

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA

FRANCISCO KNEBEL
MATEUS SALVI

Spotify:
Representação em um modelo ER

Monografia apresentada como relatório de trabalho
na disciplina de Fundamento de Banco de Dados.

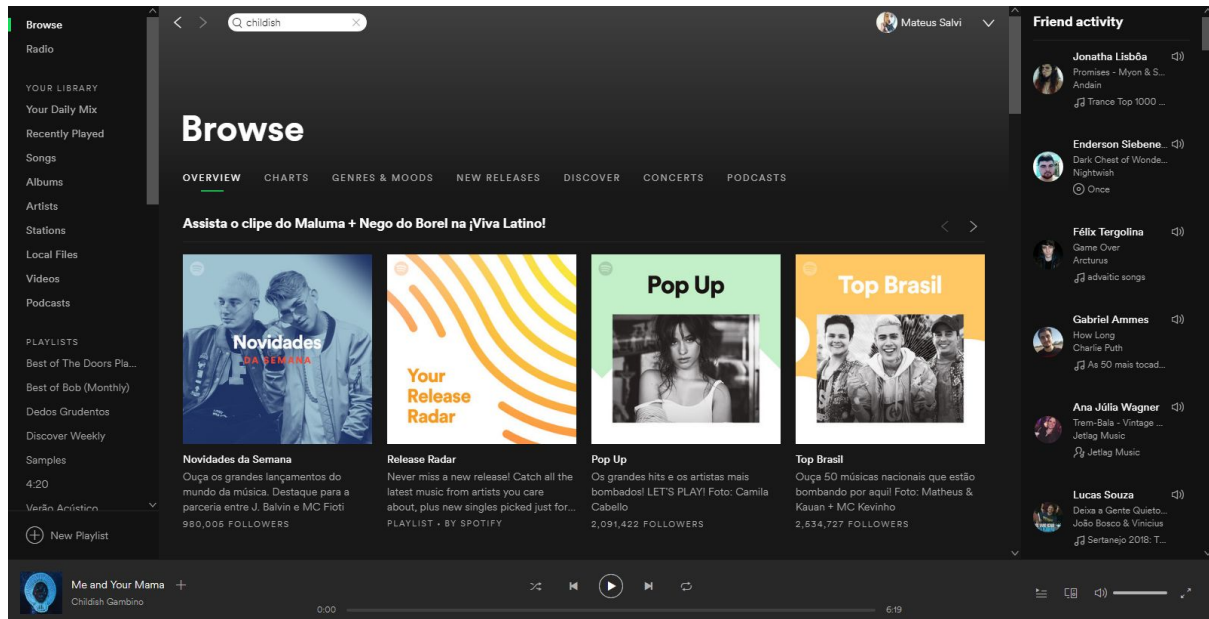
Orientador: Profa. Dra. Karin Becker

Porto Alegre
2017

FRANCISCO KNEBEL
MATEUS SALVI
Spotify:
Representação em um modelo ER
Universo de Discurso

Orientador: Profa. Dra. Karin Becker
15/12/2017

Universo de Discurso:



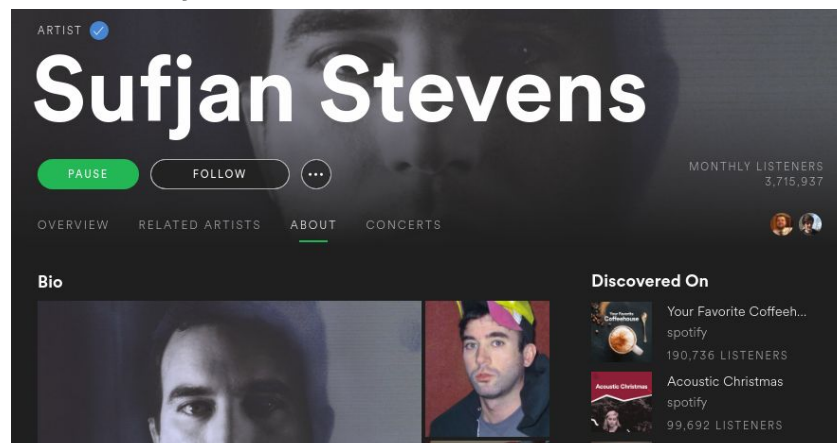
Usando como inspiração a aplicação *Spotify* (<https://www.spotify.com/br>), que permite usuários ouvirem músicas e podcasts através de um banco de faixas de diversos artistas, ver vídeos relacionados à eles, descobrir shows em datas e locais próximos e saber um pouco sobre o artista com breves descrições e histórias. Usuários também podem criar suas próprias playlists, com sua própria seleção de músicas, além de poder acompanhar playlists criadas por outras pessoas.

Os usuários se dividem entre usuários comuns, do tipo 0, que devem ouvir faixas de propagandas entre algumas reproduções de músicas (possuindo também algumas limitações que não são relevantes ao nosso UdD), e usuários premium, do tipo 1, que não escutam propagandas mas devem possuir uma assinatura mensal, além de outras funcionalidades extras previstas na aplicação.

