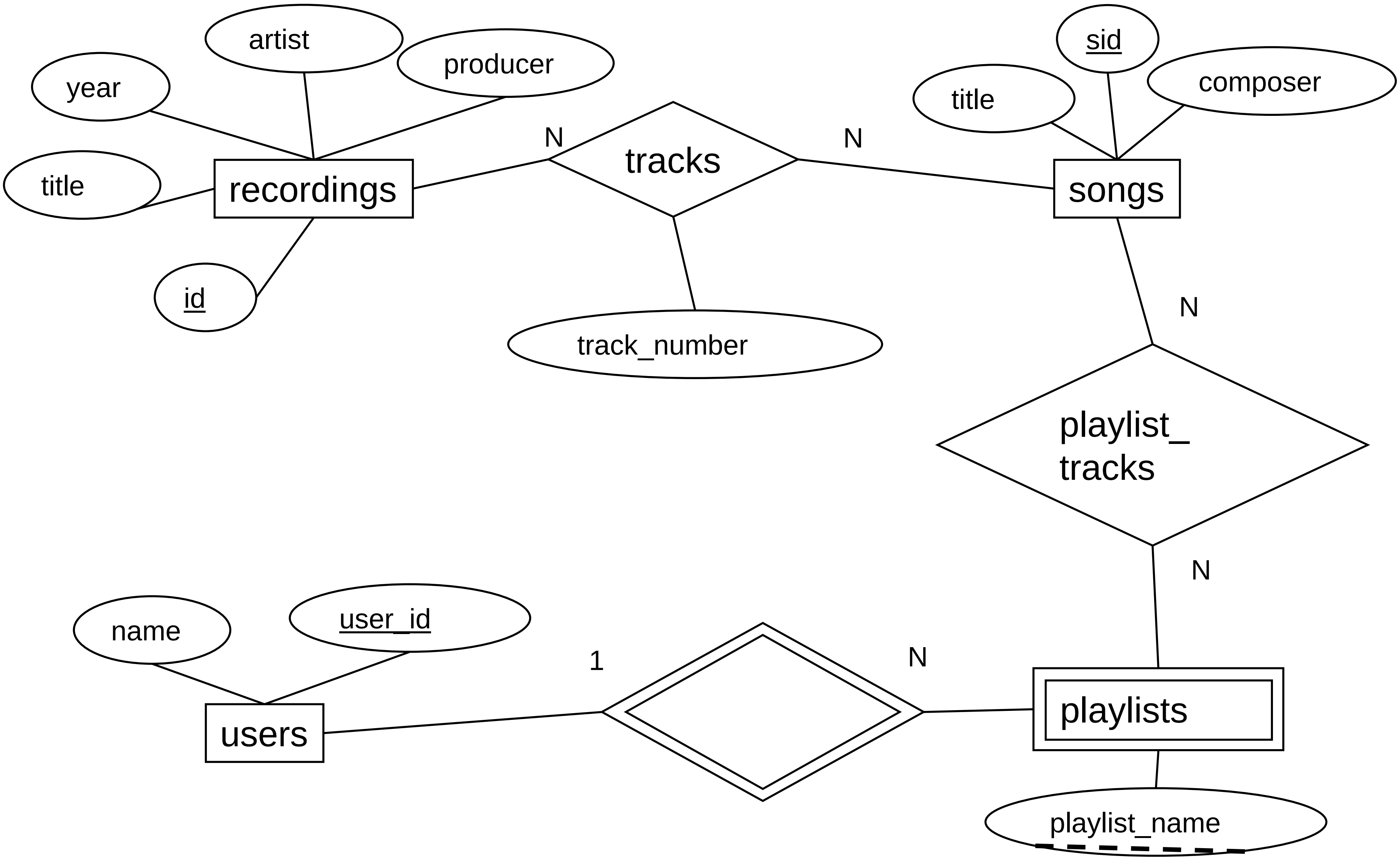
The BEATLES
0000001

The Beatles
Yellow Submarine

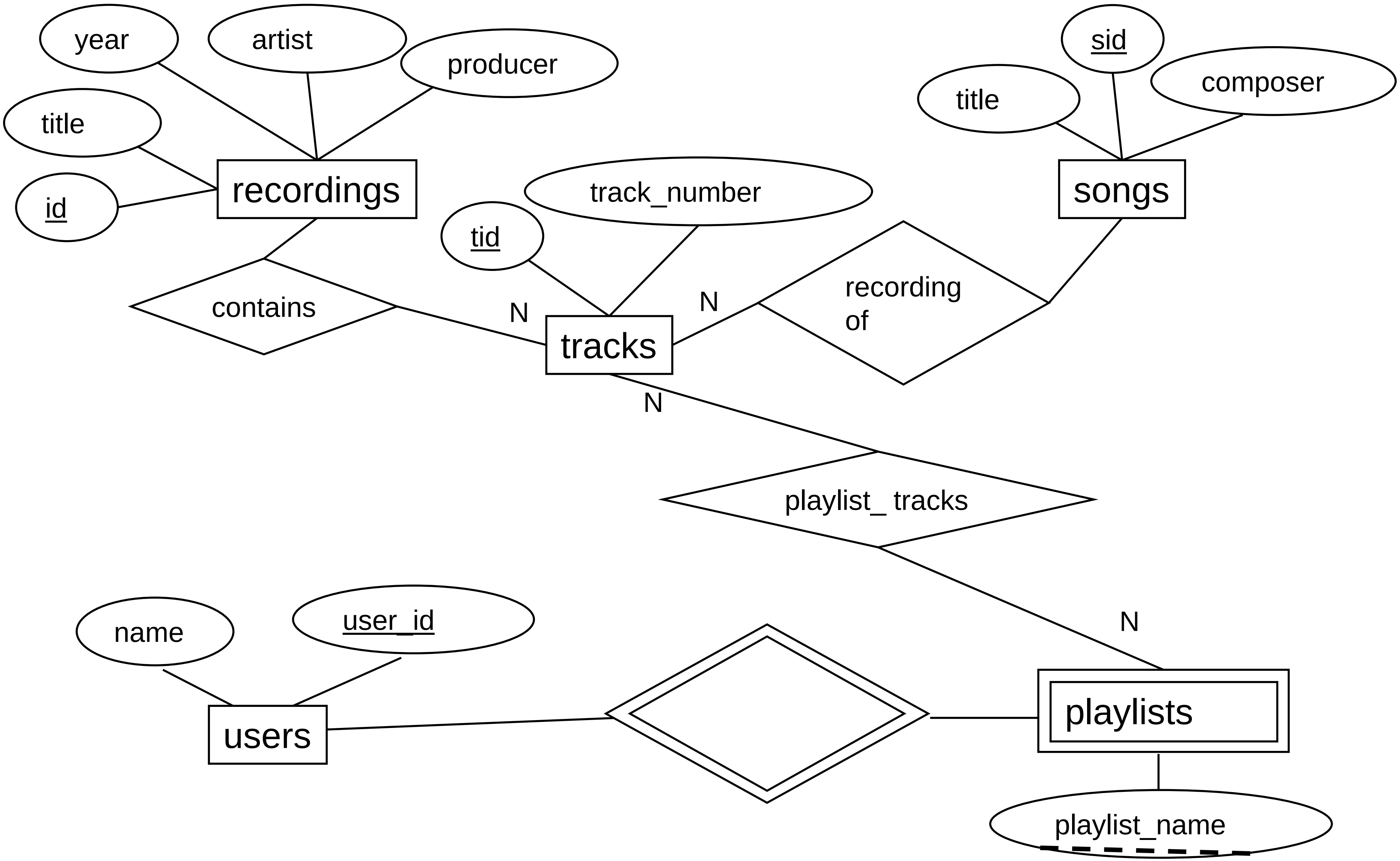


LET IT BE

E-R diagram from Assignment #1



Refactored ER Model for Assignment #2



Map ER Model to Tables (Direct Mapping)