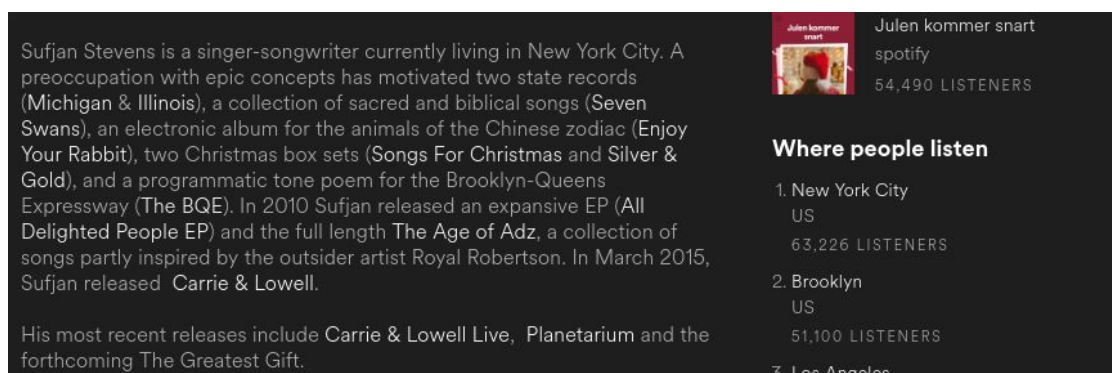
Cada usuário tem seu id único e utiliza apenas um método de pagamento formado pelo número do cartão de crédito, data de expiração, código de segurança, o dia da renovação mensal do status premium para cobrança, data dos pagamentos feitos para controle, seu nome completo para apresentação e um único e-mail para contato. Informações adicionais como cidade e país são úteis à fim de relacionar com onde as músicas são mais ouvidas criando recomendações regionais para outras pessoas. Cada nova reprodução de uma música é armazenado, de forma que assim conseguimos, de forma algorítmica, montar o gosto musical de cada usuário, para fins de recomendação de novas músicas, que o usuário não conhece ou que pode ter ouvido no passado, mas esquecido.




Playlists podem ser criadas por usuários com ou sem músicas dentro, podendo ser individuais, único criador, ou colaborativas, com mais de um criador. É possível aos usuários seguir a playlist criada por outros usuários. Cada playlist possui pelo menos um usuário criador e um título (nome da playlist dado pelo usuário), além de uma lista de músicas com a data de adição na mesma.



Artistas, em nossa aplicação, são pessoas produtoras de conteúdo. Artistas podem produzir álbuns de músicas, vídeos e podcasts. Todo artista pode inserir uma biografia para descrever um histórico de forma breve. A aplicação fornece opções de playlists onde músicas do artista podem ser encontradas, possibilitando encontrar músicas similares, e também informações como cidades onde ele é mais ouvido, concertos realizados e músicas mais ouvidas.





ALBUM

Random Access Memories

By Daft Punk
May 17, 2013 • 13 songs, 1 hr 14 min

PLAY SAVE ...

#	TITLE	Duration
1	✓ Give Life Back to Music	4:35
2	✓ The Game of Love	5:22
3	✓ Giorgio by Moroder	9:05
4	✓ Within	3:49
5	+ Instant Crush - Julian Casablancas	5:38
6	✓ Lose Yourself to Dance - Pharrell Williams	5:54
7	✓ Touch - Paul Williams	8:19
8	✓ Get Lucky - Pharrell Williams, Nile Rodgers	6:10

Álbuns são criados por um único artista, em uma data, contendo uma seleção de músicas, com seu título, ordem e duração. Um álbum pode ser categorizado por múltiplos gêneros diferentes e cada música pode ter participações de artistas não relacionados com o álbum, como indicado na imagem acima.



PODCAST

Naruhodo

By B9

PLAY FOLLOW ...

ABOUT

Naruhodo! é o podcast pra quem tem fome de aprender. Ciência, senso comum, curiosidades e muito mais. Com o leigo curioso, Ken Fujioka, e o cientista PhD, Altay de Souza.

Filtered by All Episodes ▾

TITLE	PROGRESS	DATE	DURATION
Naruhodo #109 - O cérebro humano é um computador?	<div></div>	Dec 11, 2017	32:05
Naruhodo #108 - Bebida alcoólica ajuda a falar melhor uma língua ...	<div></div>	Dec 4, 2017	27:48

Podcasts funcionam de forma parecida com álbuns, mas funcionam utilizando uma série de episódios que são incrementados ao longo do tempo, não sendo um elemento estático. Podcasts possuem um nome e uma descrição e cada episódio possui um título, uma duração e sua devida data de adição.

FRANCISCO KNEBEL

MATEUS SALVI

Spotify:
Representação em um modelo ER
Dicionário de Dados

Orientador: Profa. Dra. Karin Becker

15/12/2017

Dicionário de dados:

{Entidade: chave [cardinalidade]: tipo(tamanho) / (descrição)

relação(Representa uma chave estrangeira): [cardinalidade] com entidade relacionada }

Álbum:

id_album: int not null / (Chave primária para diferenciar cada álbum)

título: varchar (60) not null / (Nome do álbum)

data_lançamento: date not null / (Data de lançamento do álbum, não de inserção no banco)

id_radio: int / (Chave para indicar a especialização em uma rádio)

relação: [0-N] com Categorização / (O álbum deve ter se encaixar em pelo menos uma categoria)

relação: [1-N] com Álbum Composição / (O álbum é composto de pelo menos uma música)

relação: [1-1] com Álbum Produção / (Relação com o usuário que compôs o álbum)

Anunciante:

id_anunciante: int not null / (Chave primária para diferenciar cada anunciante)

nome: varchar (50) not null / (Nome de cada anunciante, não exclusivo)

relação: [0-N] com Publicação / (Um anunciante pode ou não ter anúncios)

Artista:

id_artista: int not null / (Chave primária para diferenciar cada artista)

nome: varchar (60) not null / (Nome de cada artista)

biografia[0-1]: varchar (500) / (Uma breve descrição/história do artista)

id_radio: int / (Chave para indicar a especialização em uma rádio)

relação: [0-N] com Podcast Produção / (O artista pode ou não produzir podcasts)

relação: [0-N] com Vídeo Produção / (O artista ou não pode produzir videos)

relação: [0-N] com Participação / (O artista pode ou não participar de outras músicas)

relação: [0-N] com Álbum Produção / (O artista pode ou não produzir álbuns)

relação: [0-N] com Apresentação / (O artista pode se apresentar ou não em um concerto)

Concerto:

id_show: int not null / (Chave primária para diferenciar cada show/concerto)
titulo: varchar (60) not null / (Título/nome de cada concerto/show)
local: varchar (60) not null / (Local onde ocorrerá o concerto/show)
preço: decimal(9,2) not null / (Preço dos ingressos para comparecer ao concerto)
data e hora: datetime not null / (Data e hora do concerto/show)
relação: [1-1] com Apresentação / (Cada concerto corresponde à apresentação de um artista)

Gênero:

id_genero: int not null / (Chave primária para diferenciar cada gênero)
nome: varchar (60) not null / (Nome de cada gênero)
id_radio: int / (Chave para indicar a especialização em uma rádio)
relação: [1-N] com Categorização / (Cada gênero pode categorizar um ou mais álbuns, se não existe um álbum de determinado gênero, o gênero não existe)

Música:

id_musica: int not null / (Chave primária para diferenciar cada música)
titulo: varchar (60) not null / (Título da música)
duração: int not null / (Duração de tempo da música, quantos segundos ela tem)
contador: int (64) not null / (Contador de reproduções, começa em 0, útil para saber a música mais reproduzida)
número: int not null / (Número da música em seu álbum)
id_radio: int / (Chave para indicar a especialização em uma rádio)
relação: [1-N] com Playlist Composição / (Relação para adicionar a música na playlist de um usuário)
relação: [1-N] com Rádio Composição / (A música pode estar em uma rádio)
relação: [0-N] com Reprodução / (Relação com um usuário para saber quando e se ele está reproduzindo esta música)
relação: [1-1] com Álbum Composição / (Relação com o artista compositor)
relação: [1-N] com Participação / (Relação com um artista não produtor da mesma que fez parte da música)

Pagamentos:

data: date not null / (Data do pagamento para renovação do status premium)
valor: decimal (9,2) not null / (Valor pago pelo usuário)
relação: [1-1] in Histórico / (Cada pagamento é uma tabela única no histórico do usuário)

Playlist:

id_playlist: int not null / (Chave primária para diferenciar cada playlist)
título: varchar (60) not null / (Nome dado por um usuário para sua playlist)
relação: [0-N] com Playlist Seguir / (Uma playlist pode ter de 0-N seguidores)
relação: [0-N] com Playlist Criação / (Cada playlist pode ter mais de um criador chamadas de playlists colaborativas)
relação: [0-N] com Playlist Composição / (Uma playlist não precisa de músicas para existir e pode ter N músicas)

Podcast:

id_podcast: int not null / (Chave primária para diferenciar cada podcast)
nome: varchar (50) not null / (Nome do podcast dado pelo artista)
sobre: varchar (300) not null / (Pequena descrição sobre o assunto do podcast)
id_artista: int not null / (Id do artista produtor do podcast)
relação: [1-1] com Podcast Produção / (Cada podcast é produzido por apenas um artista)
relação: [1-N] com Podcast Composição / (Um podcast é composto de pelo menos um episódio e no máximo N)

Podcast Episódio:

id_episodio: int not null / (Chave primária para diferenciar cada episódio)
nome: varchar (100) / (Nome do episódio do podcast)
duração: num (1) / (Duração em segundos que o episódio tem)
data: date (1) / (Data de divulgação do episódio na plataforma)
id_podcast: int not null / (Id do podcast qual o episódio está incluído)
relação: [1-1] com Podcast Composição / (Cada episódio só pode pertencer à um podcast)

Propaganda:

id_propaganda: int not null / (Chave primária para diferenciar cada propaganda)
título: varchar (50) not null / (Nome da propaganda)
duração: bigint not null / (Tempo de execução da propaganda em segundos)
reproduções: bigint not null / (Contador de reproduções, começa em 0, para os anunciantes saberem o alcance de suas propagandas)
investimento: decimal(9,2) not null / (Valor pago pelos anunciantes para ter sua propaganda no banco)
relação: [1-1] com Anunciante / (Cada propaganda pertence à um anunciante)

Rádio:

id_radio: int not null / (Chave primária para diferenciar cada rádio)
titulo: varchar (60) not null / (Nome da radio)
Album: int / (Código do album em que a rádio foi criada)
Musica: int / (Código da musica em que a rádio foi criada)
Genero: int / (Código do gênero em que a rádio foi criada)
Artista: int / (Código do artista em que a rádio foi criada)
relação: [1-N] in Rádio Composição / (Relação com músicas que compõem uma rádio)

Usuário:

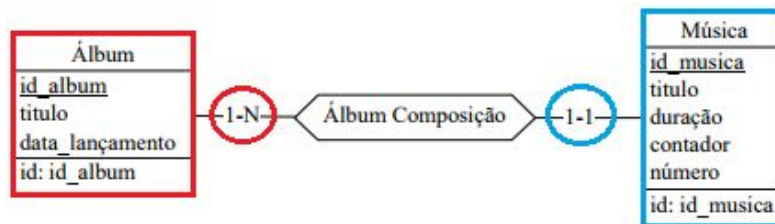
id_usuario: int not null / (Chave primária para diferenciar cada usuário)
nome: varchar (60) not null / (Nome de exibição do usuário)
senha: varchar (60) not null / (Senha de acesso do usuário)
email: varchar (60) not null / (E-mail para contato)
cidade[0-1]: varchar (60) / (Cidade de residência, para relacionar com musicas e shows)
pais[0-1]: char (2) / (Sigla de dois caracteres do país de residência)
info_pagamento[0-1]: compound (21)
nro_cartão: char (16) / (Informações do método de pagamento do usuário)
data_expiração: char (7) / (Informações do método de pagamento do usuário)
cod_segurança: char (3) / (Informações do método de pagamento do usuário)
data_renovação: char (2) / (Dia para renovação do pagamento)
data_criação: date not null / (Data em que o usuário entrou no sistema)
tipo: num (1) / (Usuários podem ter dois tipos de conta, grátis (representada aqui por 0) ou premium (representadas por 1))
relação: [0-N] com Playlist Seguir / (Um usuário pode ou não seguir quantas playlists quiser)
relação: [0-N] com Playlist Criação / Um usuário pode ou não criar quantas playlists quiser)
relação: [1-1] com Reprodução / (Cada usuário só pode ouvir uma musica por vez)
relação: [0-N] com Histórico / (Histórico de pagamentos para renovação do status premium)

Vídeo:

id_video: int not null / (Chave primária para diferenciar cada vídeo)
nome: varchar (50) not null / (Titulo do video)
duração: int not null / (Tempo de execução do vídeo em segundos)
data_adição: date not null / (Data em que o video foi adicionado no banco)
sobre[0-1]: char (300) / (Descrição sobre do que se trata o video)
id_artista: int not null / (Id do artista que produziu o vídeo)
relação: [1-1] com Video Produção / (Relação com o artista produtor do video)

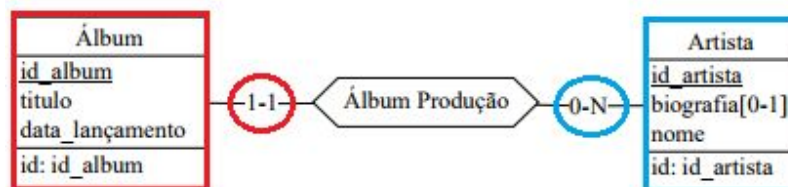
Relações:

Álbum Composição:



([1-N] : Álbum [1-1] : Música) / (Cada álbum é composto de pelo menos uma música, cada música só pode aparecer uma vez em cada álbum)

Álbum Produção:



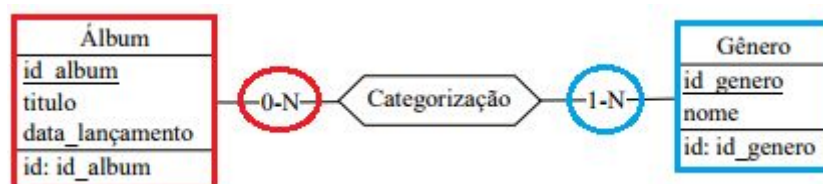
([1-1] : Álbum [0-N] : Artista) / (Cada álbum é produzido por apenas um artista, artistas não precisam ter um álbum no banco e podem ter quantos quiserem)

Apresentação:



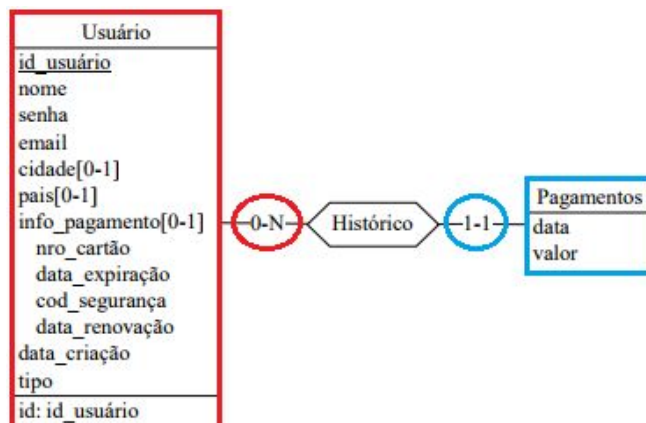
([1-1] : Concerto [0-N] : Artista) / (Cada concerto tem um artista e cada artista pode ou não ter um concerto)

Categorização:



([1-N] : Gênero [0-N] : Álbum) / (Cada álbum tem pelo menos um gênero e cada gênero não precisa de um álbum no banco para existir)

Histórico:



([1-1] : Pagamentos [0-N] : Usuário) / (Cada pagamento do usuário é único, mas o usuário pode ou não ter pagamentos)

Musica Participação:



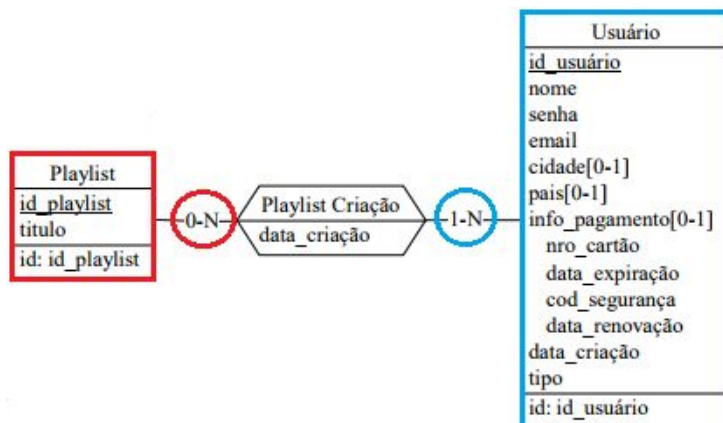
([0-N] : Artista [1-N] : Música) / (Cada música pode ter a participação de artistas, diferentes do artista que produziu o álbum)

Playlist Composição:



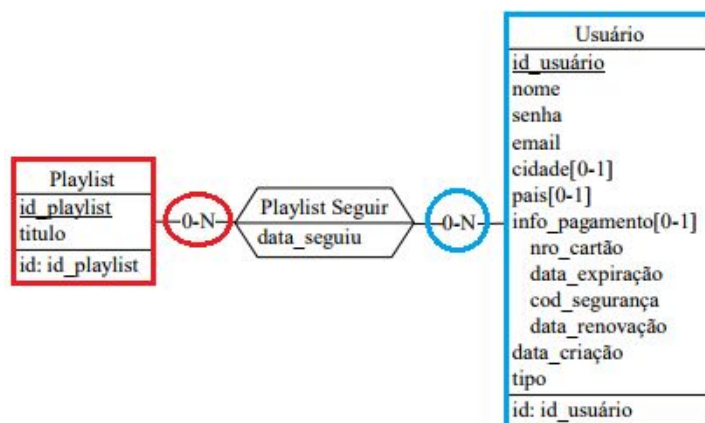
([0-N] : Playlist [0-N] : Música data_adição: date) / (Cada playlist é ou não composta de músicas)

Playlist Criação:



([0-N] : Playlist [0-N] : Usuário data_criação: date) / (Cada usuário pode ou não criar playlists e cada playlist é criada por pelo menos um usuário, podendo ser colaborativa)

Playlist Seguir:



([0-N] : Playlist [0-N] : Usuário data_seguiu: date) / (Cada usuário pode seguir ou não playlists e cada playlist possui ou não seguidores)

Podcast Composição:



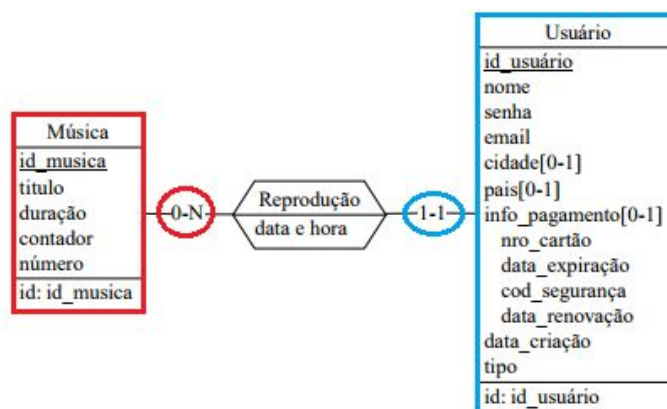
([1-N] : Podcast [1-1] : Podcast Episódio) / (Cada playlist é composta ou não de músicas e músicas podem ou não fazer parte de uma playlist)

Podcast Produção:



([1-1] : Podcast [0-N] : Artista) / (Cada podcast só é produzido por um artista, mas um artista pode ou não produzir podcasts)

Reprodução :



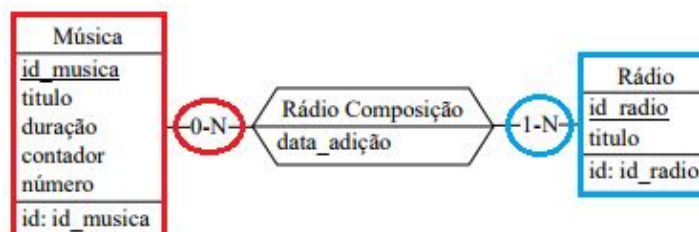
([0-N] : Música [1-1] : Usuário data e hora: datetime) / (Cada reprodução de uma faixa que o usuário faz, para poder comparar seus gostos e fazer sugestões pertinentes)

Publicação:



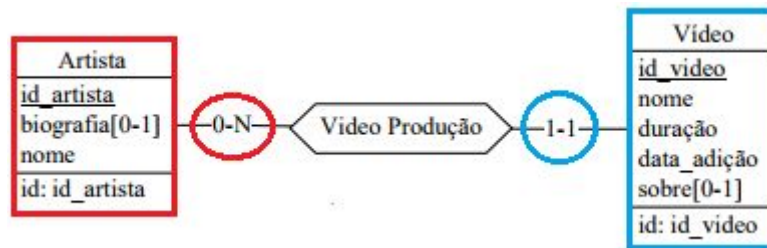
([1-1]: Propaganda [0-N]: Anunciante) / (Anunciantes podem ou não ter propagandas e propagandas só pertencem à um anunciante)

Rádio Composição:



([1-N] : Rádio [1-N] : Música data_adição: date) / (Cada rádio é composta de pelo menos uma música, mas músicas não precisam compor uma rádio)