- Entities become tables, their attributes define the columns and primary keys
- Weak entities become tables and their keys are the combination of their own descriminator and the key from the strong entity they are related to.
- N:N relationships become new tables
- 1:N and 1:1 relationships can be handled by adding additional columns to existing tables
- Identify foreign keys (columns in one table that refer to the key of another table)
- Decide on the data types for each column
- Write to SQL to CREATE the table

…Map ER Model to Tables

recordings				
id	title	artist	year	producer

songs		
sid	title	composer

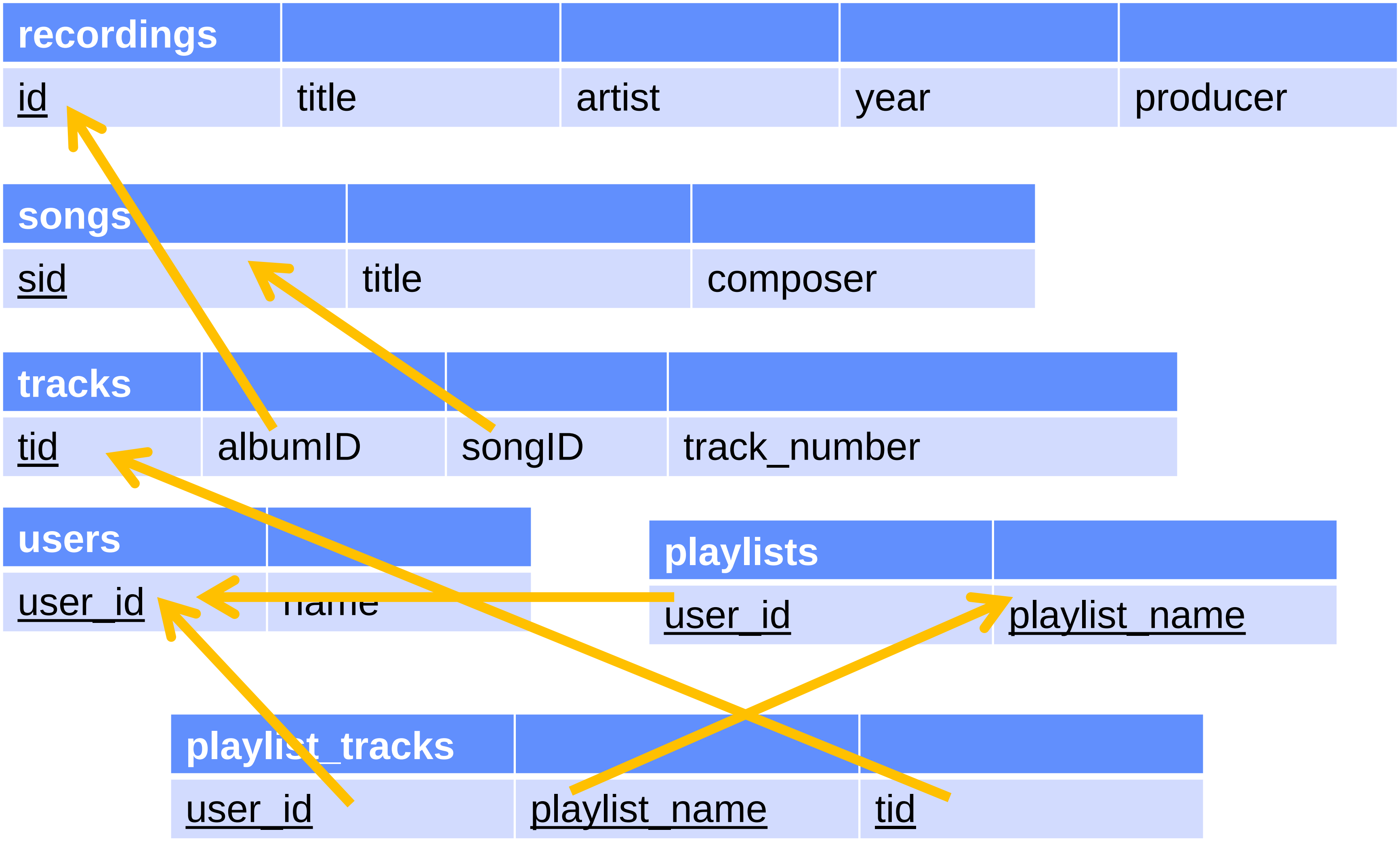
tracks			
tid	albumID	songID	track_number