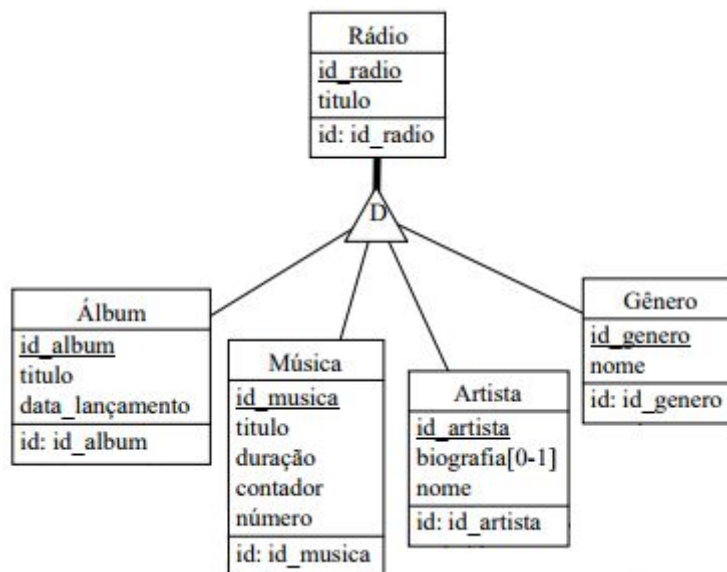
Video Produção:



([1-1] : Vídeo [0-N] : Artista) / (Cada artista produz ou não vídeos e cada vídeo é produzido por apenas um artista)

Especialização:

Rádio: Especialização disjunta, para que cada rádio seja baseada em exatamente uma das seguintes entidades: artista, música, álbum ou gênero musical. Uma rádio é uma seleção de músicas parecidas com o atributo em que ela é baseada.