users	
user_id	name

playlists	
user_id	playlist_name

playlist_tracks		
user_id	playlist_name	tid

References and Referential Integrity



recordings				
id	title	artist	year	producer

songs		
sid	title	composer

tracks			
tid	albumID	songID	track_number

users	
user_id	name

playlists	
user_id	playlist_name

playlist_tracks		
user_id	playlist_name	tid

SQL CREATE TABLE

recordings				
id	title	artist	year	producer

```
create table recordings(
      id int primary key not null,
      title varchar(30 ) NOT NULL,
      artist varchar(30) NOT NULL,
      producer varchar(30) default NULL,
      year int
      );
```

SQL CREATE TABLE

songs		
sid	title	composer

```
create table songs (
    sid int primary key,
    title varchar(30) NOT NULL,
    composer varchar(30) NOT NULL
);
```

SQL CREATE TABLE

tracks			
tid	albumID	songID	track_number

```
create table tracks (
    tid ingeter NOT NULL primary key,
    albumID int ~~NOT NULL~~
        references recordings(id),
    songID int NOT NULL references songs(sid),
    track_number int NOT NULL,
);
```

SQL CREATE TABLE

users	
user_id	name

```
create table users (
    user_id varchar(30) primary key not null,
    name varchar(30) not null
);
```

SQL CREATE TABLE

playlists	
user_id	playlist_name

```
create table playlists(
    user_id varchar(30)
        not null references users(user_id),
    playlist_name varchar(30) not null,
    primary key (user_id,playlist_name)
);
```

SQL CREATE TABLE

playlist_tracks		
user_id	playlist_name	tid

```
create table playlist_tracks(
   user_id varchar(30) not null,
   playlist_name varchar(30) not null,
   track_id int not null
      references tracks (tid),
   primary key
(user_id,playlist_name,track_id),
   foreign key (user_id,playlist_name)
references playlists(user_id,playlist_name),
);
```

"Impedence" Miss-match between Data and OO Models

Data Model (Relational Database)

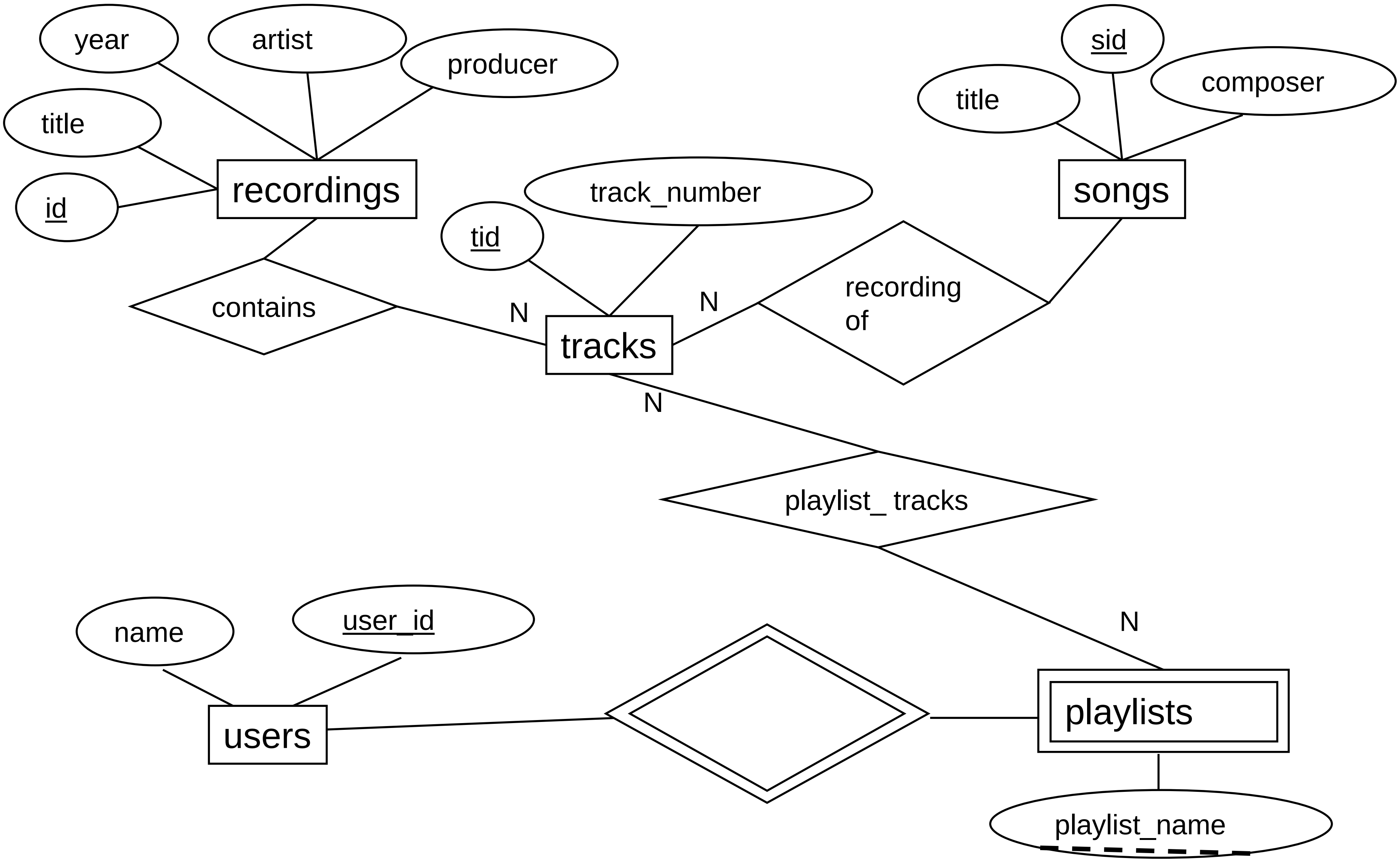
- Avoids pointers (at all costs) and uses data attributes (e.g. ID's) instead.
- References are from dependents to parents. (opposite to direction of likely navigation)

OO Object Models (in memory objects)