FRANCISCO KNEBEL

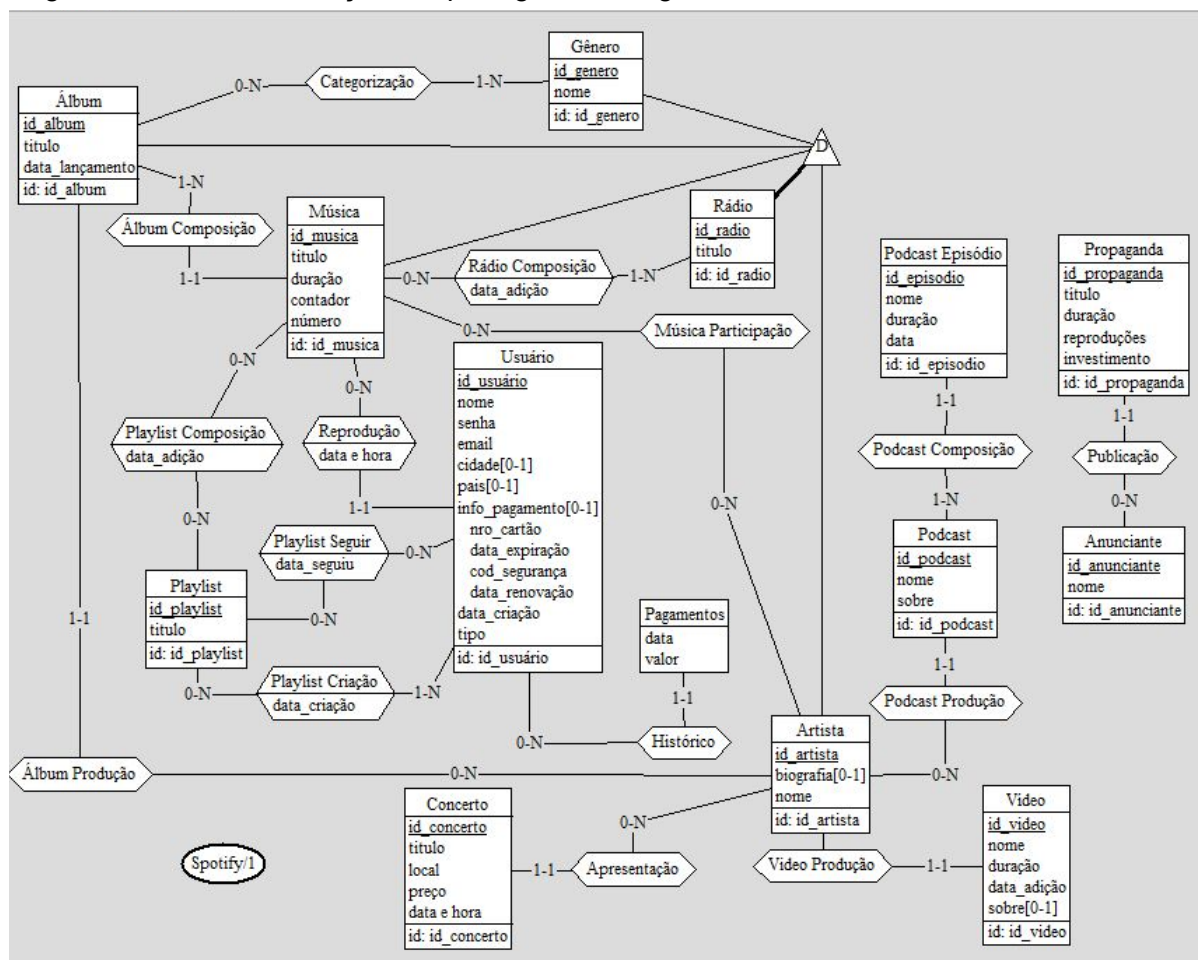
MATEUS SALVI

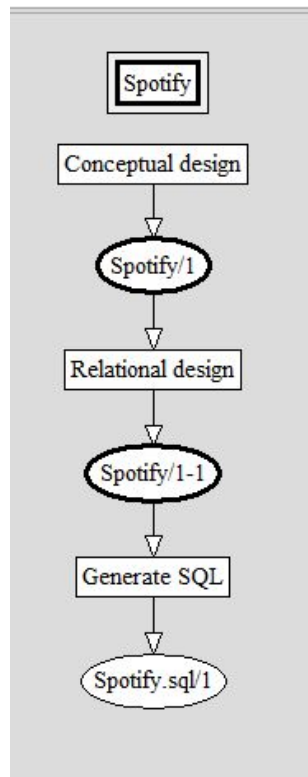
Spotify: Representação em um modelo ER Mapeamento Relacional

Orientador: Profa. Dra. Karin Becker

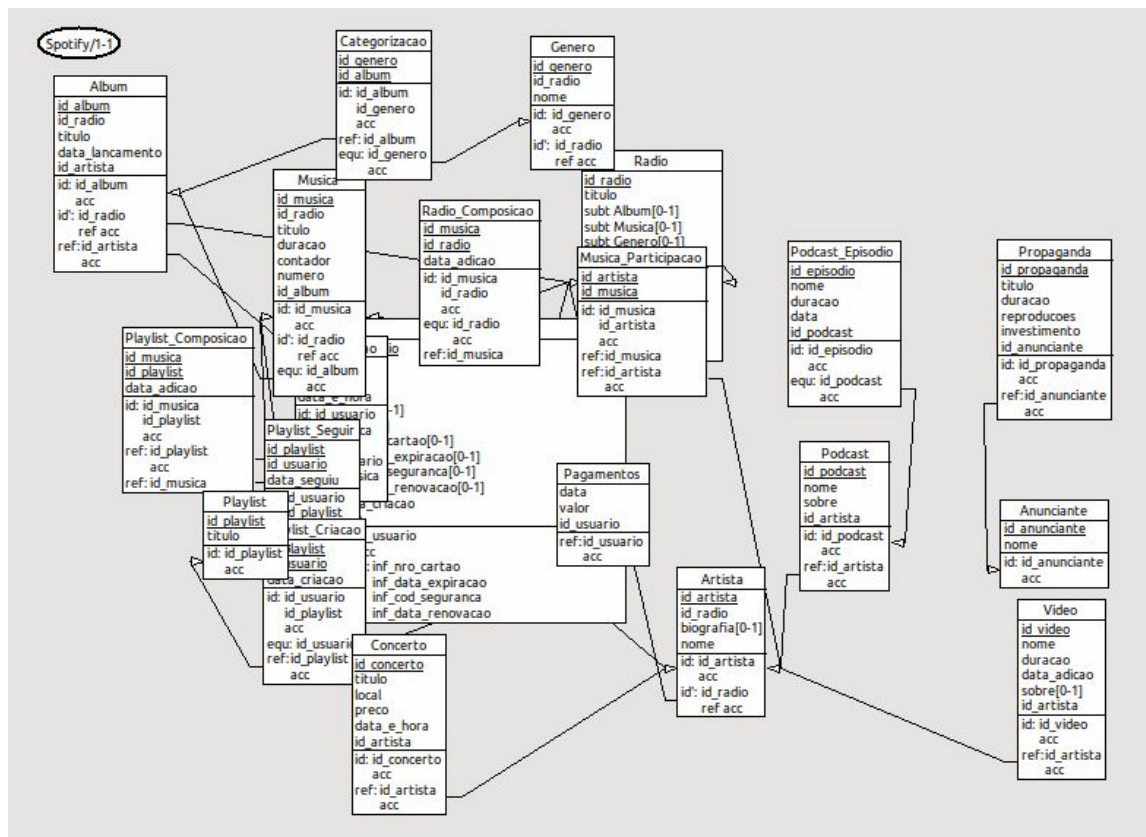
15/12/2017

As regras de transformação usadas neste projeto foram as do programa DB-Main, onde todo o projeto foi desenvolvido. Usando essa ferramenta CASE, montamos um diagrama conceitual das entidades com os seus relacionamentos. Após finalizada essa etapa, o DB-Main possibilita a transformação do diagrama conceitual para sua forma relacional, seguindo as restrições impostas no diagrama, como as cardinalidades, chaves primárias e atributos únicos, tipos compostos e simples, além da especialização criada. Cada relacionamento tem suas cardinalidades ao lado da tabela referente (Não o inverso, como visto em aula). Os tipos das entidades no DB-Main eram incompatíveis com os tipos usados no SQL (como o tipo num, que necessitava ser int). No dicionário de dados foram inseridas informações detalhadas sobre cada entidade, relacionamento e atributo do diagrama com uma descrição em português e imagens relevantes.





A imagem acima descreve o processo efetuado automaticamente pelo DB-Main. O arquivo relacional não segue as disposições físicas definidas no design conceitual, deixando na forma demonstrada abaixo. Com o diagrama relacional, exportamos essa informação usando a opção em 'File > Generate > MySQL...' para um documento .sql, contendo todas as tabelas e restrições de integridade impostas na parte conceitual.



Para incrementar o relacional gerado, inserimos no final do documento *tabelas.sql* dois gatilhos para a base de dados, relacionados com a tabela *Reproducao*.

Um trigger (*incrementaContadorMusica*) foi utilizado para gerar as reproduções das músicas. Toda vez que uma nova inserção é efetuada na tabela (que em nossa aplicação, é quando um usuário reproduz uma música) soma-se 1 ao contador de reproduções da música em específico. Em *instancias.sql*, foram inseridas 2850 reproduções pseudo-aleatórios sobre todo o grupo de músicas, com algumas delas focando em grupos menores para mostrarmos músicas com mais reproduções e uma música em específico com muito mais reproduções que as demais. Foi inserido também um trigger (*decrementaContadorMusica*) para decrementar as reproduções de uma música no caso de remoção de tuplas da tabela, algo necessário na aplicação, mas desnecessário para efeitos da proposta do trabalho, mas testável caso uma tupla de *Reproducao* for removida diretamente do banco de dados.

FRANCISCO KNEBEL

MATEUS SALVI

Spotify: Representação em um modelo ER Aplicação

Orientador: Profa. Dra. Karin Becker

15/12/2017

A aplicação desenvolvida para teste do banco de dados foi criada utilizando NodeJS. Informações para execução estão inseridas no documento README.md do projeto, para inicialização do servidor que irá executar as pesquisas pré-definidas. Após inicializar, o usuário pode acessar a plataforma em *localhost:3000*.

O arquivo *server/api.js* contém os dados da conexão ao banco, como usuário, senha e nome do banco de dados. Lá é efetuada a conexão com o banco de dados, utilizando o pacote *mysql*, disponível no [NPM](#).

```
const mysql = require('mysql');

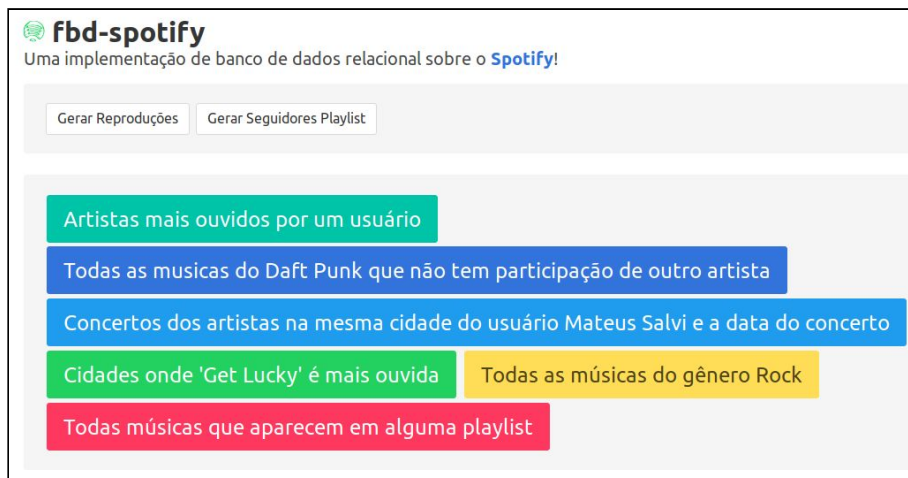
const connection = mysql.createConnection({
  host    : 'localhost',
  user    : 'root',
  password: 'root',
  database: 'Spotify'
});

module.exports = (app) => {
  connection.connect();

  require('./comandos')(app, connection);
}
```

Em *server/comandos.js* estão escritos os seis comandos criados para teste da base de dados proposta. Foi criado um endpoint */query/:i*, onde *i* representa um número de 1 até 6. Ao acessar esse endereço, a aplicação servidor executa um chamado para o banco e retorna para o usuário os resultados recebidos. No lado do cliente, o arquivo *docs/scripts/query.js* que efetua o chamado para o servidor da query desejada, inserindo as informações em uma tabela.

```
app.get('/query/4', (req, res) => {
  // As cidades onde a música 'Get Lucky' é ouvida com o número de reproduções,
  // ordenando pelo número de reproduções em forma decrescente.
  connection.query(
    `
    SELECT count(cidade) as reproducoes, cidade
    FROM Reproducao
    JOIN Musica ON
      (Musica.titulo = 'Get Lucky' AND Reproducao.id_musica = Musica.id_musica)
    JOIN Usuario USING (id_usuario)
    GROUP BY cidade
    ORDER BY reproducoes DESC
    `, (error, results, fields) => {
    res.send({ error, results, fields });
  });
});
```



Para o cliente, ao acessar *localhost:3000*, uma interface com 6 botões foi inserida, e ao selecionar uma opção, é retornado em uma tabela as informações relevantes à pesquisa efetuada, além de quantas tuplas foram retornadas na pesquisa.

Concertos dos artistas na mesma cidade do usuário Mateus Salvi e a data do concerto
Tuplas: 7.

nome	titulo	data_e_hora
Kyuss	Welcome To Sky Valley	2017-12-27T22:40:00.000Z
Led Zeppelin	Led Zeppelin Reunite	2018-01-19T22:00:00.000Z
The Beatles	The Beattles: Love Me Do	2018-01-21T00:30:00.000Z
Black Sabbath	Black Sabbath Reunion	2018-02-02T23:00:00.000Z
Panda Bear	Panda Bear World Tour	2018-02-08T21:00:00.000Z
Nile Rodgers	Nile Rogers South America Tour	2018-02-28T23:45:00.000Z
Julian Casablancas	Julian Casablancas In Brazil	2018-03-06T00:15:00.000Z