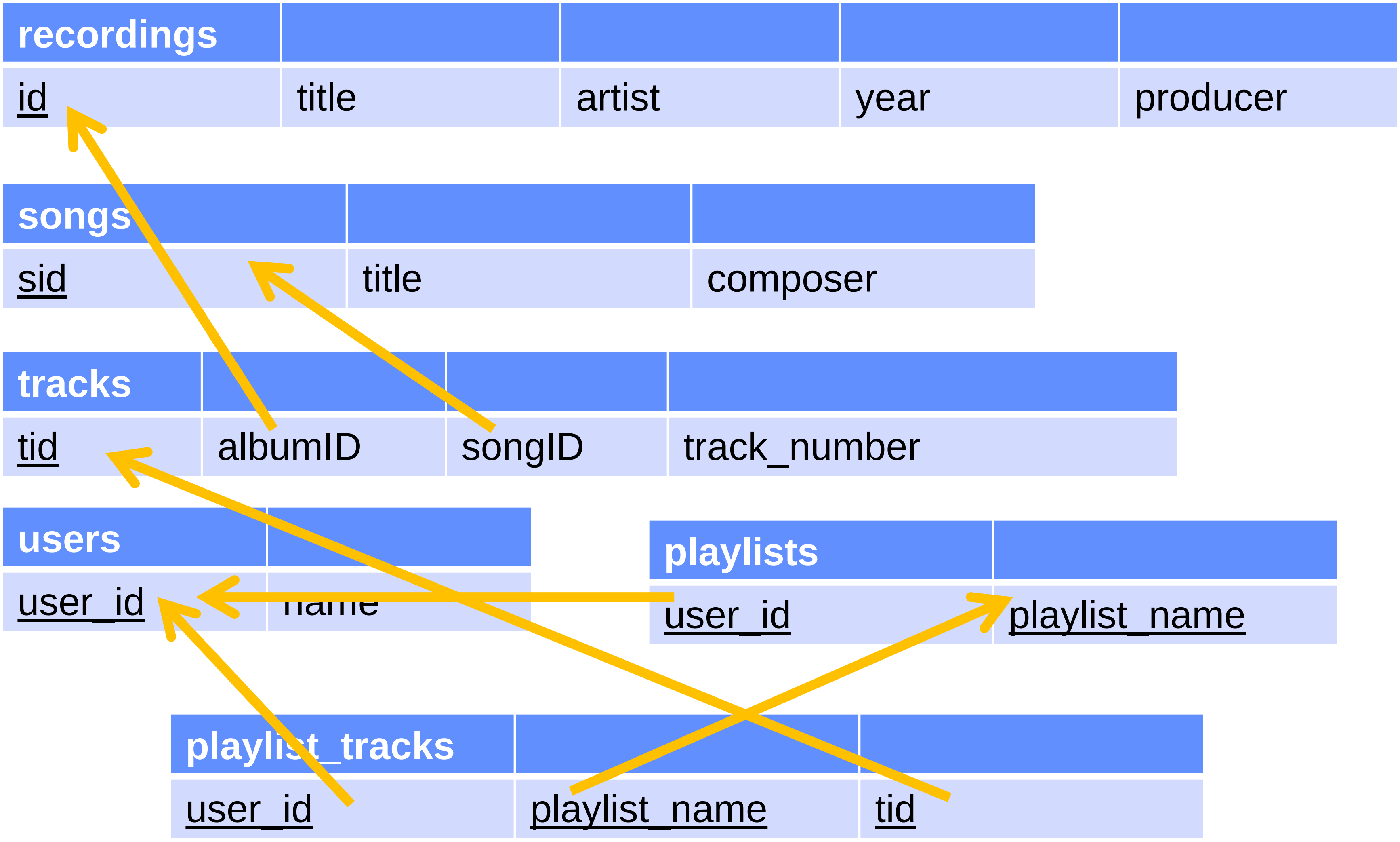
- Uses pointers for references
- References are from parents to dependents (in direction of likely navigation).

- Referential Integrity: Must never refer to a non-existing object or data.

Refactored ER Model for Assignment #2



References



recordings				
id	title	artist	year	producer

songs		
sid	title	composer

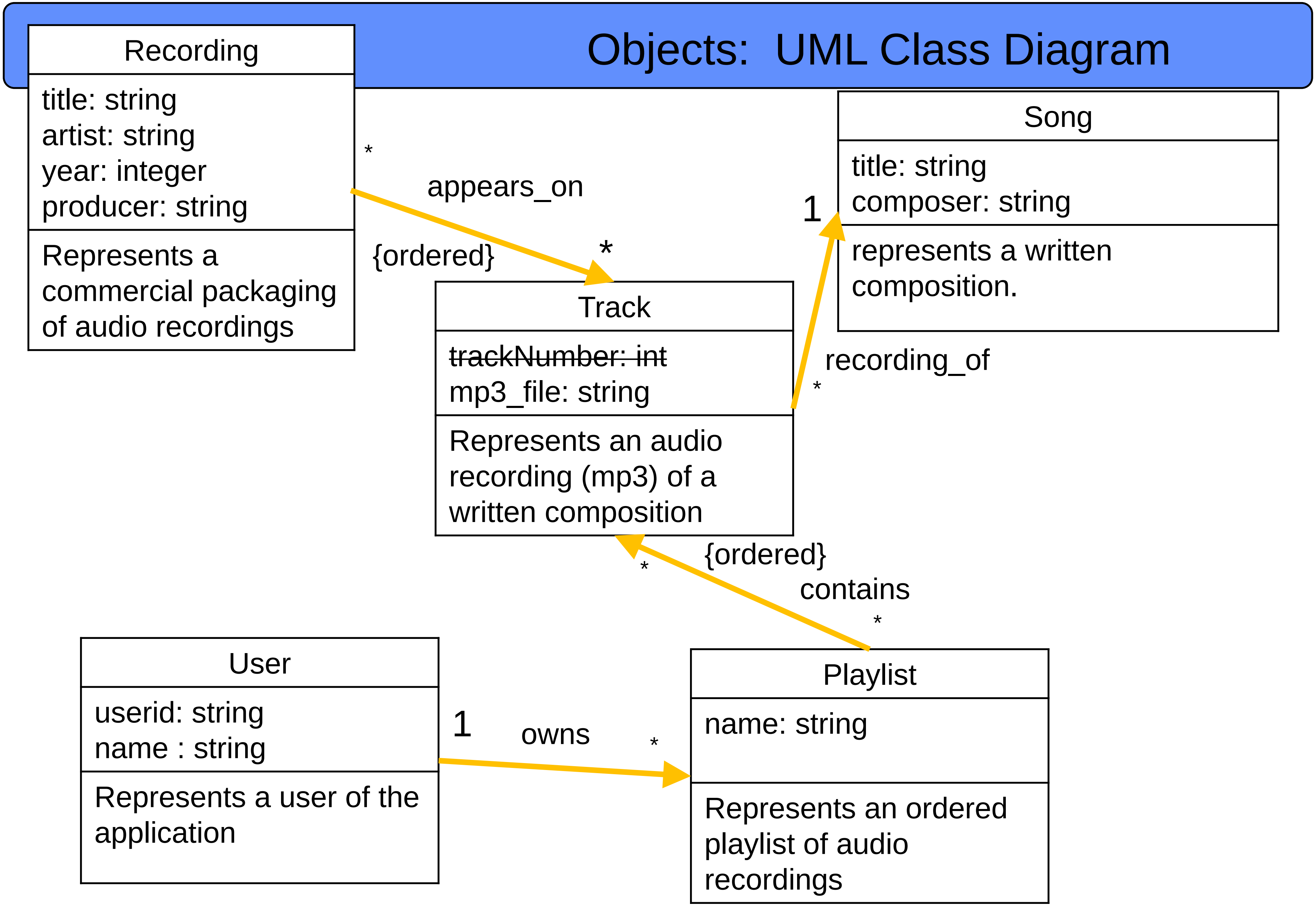
tracks			
tid	albumID	songID	track_number

users	
user_id	name

playlists	
user_id	playlist_name

playlist_tracks		
user_id	playlist_name	tid

Objects: UML Class Diagram



Recording

title: string
artist: string
year: integer
producer: string

Represents a commercial packaging of audio recordings

* appears_on {ordered} *

Track

~~trackNumber: int~~
mp3_file: string

Represents an audio recording (mp3) of a written composition

recording_of * 1

Song

title: string
composer: string

represents a written composition.

{ordered} contains * *

Playlist

name: string

Represents an ordered playlist of audio recordings

1 owns *

User

userid: string
name : string

Represents a user of the application